بسمه‌تعالی

نکات و راهنمایی‌ها آزمایشگاه مخابرات دیجیتال

استاد: دکتر علی الفت

ویرایش ۱ (آزمایشی)

فهرست مطالب آزمایشگاه مخابرات دیجیتال

بسمه تعالی

آزمایش اول

آشنایی با نرم‌افزار MATLAB

شرح آزمایش

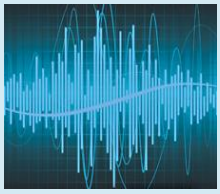
آزمایش ۱-۱: محاسبات جبری و گرافیک

۱.
۲.
۳.

آزمایش ۲-۱: ریاضیات و برنامه‌نویسی

۱.
۲.
۳.
۴.
۵.

بسمه تعالی



آزمایش دوم

پردازش سیگنال دیجیتال با نرم‌افزار MATLAB

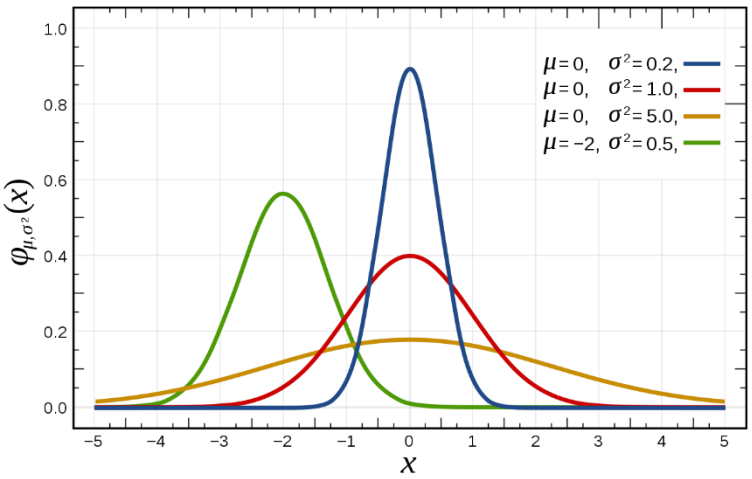
شرح آزمایش

آزمایش ۱-۲: تولید سیگنال دیجیتال و عملیات بر روی آن

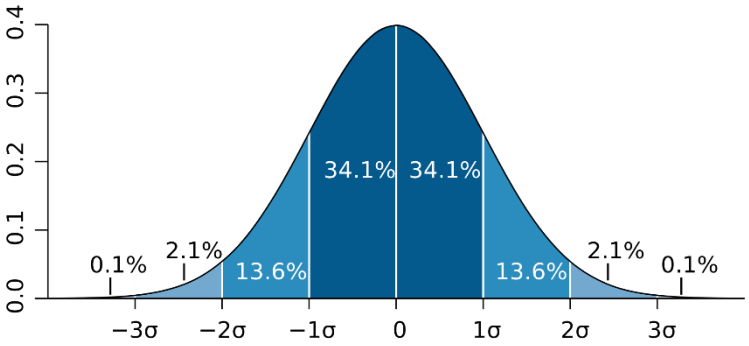
۱. **دنباله‌های تصادفی:** در ابتدای تولید همه‌ی بخش‌ها rng(1) قرار دهید تا نمودارها یکسان باشد.

أ.

تابع چگالی احتمال گاوسی



درصدهای زیر برای یک توزیع گاوسی نرمال می‌توان به خاطر سپرد.



ب. فرض می‌شود x و y دو متغیر تصادفی مستقل با تابع‌های چگالی $f_X(x)$ و $f_Y(y)$ باشند و برای همه‌ی مقادیر x و y تعریف شده باشند. آن‌گاه $z = x + y$ یک متغیر تصادفی با تابع چگالی احتمال $f_Z(z)$ می‌باشد که f_Z کانولوشن یا پیچش f_X و f_Y است.

اثبات:

$$
\begin{aligned}
F_Z(z) &= \mathrm{P}(Z \le z) && \text{(by the definition of distribution function)}\\
&= \mathrm{P}(X+Y \le z)\\
&= \mathrm{P}(X \le z-Y)\\
&= \mathrm{E}[\mathrm{P}(X \le z-Y|Y=y)] && \text{(by the law of iterated expectations)}\\
&= \mathrm{E}[F_X(z-Y)] && \text{(because } X \text{ and } Y \text{ are independent)}
\end{aligned}
$$

$$
\begin{aligned}
& f_Z(z)\\
&= \frac{d}{dz}\mathrm{E}[F_X(z-Y)]\\
&= \mathrm{E}\left[\frac{d}{dz}F_X(z-Y)\right] && \text{(interchanging differentiation and expectation)}\\
&= \mathrm{E}[f_X(z-Y)]\\
&= \int_{-\infty}^{\infty} f_X(z-y)f_Y(y)dy && \text{(by the definition of expected value)}
\end{aligned}
$$

ت.

۲. **کاهش نرخ نمونه‌برداری:**

أ.

ب.

ت.

۳. **افزایش نرخ نمونه‌برداری:**

أ.

ب.

آزمایش ۲-۲: عبور سیگنال گسسته از یک سیستم

۱. **حل معادله‌ی تفاضلی:**

أ. دستور `filter` نرم‌افزار MATLAB

filter

1-D digital filter

collapse all in page

Syntax

```
y = filter(b,a,x)
y = filter(b,a,x,zi)
y = filter(b,a,x,zi,dim)
[y,zf] = filter( __ )
```

Description

example

`y = filter(b,a,x)` filters the input data `x` using a rational transfer function defined by the numerator and denominator coefficients `b` and `a`.

If `a(1)` is not equal to `1`, then `filter` normalizes the filter coefficients by `a(1)`. Therefore, `a(1)` must be nonzero.

- If `x` is a vector, then `filter` returns the filtered data as a vector of the same size as `x`.
- If `x` is a matrix, then `filter` acts along the first dimension and returns the filtered data for each column.
- If `x` is a multidimensional array, then `filter` acts along the first array dimension whose size does not equal 1.

More About

collapse all

Rational Transfer Function

The input-output description of the `filter` operation on a vector in the Z-transform domain is a rational transfer function. A rational transfer function is of the form

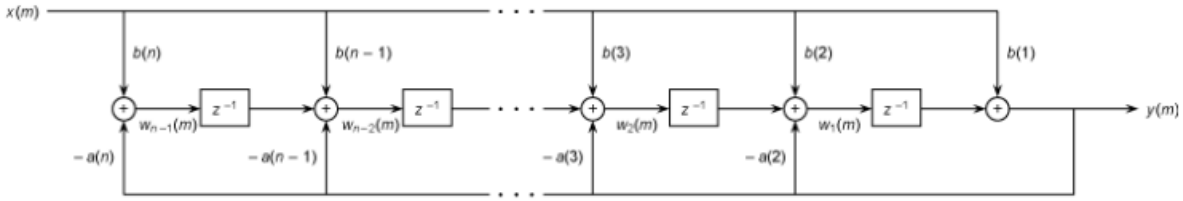
$$Y(z) = \frac{b(1) + b(2)z^{-1} + \ldots + b(n_b+1)z^{-n_b}}{1 + a(2)z^{-1} + \ldots + a(n_a+1)z^{-n_a}} X(z),$$

which handles both FIR and IIR filters [1]. n_a is the feedback filter order, and n_b is the feedforward filter order. Due to normalization, assume $a(1) = 1$.

You also can express the rational transfer function as the difference equation

$$\begin{aligned} a(1)y(n) = {} & b(1)x(n) + b(2)x(n-1) + \ldots + b(n_b+1)x(n-n_b) \\ & -a(2)y(n-1) - \ldots - a(n_a+1)y(n-n_a). \end{aligned}$$

Furthermore, you can represent the rational transfer function using its direct-form II transposed implementation, as in the following diagram. Here, $n_a = n_b$.



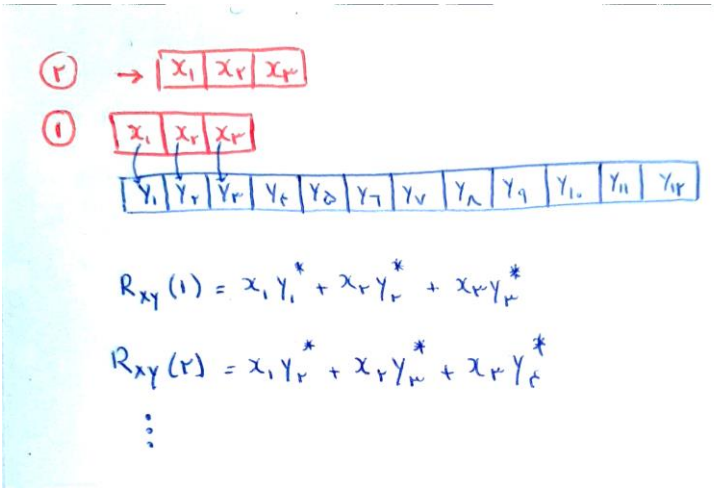
The operation of `filter` at sample m is given by the time-domain difference equations

$$\begin{aligned} y(m) &= b(1)x(m) + w_1(m-1) \\ w_1(m) &= b(2)x(m) + w_2(m-1) - a(2)y(m) \\ \vdots \quad &= \quad \vdots \qquad \vdots \\ w_{n-2}(m) &= b(n-1)x(m) + w_{n-1}(m-1) - a(n-1)y(m) \\ w_{n-1}(m) &= b(n)x(m) - a(n)y(m). \end{aligned}$$

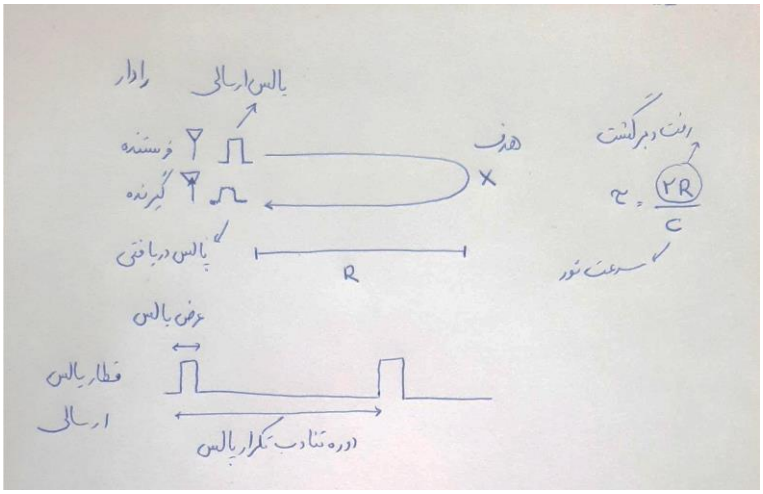
ب.

۲. **تابع همبستگی و کاربرد آن:**

أ.



ب. مختصری از رادار:



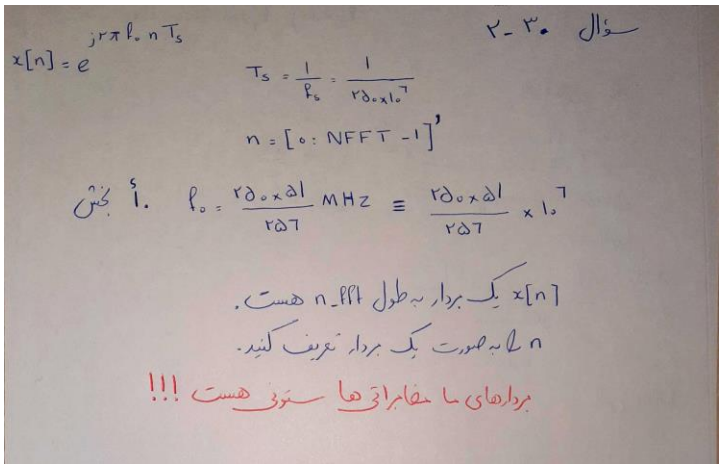
فایل‌های داده شده نمونه‌های زمانی پالس ارسالی و نمونه‌های دریافتی هستند و همگی مختلط می‌باشند. در واقع این سیگنال‌ها معادل پایین‌گذر سیگنال واقعی است که ارسال یا دریافتی می‌شود. با معادل‌پایین‌گذر در آزمایش بعد بیشتر آشنا می‌شوید.

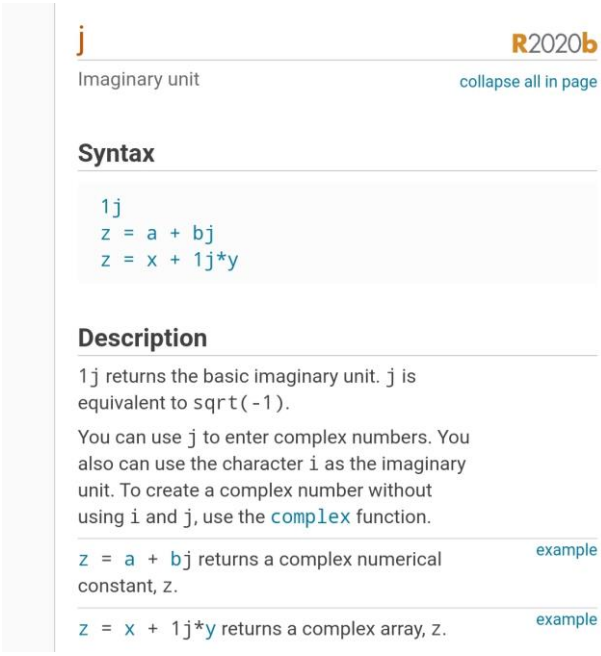
نیازی نیست برای به دست آوردن فاصله و دوره‌تناوب تکرار پالس از توابع MATLAB برای محاسبه‌ی پیک استفاده‌کنید. با بزرگ‌نمایی نمودار همبستگی و قرایت نقطه‌ی قله می‌توان فاصله و دوره‌تناوب تکرار پالس را به دست آورد.

آزمایش ۳-۲: تحلیل حوزه‌ی فرکانس

۱. **محاسبه‌ی طیف سیگنال:**

أ.





j R2020b

Imaginary unit collapse all in page

Syntax

```
1j
z = a + bj
z = x + 1j*y
```

Description

1j returns the basic imaginary unit. j is equivalent to sqrt(-1).

You can use j to enter complex numbers. You also can use the character i as the imaginary unit. To create a complex number without using i and j, use the complex function.

z = a + bj returns a complex numerical constant, z. example

z = x + 1j*y returns a complex array, z. example

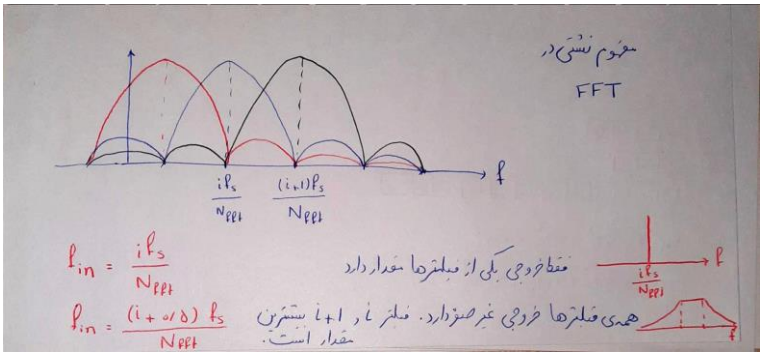
ب.

ت.

ث.

ج.

ح.



نشتی خانه‌های FFT غیرمجاور را می‌توان با ضرب نمونه‌های زمانی در یک پنجره مانند blackman-hariss کم نمود.

خ.

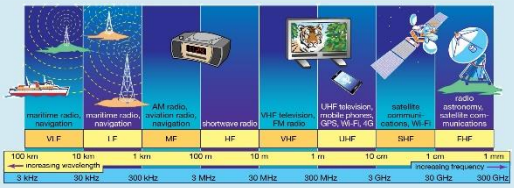
د.

ذ.

ر.

بسمه تعالی

آزمایش سوم



رادیونرم‌افزار و پایش طیف سیگنال‌های رادیویی

شرح آزمایش

آزمایش ۱-۳: معادل‌پایین‌گذر و میان‌گذر سیگنال

۱. **تولید فرآیند میان‌گذر و پایین‌گذر:** در صورتی که یک فرآیند تمام‌گذر از یک فیلتر پایین‌گذر عبور کند، یک فرآیند پایین‌گذر حاصل می‌شود. در این‌جا می‌خواهیم یک فرآیند پایین‌گذر را به معادل میان‌گذر تبدیل نماییم و مجدداً آن را به معادل پایین‌گذر تبدیل کنیم. کلیه‌ی طیف‌ها می‌بایست بر حسب dBm باشد و برچسب‌گذاری محور فرکانس به درستی انجام و بر حسب MHz بیان شود.

أ. **تولید فرآیند پایین‌گذر:** دنباله‌ی تصادفی مختلط با ۴۰۹۶ نمونه‌ی مستقل و با توزیع گاوسی مختلط با میانگین صفر و واریانس ۱ تولید نمایید. با استفاده از ابزار طراحی فیلتر نرم‌افزار MATLAB (دستور `filterDesigner` را در پنجره‌ی دستور وارد نمایید و ضرایب فیلتر را به workspace ارسال نمایید) یک فیلتر پایین‌گذر FIR از نوع Equiripple و با پارامترهای فیلتری $f_s = 28.8\text{MHz}$، $f_{\text{pass}} = 1.4\text{MHz}$، $f_{\text{stop}} = 2\text{MHz}$، $A_{\text{pass}} = 0.5\text{dB}$ و $A_{\text{stop}} = 70\text{dB}$ طراحی نمایید. پس از به دست آوردن ضرایب فیلتر با استفاده از عمل کانولوشن خروجی این فیلتر را برای ورودی تولید شده، به دست آورید. طیف ۴۰۹۶ نقطه‌ای فرآیند تمام‌گذر ورودی و فرآیند پایین‌گذر خروجی را به دست آوردید و در یک شکل با دو زیرنمودار رسم نمایید. (**راهنمایی:** خروجی عمل کانولوشن تعداد نقاط بیشتری دارد و می‌بایست این تعداد نقاط را برابر با ۴۰۹۶ نقطه نمود.)

ب. **تولید فرآیند میان‌گذر معادل یک فرآیند پایین‌گذر:** حال با استفاده از مباحثی که در بخش تئوری مطرح شد، فرآیند پایین‌گذر تولید شده را به یک فرآیند میان‌گذر بر روی فرکانس 3.57MHz منتقل نمایید. حال طیف ۴۰۹۶ نقطه‌ای هر سه فرآیند را در یک شکل با سه زیرنمودار رسم نمایید.

ت. **تولید فرآیند پایین‌گذر از یک سیگنال میان‌گذر:** فرآیند تولید شده در بخش ب را دوباره به معادل پایین‌گذر خود تبدیل نموده و خروجی را از سه فیلتر پایین‌گذر مختلف با پهنای‌باند 0.7MHz، 1.4MHz و 2.8MHz عبور دهید. طیف فرآیند خروجی این سه فیلتر را با طیف بخش أ مقایسه نمایید و در هر مورد میانگین مربع خطای خروجی زمانی هر فیلتر را با طیف بخش أ به دست آوردید. فیلترها را با استفاده از دستور `fir1` تولید نموده و مرتبه‌ی فیلتر را برابر با ۱۲۰ در نظر بگیرید. (فرکانس نمونه‌برداری همان $f_s = 28.8\text{MHz}$ می‌باشد.)

ث. ★ **کار با صوت:** طیف سیگنال `test` که در اختیار شما قرار گرفته را رسم نمایید و با انتخاب فرکانس مرکزی برابر با 40kHz آن را به یک سیگنال باند پایه تبدیل نمایید. حال آن را از یک فیلتر پایین‌گذر 20kHz، 5kHz و 500Hz عبور دهید. خروجی هر یک از گوش دهید. موارد مشاهده شده را توضیح دهید. فرکانس نمونه‌برداری صوت برابر با 160kHz است.

آزمایش ۲-۳: کار با رادیونرم‌افزار ADALM-PLUTO

۱. **اطمینان از نرم‌افزار بسته‌ی پشتیبانی سخت‌افزاری ADALM-PLUTO:** درون نرم‌افزار MATLAB دستور **`sdrdev('Pluto')`** را وارد نمایید و اطمینان حاصل نمایید که این دستور برای نرم‌افزار MATLAB شناخته شده باشد و هیچ پیغام خطایی ظاهر نمی‌شود.

۲. **اطمینان از سخت‌افزار بسته‌ی پشتیبانی سخت‌افزاری ADALM-PLUTO:**

أ. **اتصال ADALM-PLUTO:** رادیو نرم‌افزار ADALM-PLUTO خود را درون یکی از درگاه‌های USB2.0 یا USB3.0 وارد نمایید. در این مرحله اتصال آنتن لازم نیست ولی اتصال آن نیز بلامانع است.

ب. **بررسی شناخته شدن سخت‌افزار ADALM-PLUTO:** عبارت **`my_pluto = findPlutoRadio`** را درون پنجره‌ی دستور MATLAB وارد نماید. اگر سخت‌افزار شناخته شود عبارت زیر ظاهر می‌شود.

```
>> my_pluto = sdrinfo
my_pluto =
      struct with fields:
            RadioID: 'usb:0'
          SerialNum: '1000002355237307002300310902l6eaeb'
```

اگر سخت‌افزار قابل شناسایی نباشد و یا به رایانه متصل نباشد یک خروجی ساختار تهی دریافت خواهید نمود. می‌توانید سخت‌افزار خود را قطع نمایید تا خروجی زیر را دریافت کنید.

```
>> my_pluto = sdrinfo
my_pluto
0×1 empty struct array with fields:

    RadioID
    SerialNum
```

۳. **پارامترها و راه‌اندازی اولیه‌ی رادیونرم‌افزار ADALM-PLUTO:** به منظور راه‌اندازی رادیونرم‌افزار ADALM-PLUTO نیاز به آشنایی با پارامترها و شیوه‌ی تعریف شیءهای رادیونرم‌افزار است. در ادامه کلیت کدی برای راه‌اندازی اولیه‌ی این رادیونرم‌افزار آمده است. این کد را وارد نمایید و از اجرای آن و آشنایی با کلیه‌ی پارامترها مطمئن شوید.

```
%% Parameters
fs = 10e6;                % Baseband Sampling Rate (65105 to 61.44e6 Hz)
frame_size = 4096;        % Samples per Each Frame (< 2^20)
% Transmitter Parameters
tx_fc = 325e6;            % Set Transmitter Center Frequency
                          % (AD9363: 325-3800MHz) (AD9364: 70-6000MHz)
tx_gain = -30;            % Set Transmitter Attenuation as a Negative Gain
                          % (-89.75 to 0 dB)
tx_address = 'usb:0';     % Set Transmitter Identification Number
% Receiver Parameters
rx_fc = 325e6;            % Set Receiver Center Frequency
                          % (AD9363: 325-3800MHz) (AD9364: 70-6000MHz)
rx_gain = 20;             % Set Receiver Gain (-4dB to 71dB)
rx_address = 'usb:0';     % Set Receiver Identification Number
% Initialize ADALM-PLUTO
dev = sdrdev('Pluto');              % Create Radio Object for ADALM-PLUTO
setupSession(dev)
configurePlutoRadio('AD9363');      % Configure ADALM-PLUTO Radio Firmware
% Define Transmitter Object
tx = sdrtx('Pluto','RadioID',tx_address); % CreateTransmitterSystem Object
tx.CenterFrequency = tx_fc;         % Set Transmitter Center Frequency
tx.Gain = tx_gain;                  % Set Transmitter Gain
tx.BasebandSampleRate = fs;         % Set Baseband Sampling Rate
% Define Receiver Object
rx = sdrrx('Pluto','RadioID',rx_address); % Create Receiver System Object
rx.CenterFrequency = rx_fc;         % Set Receiver Center Frequency
rx.BasebandSampleRate = fs;         % Set Baseband Sampling Rate
rx.SamplesPerFrame = frame_size;    % Samples per Each Frame (< 2^20)
rx.GainSource = 'Manual';           % AGC Settings
```

```
rx.Gain = rx_gain;                  % Receiver Gain
rx.OutputDataType = 'double';       % Output Data Type
```

۴. **ارسال متوالی سیگنال با فرستنده‌ی ADALM-PLUTO:** زمانی که رادیونرم‌افزار ADALM-PLUTO روشن شد، حتی اگر درخواستی ارسال نشده باشد، فرستنده شروع به ارسال داده می‌کند و آخرین محتوای بافر خود را ارسال می‌نماید. این وضعیت تا زمانی که رادیونرم‌افزار روشن است، ادامه دارد. حال اگر بنا باشد یک داده به صورت متوالی ارسال شود، از دستور **transmitRepeat** استفاده می‌شود. در این‌جا می‌خواهیم یک سیگنال سینوسی به فاصله‌ی 1MHz نسبت به فرکانس مرکزی 325MHz تولید نماییم.

گام ۱. مانند کد زیر یک سیگنال سینوسی ارسال نمایید.

```
% Transmit Repeat
f_offset = 1e6;
t = (0:2^14-1)'/fs;
sin_wave = exp(1j*2*pi*f_offset*t);
tx.transmitRepeat(sine_wave);
```

۵. **دریافت سیگنال با استفاده از گیرنده‌ی ADALM-PLUTO:** در این‌جا بناست پهنای‌باندی برابر 10MHz حول فرکانسی مرکزی 325MHz را دریافت نموده و طیف توان و نمونه‌های زمانی معادل باندپایه‌ی سیگنال نمایش داده شود. برای این بخش خروجی فرستنده را با کابل به ورودی گیرنده وصل نمایید.

گام ۱. اطمینان حاصل شود که پارامترهای رادیونرم‌افزار به نحوی تنظیم شده باشد که فرکانس مرکزی آن 325MHz، بهره‌ی گیرنده برابر 20dB، نرخ نمونه‌برداری برابر 10MHz، طول داده‌های هر فریم برابر ۴۰۹۶ نقطه، نوع داده‌های خروجی به صورت double و زمان توقف دریافت داده برابر ۶۰ ثانیه باشد.

گام ۲. حلقه‌ی تکرار برنامه‌ی خود را با استفاده از دستورهای **tic** و **toc** و بر اساس متغیر زمان توقف دریافت داده (**stop_time**) پیاده‌سازی نمایید.

گام ۳. با استفاده از فراخوانی شیء گیرنده با وارد کردن **rx()** یک فریم از ADALM-PLUTO را دریافت نمایید و در متغیر **pluto_data** ذخیره نمایید.

گام ۴. FFT داده‌های به دست آمده از رادیونرم‌افزار را محاسبه و طیف توان آن را بر حسب dBm به دست آورید و در متغیر **pluto_data_fft** قرار دهید.

گام ۵. بر اساس متغیر **pluto_data** و **pluto_data_fft** دو نمودار زیر هم رسم نمایید. نمودار بالا مقدار حقیقی یا موهومی داده‌های زمانی و نمودار پایین طیف توان لحظه‌ای و بیشینه‌ی طیف توان لحظه‌ای در کل بازه‌ی زمان دریافت (Max Hold) را نمایش می‌دهد که با دریافت داده‌ی جدید به روز می‌شود. (**راهنمایی:** برای این کار از دستورهای **refreshdata** و **drawnow** استفاده نمایید. بخش Automatically Refresh Plot After Changing Data در راهنمای MATLAB را مطالعه نمایید.)

آزمایش ۳-۳: پایش طیف فرکانسی با استفاده از رادیو نرم‌افزار

۱. **جاروب فرکانسی رادیو نرم افزار ADALM-PLUTO:** برنامه ای بنویسید که کل گستره‌ی فرکانسی 75MHz تا 6GHz را با رادیونرم‌افزار ADALM-PLUTO جاروب نماید و طیف توان کل این بازه را در قالب یک نمودار ارایه دهد.

گام ۱. پارامترهای رادیونرم‌افزار باید در یک حلقه به نحوی تنظیم شود که از فرکانس مرکزی 100MHz تا 6GHz با گام‌های 50MHz جابه‌جا شود. بهره‌ی گیرنده برابر 30dB، نرخ نمونه‌برداری برابر 50MHz، طول داده‌های هر فریم برابر ۴۰۹۶ نقطه، نوع داده‌های خروجی به صورت double باشد.

گام ۲. در هر گام ۵ فریم را قرایت کرده و نادیده بگیرید. قدرمطلق FFT مربوط به ۱۰ فریم بعدی را محاسبه کرده و میانگین این ۱۰ فریم را محاسبه کنید. تعداد نمونه‌های طیف میانگین را با ضریب ۱۶ کاهش دهید. (**راهنمایی:** قبل از FFT می‌بایست مقدار DC هر فریم دریافتی را حذف نمود. برای کاهش تعداد نمونه از دستور **decimate** استفاده نمایید.)

گام ۳. طیف میانگین با نرخ کاهش‌یافته را به یک متغیر اضافه نمایید. طیف حاصل را به dBm تبدیل کرده و نمایش دهید.

۲. **جستجوی ایستگاه‌های رادیو FM با رادیونرم‌افزار ADALM-PLUTO:** سیگنال‌های رادیو FM در محدوده‌ی فرکانسی 88MHz تا 108MHz قرار دارند. سیگنال‌های حامل ایستگاه‌ها به فاصله‌ی 200kHz از یک‌دیگر قرار گرفته‌اند. با استفاده از برنامه‌های قبل انجام دهید.

گام ۱. به کمک ADALM-PLUTO بیان کنید در محدوده‌ی فرکانسی ذکر شده چه تعداد ایستگاه رادیو FM وجود دارد؟

گام ۲. فرکانس ایستگاه‌های قوی رادیو FM را ثبت نمایید.

گام ۳. قوی‌ترین سیگنال رادیو FM را پیدا نمایید و تنها این سیگنال را با استفاده از رادیونرم‌افزار دریافت نمایید و رفتار زمانی بخش حقیقی و موهومی معادل باندپایه‌ی آن را تحلیل نمایید.

گام ۴. با استفاده از کدی که در اختیار شما قرار می‌گیرد، به این ایستگاه رادیویی گوش دهید.

۳. **پایش طیف تلفن همراه - نسل ۲ با ADALM-PLUTO:** در استاندارد GSM-900 مربوط به نسل دو پیاده‌سازی شده در ایران گستره‌ی فرکانسی 890MHz تا 915MHz مربوط به ارسال به ایستگاه پایه[1] و گستره‌ی فرکانسی 935MHz تا 960MHz مربوط به دریافت از ایستگاه پایه است. در استاندارد GSM-1800 گستره‌ی فرکانسی 1710MHz تا 1785MHz برای ارسال به ایستگاه پایه و گستره‌ی فرکانسی 1805MHz تا 1880MHz برای دریافت از ایستگاه پایه در نظر گرفته شده است. در این محدوده‌های فرکانسی به دنبال سیگنال تلفن همراه نسل دو بگردید.

گام ۱. کانال‌های موبایل نسل دوم دارای چه پهنای کانالی هستند و در چه فاصله‌ای از یکدیگر قرار گرفته‌اند؟

گام ۲. تلفن همراه خود را برای ارتباط با نسل دو تنظیم نمایید و با گرفتن تماس تلفنی و ارسال پیام سعی کنید سیگنال تلفن همراه خود را مشاهده نمایید.

گام ۳. رفتار طیف نسل دوم را با توجه به نمودار آبشاری تحلیل نمایید. این طیف با کدام طیف دسترسی چندگانه تطبیق دارد.

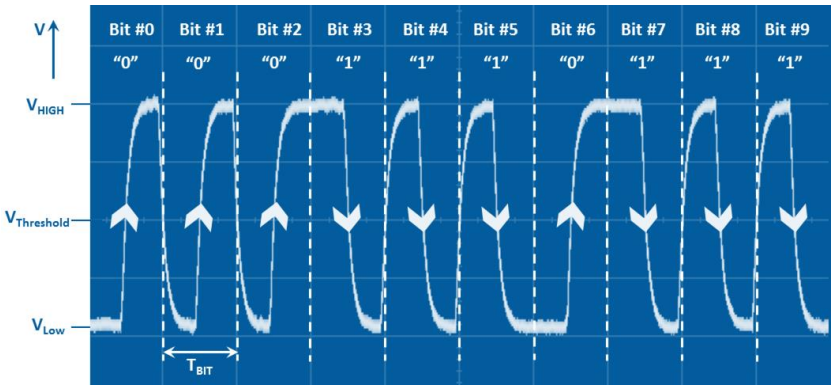
۴. **پایش طیف ریموت خودرو:** رادیو نرم‌افزار ADALM-PLUTO را در فرکانس 433.9MHz تنظیم نمایید و ریموت خودرو را فشار دهید.

گام ۱. فرکانس نمونه‌برداری را برابر با 250kHz قرار داده و طول بسته‌ها را برابر 65536 در نظر بگیرید. رفتار طیف را تحلیل نمایید.

گام ۲. با تنظیم دستی فرکانس مرکزی ADALM-PLUTO سعی کنید سیگنال را به باندپایه انتقال دهید و نمونه‌های زمانی آن را رسم کنید.

گام ۳. با انتخاب یک سطح آستانه، اگر سیگنال از این آستانه بالاتر بود نمایش طیف و نمونه‌های زمانی متوقف نمایید و از نمونه‌های آن برای گام آینده استفاده کنید.

گام ۴. با استفاده از کدگزاری منچستر که در زیر آمده است، کد ریموت خودرو را استخراج نمایید.



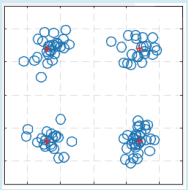
[1] base station

مراجع

[1] R. W. Stewart, K. Barlee, L. Crockett, and D. Atkinson, *Software Defined Radio using MATLAB & Simulink and the RTL-SDR*. 2015.

[2] T. F. Collins, R. Getz, D. Pu, and A. M. Wyglinski, *Software-defined radio for engineers*. Norwood, MA: Artech House, 2018.

بسمه تعالی



آزمایش چهارم

معرفی مدولاسیون دیجیتال خطی

شرح آزمایش

آزمایش ۱-۴: پیاده‌سازی فرستنده مدولاسیون PAM

در این آزمایش بنا داریم تعدادی بسته‌ی داده را برای ارسال از طریق مدولاسیون 4PAM آماده نماییم. از آن‌جا که در آزمایش‌های آینده، بناست این فرستنده با استفاده از رادیو نرم‌افزار ADALM-PLUTO پیاده‌سازی شود، بنابراین می‌بایست با استفاده از پارامترها و ملاحظات عملی، این کار انجام شود.

پارامترهایی که در این پیاده‌سازی اهمیت دارد شامل نرخ ارسال بیت (**bit_rate**)، ضریب افزایش نرخ نمونه‌برداری سمبل‌ها یا تعداد نمونه‌های هر سمبل (**smpl_per_symbl**)، تعداد سمبل‌های ارسالی در هر بسته‌ی داده (**pkt_size**) و مدت زمان ارسال داده (**stop_time**) می‌باشد. در این آزمایش مقدار پارامترهای مورد نیاز به صورت جدول 1 می‌باشد.

جدول 1 پارامترهای آزمایش ۱-۴

پارامتر	smpl_per_symbl	pkt_size
مقدار	۸	۱۰

۱. **تولید بیت:** با استفاده از تابع **bit_gen** تعدادی بیت ۰ و ۱ را متناسب با طول بسته‌ی داده و با در نظر گرفتن پارامترهای مدولاسیون 4PAM تولید نموده و آن را درون ماتریس **b_tx** ذخیره نمایید. ماتریس **b_tx** یک ماتریس با اندازه‌ی pkt_size×k (۱۰×۲) می‌باشد.

۲. **نگاشت بیت به سمبل:**

گام ۱. **تولید دنباله‌ی کد گری به صورت باینری:** تابعی بنویسید که آرگومان ورودی آن تعداد بیت‌های دنباله باشد و خروجی آن کد گری مربوط به این دنباله باشد. سطر اول این تابع می‌بایست به صورت زیر باشد.

```
function [b_gray] = gray_code(k)
```

گام ۲. **کدگذاری گری بیت‌های تولیدی:** با استفاده از تابع **gray_code**، ماتریس مربوط به کدگذاری گری باینری مدولاسیون 4PAM را تولید کنید و آن را در ماتریس **b_gray** ذخیره کنید. بردار جدیدی متناظر با ماتریس **b_tx** و با نام **sym_idx** تولید نمایید و نگاشتی یک‌به‌یک بین سطرهای ماتریس **b_tx** و ماتریس **b_gray** برقرار کنید. به عبارتی سطر iام بردار **sym_idx**، شماره‌ی سطری از **b_gray** را نشان می‌دهد که برابر با سطر iام ماتریس **b_tx** باشد.

گام ۳. **تولید سمبل‌های ارسالی:** با استفاده از تابع **constellation**، تمامی سمبل‌های ارسالی یک مدولاسیون 4PAM را تولید کرده و سمبل‌های متناظر با هر سطر بردار **sym_idx** را تولید کنید. این سمبل‌ها را در بردار **mod_sym** ذخیره نمایید.

۳. **شکل‌دهی پالس ارسالی:** همان‌طور که در بخش تئوری بیان گردید، سیگنال ارسالی را می‌توان با دو روش ضرب کرونیکر و کانولوشن تولید و پالس ارسالی را شکل‌دهی نمود. تابعی بنویسید که بردار شماره‌ی سمبل‌های تولیدشده (**sym_idx**)، مدولاسیون (**modulation**)، مرتبه‌ی مدولاسیون (**M**)، نرخ نمونه‌برداری (**fs**)، تعداد نمونه‌های هر سمبل (**smpl_per_symbl**)، نام تابع شکل‌دهنده‌ی پالس (**pulse_name**) و روش تولید شکل‌دهی پالس (**mode**) را دریافت کند و نمونه‌های سیگنال ارسالی متناظر با یک بسته و منظومه‌ی سیگنالی را تولید کند. سطر اول این تابع می‌بایست به صورت زیر باشد.

```
function [tx_smpl, cons] = pulse_modulation(sym_idx, modulation, M, fs,
smpl_per_symbl, pulse_name , pulse_shape_mode, varargin)
```

توجه داشته باشید که گام ۳ سوال قبل را مجدد در قالب یک تابع منسجم انجام داده‌ایم. در این‌جا **pulse_shape_mode** روش شکل‌دهی پالس می‌باشد که یکی از دو حالت **'kron'** و **'conv'** را اختیار می‌کند.

با استفاده از این تابع نمونه‌های ارسالی مربوط به یک بسته‌ی داده را آماده‌ی ارسال کرده و حاصل را در بردار **tx_out** ذخیره نمایید. تابع شکل‌دهی پالس در این آزمایش پالس مثلثی است. اثر شکل پالس‌ها در آزمایش بعد مورد بررسی قرار می‌گیرد.

(**راهنمایی ۱:** ضرب کرونیکر در برنامه‌ی MATLAB با استفاده از دستور **kron** انجام می‌شود که شیوه‌ی استفاده از آن به صورت **tx_out = kron(s, p)** می‌باشد.)

(**راهنمایی ۲:** برای انجام عمل Zero Padding از تابع **upsmpl** خود استفاده کنید. دقت نمایید که تابع شما نباید بعد از آخرین سمبل صفر اضافه نماید. حال سیگنال ارسالی را توسط عملگر کانولوشن و به صورت **tx_out = conv(s_zero_pad, p)** تولید نمایید.)

آزمایش ۲-۴: پیاده‌سازی گیرنده‌ی مدولاسیون PAM

در ادامه‌ی می‌خواهیم گیرنده‌ی یک مدولاسیون PAM را پیاده‌سازی نماییم.

۱. **تشخیص قدرت سیگنال دریافتی در راستای سیگنال‌های پایه:** تابعی بنویسید که دو بردار **rx_smpl** و **p** را دریافت کند و قدرت سیگنال را در راستای **p** محاسبه کند. این تابع باید بتواند قدرت سیگنال را در راستای **p**، با استفاده از هر دو روش correlator و فیلتر منطبق محاسبه کند. سطر اول این تابع می‌بایست به صورت زیر باشد.

```
function [rx_sym] = corr_match(rx_smpl, p, smpl_per_symbl, rx_mode)
```

در این تابع **rx_mode** می‌تواند یکی از دو حالت **'correlator'** و یا **'matched_filter'** را اختیار کند.

۲. **گیرنده‌ی حداقل فاصله:** تابعی بنویسید که منظومه‌ی سیگنالی (**constellation**) و بردار سمبل‌های دریافتی (**rx_sym**) را به عنوان ورودی بگیرد و در خروجی، برداری به طول r تولید کند. المان iام بردار سمبل‌های آشکارشده‌ی **det_sym**، نزدیکترین مقدار **constellation** به المان iام بردار **rx_sym** خواهد بود. سطر اول این تابع به صورت زیر می‌باشد.

```
function [det_sym] = min_dist_detector(rx_sym, constellation);
```

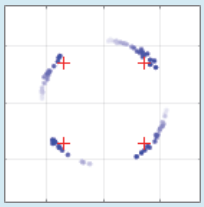
۳. **دمدولاسیون یا آشکارسازی پالس:** نمونه‌های ارسال شده (**tx_smpl**) را به عنوان نمونه‌های دریافتی درون بردار **rx_smpl** قرار دهید. در این‌جا می‌خواهیم تابعی بنویسیم که دو عمل قبل را به صورت منسجم در قالب یک تابع انجام دهد. ورودی‌های این تابع نمونه‌های سیگنال دریافتی (**rx_signal**)، نام مدولاسیون (**modulation**)، مرتبه‌ی مدولاسیون (**M**)، نرخ نمونه‌برداری (**fs**)، تعداد نمونه‌های هر سمبل (**smpl_per_symbl**)، نام تابع شکل‌دهنده‌ی پالس (**pulse_name**) و روش تشخیص قدرت سیگنال دریافتی در راستای سیگنال‌های پایه (**mode**) را دریافت کند و خروجی آن سمبل‌های دریافتی (**rx_sym**) و اندیس سمبل‌های آشکارشده (**det_sym_idx**) باشد.

```
function [det_sym_idx, rx_sym_tot] = pulse_demodulation(rx_smpl, modulation, M,
fs, smpl_per_symbl, pulse_name , rx_mode, varargin)
```

مراجع

[1] J. G. Proakis and M. Salehi, *Digital Communications*. Boston: McGraw-Hill, 2008.
[2] M. Rice, *Digital Communications: A Discrete-Time Approach*. Upper Saddle River, NJ: Pearson Education, 2009.
[3] R. W. Stewart, K. Barlee, L. Crockett, and D. Atkinson, *Software Defined Radio using MATLAB & Simulink and the RTL-SDR*. 2015.

بسمه تعالی



آزمایش پنجم

کاوش در مدولاسیون دیجیتال خطی

شرح آزمایش

آزمایش ۱-۵: مقداردهی‌های اولیه

در این آزمایش بنا داریم تعدادی بسته‌ی داده را برای ارسال از طریق مدولاسیون‌های خطی آماده نماییم. از آن‌جا که در این آزمایش بناست این فرستنده با استفاده از رادیو نرم‌افزار ADALM-PLUTO پیاده‌سازی شود، می‌بایست با استفاده از پارامترها و ملاحظات عملی، این کار انجام شود.

پارامترهایی که در این پیاده‌سازی اهمیت دارد شامل نرخ نمونه‌برداری (**fs**)، ضریب افزایش نرخ نمونه‌برداری سمبل‌ها یا تعداد نمونه‌های هر سمبل (**smpl_per_symbl**)، تعداد سمبل‌های ارسالی در هر بسته‌ی داده (**pkt_size**) و مدت زمان ارسال داده (**stop_time**) می‌باشد. در این آزمایش مقدار پارامترهای مورد نیاز به صورت جدول 1 می‌باشد.

جدول 2 پارامترهای آزمایش ۵

پارامتر	fs	smpl_per_symbl	pkt_size	stop_time
مقدار	10MHz	۸	شبیه‌سازی: ۱۰۰۰۰۰۰ سخت‌افزار: ۱۰۰۰	۱۰۰ ثانیه

دقت نظر داشته باشید که در هر آزمایش تنها می‌خواهیم برنامه‌هایی که در آزمایش قبل نوشته شده است را تکمیل نماییم. هم‌چنین ابتدای برنامه‌نویسی **pkt_size** را مقدار کوچکی مانند ۱۰ قرار دهید و پس از اطمینان از برنامه مقدار آن را افزایش دهید.

۱. **تولید M-File مربوط به مقدار دهی اولیه:** در آزمایش‌ها می‌خواهیم، مدولاسیون‌های مختلف، عوامل غیرایده‌آل کانال، استفاده یا عدم استفاده از سخت‌افزار، شکل پالس‌های مختلف و ... را مورد بررسی قرار دهیم. از این رو بهتر است برنامه به صورت یک کل نوشته شود تا با تغییر چند پارامتر در فایل مقداردهی اولیه، تغییرات مد نظر در کل برنامه اعمال شود.

در ادامه قالب کلی این پارامترها آورده شده است. فایل **dcl_init.m** را ایجاد نموده و این موارد را در آن وارد نمایید. سپس می‌بایست این فایل را از برنامه‌ی اصلی خود فراخوانی نمایید.

```
%% Simulation Parameters
flg_hrdwr_usg = 0;
stop_time = 100;

%% Receiver Parameters
fs = 10e6;                          % Baseband Sampling Rate (65105 to 61.44e6 Hz)
ts = 1/fs;                          % Baseband Sampling Time
pkt_size = 1e6;                     % Number of Symbol in Each Packet
rx_alg = 0;                         % Receiver Detection Algorithm
cmpnst_mode = 0;                    % Compensate Mode (0: No Compensation, 1:
Amplitude Compensation, 2: Phase Compensation, 3: Compensation)
%% Modulation Parameters
modulation = 'psk';                 % Modulation Name ('psk', 'pam', 'qam', 'fsk')
```

```
k = 2;                              % Bit Per Symbol
M = 2^k;                            % Modulation Order

smpl_per_symbl = 8;                 % Sample Per Symbol
Ts = smpl_per_symbl*ts;             % Symbol Time

flg_gray_encode = 1;                % Gray Code Usage Flag
mod_det_opt = 'coherent';           % Modulation Detection Option ('coherent',
'noncoherent')

% Pulse Shape Parameters
pulse_name = 'triangular';          % Name of Pulse Shaping Function
beta = 0.99;                        % Parameter for RC, RRC and Gaussian Pulse Shape
span_in_symbl = 0;                  % Trunctation Length for RC, RRC and Gaussian
Pulse Shape (Multiple of Symbol Time)

% Header Option
flg_add_hdr = 0;                    % Flag For Having Packets with Header

% SNR Bound for BER Plots
snr_min = 0;                        % Minimum SNR (dB)
snr_max = 10;                       % Maximum SNR (dB)
snr_step = 1;                       % SNR Step (dB)
snr_db = snr_min:snr_step:snr_max;  % SNR Vector (dB)

%% Channel Parameters
chnl_delay_in_smpl = 0;             % Channel Delay in Sample
chnl_phase_offset = 0 * pi/180;     % Channel Phase Offset
chnl_freq_offset = 0;               % Channel Frequency Offset

%% Hardware Parameters
% Transmitter Parameters
tx_fc = 2400e6;                     % Set Transmiter Center Frequency (AD9363: 325-
3800MHz) (AD9364: 70-6000MHz)
tx_gain = 0;                        % Set Transmiter Attenutaion As a Negative Gain
(-89.75 to 0 dB)
tx_address = 'usb:0';               % Set Transmiter Identification Number

% Receiver Parameters
rx_fc = 2400e6;                     % Set Receiver Center Frequency (AD9363: 325-
3800MHz) (AD9364: 70-6000MHz)
rx_gain = 20;                       % Set Receiver Gain (-4dB to 71dB)
rx_address = 'usb:0';               % Set Receiver Identification Number

% Initialize ADALM-PLUTO
if flg_hrdwr_usg
    dev = sdrdev('Pluto');          % Create Radio Object for ADALM-PLUTO
    setupSession(dev)
    configurePlutoRadio('AD9364');  % Configure ADALM-PLUTO Radio Firmware
end
```

آزمایش ۲-۵: پیاده‌سازی فرستنده مدولاسیون

۱. **تولید بیت:** با استفاده از تابع **bit_gen** تعدادی بیت ۰ و ۱ را متناسب با طول بسته‌ی داده و با در نظر گرفتن پارامترهای مدولاسیون 4PSK تولید نموده و آن را درون ماتریس **b_tx** ذخیره نمایید. ماتریس **b_tx** یک ماتریس با اندازه‌ی pkt_size×k می‌باشد.

۲. **نگاشت بیت به سمبل:**

گام ۱. **کدگذاری گری بیت‌های تولیدی:** با استفاده از تابع **gray_code**، ماتریس مربوط به کدگذاری گری مدولاسیون 4PSK را تولید کنید و آن را در ماتریس **b_gray** ذخیره کنید. بردار جدیدی متناظر با ماتریس **b_tx** و با نام **sym_idx** تولید نمایید و نگاشتی یک‌به‌یک بین سطرهای ماتریس **b_tx** و ماتریس **b_gray** برقرار کنید. به عبارتی سطر iام بردار **sym_idx**، شماره‌ی سطری از **b_gray** را نشان می‌دهد که برابر با سطر iام ماتریس **b_tx** باشد. (راهنمایی: بهتر است سطرهای ماتریس b_gray را به یک عدد صحیح تبدیل نمایید.)

گام ۲. **کدگذاری طبیعی بیت‌های تولیدی:** با استفاده از فلگ **flg_gray_encode**، قابلیتی را به برنامه‌ی خود اضافه نمایید که بتوان کدگزاری گری را اعمال ننمود.

گام ۳. **تولید سمبل‌های ارسالی:** با استفاده از تابع **constellation**، تمامی سمبل‌های ارسالی مدولاسیون 4PSK را تولید کرده و سمبل‌های متناظر با هر سطر بردار **sym_idx** را تولید کنید. این سمبل‌ها را در بردار **mod_sym** ذخیره نمایید.

۳. **شکل‌دهی پالس ارسالی:** با استفاده از تابع **pulse_modulation** نمونه‌های ارسالی مربوط به یک بسته‌ی داده را آماده‌ی ارسال کرده و حاصل را در بردار **tx_smpl** ذخیره نمایید. تابع شکل‌دهی پالس را فعلا مثلثی در نظر بگیرید.

آزمایش ۳-۵: مدل‌سازی کانال

۱. **افزودن تأخیر در کانال:** برای شبیه‌سازی تأخیر در کانال به اندازه‌ی پارامتر **chnl_delay_in_smpl** به ابتدای بردار **tx_smpl** صفر اضافه نمایید. حاصل را درون بردار **tx_smpl_delayed** قرار دهید. دقت نمایید که طول بردار افزایش می‌یابد و می‌بایست در ادامه‌ی برنامه اثر این افزایش لحاظ شود. در این‌جا فعلا مقدار تأخیر برابر با صفر قرار دهید.

۲. **اعمال اختلاف فاز در کانال:** برای اعمال اختلاف فاز در کانال می‌بایست بردار **tx_smpl_delayed** را در **exp(1i*chnl_phase_offset)** ضرب شود. حاصل را در بردار **rx_smpl** قرار دهید. در این‌جا نیز آفست فاز را برابر با صفر قرار دهید.

۳. **شبیه‌سازی کانال با نویز سفید گاوسی**

گام ۱. **تعیین واریانس نویز بر اساس نسبت سیگنال به نویز (E_b/N_0):** ابتدا بر اساس خروجی تابع **constellation**، مقدار متوسط انرژی سمبل را به دست آورید (**Es_avg**). حال با استفاده از متوسط انرژی سمبل، مقدار انرژی متوسط بیت را به دست آورده و برابر متغیر **Eb** قرار دهید. مقدار E_b/N_0 را برابر با 10dB در نظر بگیرید (به عبارتی پارامترهای **snr_min** و **snr_max** برابر 10 تنظیم می‌شوند). سپس واریانس نویز را بر اساس نسبت E_b/N_0 به دست آورید و آن را درون متغیر **var_noise** قرار دهید.

گام ۲. **افزودن نویز به سیگنال:** با استفاده از تابع **randn** یک بردار نویز مختلط با واریانس **var_noise** و ابعاد برابر با **tx_smpl** تولید نمایید و آن را **noise_smpl** نامگذاری نمایید. سپس این بردار را با بردار **rx_smpl** جمع نموده و حاصل را **rx_smpl_noise** بنامید.

آزمایش ۴-۵: پیاده‌سازی گیرنده‌ی مدولاسیون

در ادامه‌ی فایل آزمایش قبل می‌خواهیم گیرنده‌ی یک مدولاسیون خطی را پیاده‌سازی نماییم.

۱. **دمدولاسیون یا آشکارسازی پالس:** نمونه‌های دریافت شده به همراه نویز (**rx_smpl_noise**) را که اثر تأخیر کانال در آن جبران شده را درون بردار **rx_smpl** قرار دهید. با استفاده از تابع **pulse_demodulation** با ورودی‌های نمونه‌های سیگنال دریافتی (**rx_smpl**)، نام مدولاسیون (**modulation**)، مرتبه‌ی مدولاسیون (**M**)، نرخ نمونه‌برداری (**fs**)، تعداد نمونه‌های هر سمبل (**smpl_per_symbl**)، نام تابع شکل‌دهنده‌ی پالس (**pulse_name**) و روش تشخیص قدرت سیگنال دریافتی در راستای سیگنال‌های پایه (**mode**)، سمبل‌های دریافتی (**rx_sym**) و اندیس سمبل‌های آشکارشده (**det_sym_idx**) را به دست آورید.

۲. **تبدیل از کدگذاری گری به کدگذاری باینری:** به کمک بردار **det_sym_idx** و ماتریس **b_gray** اندیس سمبل‌های آشکارشده‌ی با کدگذاری باینری را به دست آوردید و درون متغیر **det_bit** قرار دهید. برای این منظور متناظر با محتوای هر سطر **det_sym_idx** سطری متناظر با آن محتوا از **b_gray** انتخاب می‌شود. این عمل را می‌بایست برای حالتی که از کدگزاری باینری نیز استفاده می‌شود و **flg_gray_encode = 0** است نیز مدیریت شود.

۳. **محاسبه‌ی خطای سمبل:** با مقایسه‌ی تعداد اختلاف‌های بردارهای **det_sym_idx** و **sym_idx** و محاسبه نسبت اختلاف این دو بردار به تعداد کل سمبل‌ها، خطای سمبل را به دست آورده و آن را در متغیر **ser** قرار دهید.

۴. **تبدیل سمبل به بیت و محاسبه‌ی خطای بیت:** با مقایسه‌ی تعداد اختلاف ماتریس‌های **det_bit** و **b_tx** و محاسبه نسبت تعداد اختلاف این دو ماتریس به تعداد کل بیت‌ها، خطای بیت را به دست آورده و آن را در متغیر **ber** قرار دهید.

آزمایش ۵-۵: خواسته‌های کلی

۱. **رسم نمودار نرخ خطای بیت:** با تغییر نسبت سیگنال به نویز بین 0 تا 10dB نرخ خطای بیت را برای مدولاسیون‌های 16QAM، 8PSK و 4PAM را به دست آورده و در یک نمودار رسم نمایید و آن را با خروجی تابع **berawgn** نرم‌افزار MATLAB مقایسه نمایید. این کار را برای شکل موج مثلثی و root raised cosine انجام دهید. هم‌چنین اثر اعمال کدگزاری گری و کدگزاری باینری را نیز بر روی نمودار نرخ خطای بیت نشان دهید.

۲. **اثر تأخیر در کانال:**

در این‌جا فرض نمایید مدولاسیون 4PSK است و نسبت سیگنال به نویز (E_b/N_0) برابر با 10dB است.

گام ۱. **اثر تأخیر در کانال بر روی نرخ خطای بیت و منظومه‌ی سیگنالی:** تأخیر کانال را برابر با گرد شده‌ی 0.1، 0.5 و 0.8 برابر طول یک سمبل قرار دهید (یک عدد صحیح معادل تعداد نمونه‌های زمانی تأخیر یافته). نرخ خطای بیت را به دست آورد و با حالت بدون تأخیر مقایسه نمایید. منظومه سیگنالی حالت‌های فوق را نیز با یکدیگر مقایسه کرده و علت مشاهدات خود را توضیح دهید. (**rx_alg = 0** و **flg_hrdwr_usg = 0**)

گام ۲. **اضافه کردن هدر:** برای یافتن نقطه‌ی شروع درست داده‌ها، تعدادی داده‌ی مشخص به عنوان هدر در ابتدای رشته بیت‌های ارسالی قرار دهید. نمونه‌های زمانی مربوط به هدر را از بردار **tx_smpl** جدا کرده و در متغیر **hdr_smpl** ذخیره نمایید. بیت‌های هدر می‌بایست به صورت زیر باشد. این رشته بیت خواص خودهمبستگی خوبی دارد.

```
hder_bit = repmat(de2bi(hex2dec('1C6387FF5DA4FA325C895958DC5'))',1,k);
```

گام ۳. **تخمین زمان شروع بسته و جبران اثر تأخیر کانال:** با استفاده از همبستگی بردار **rx_smpl_noise** و نمونه‌های زمانی هدر **hdr_smpl**، نقطه‌ی شروع بسته‌ی داده را تخمین بزنید. با مقایسه‌ی نرخ خطای بیت حالت بدون تأخیر و حالتی که نقطه‌ی شروع داده‌ها تخمین زده می‌شود، از عملکرد برنامه‌ی خود اطمینان حاصل نمایید. (**rx_alg = 1** و **flg_hrdwr_usg = 0**)

۳. **اعمال فاز در کانال و تخمین فاز:**

در این‌جا فرض نمایید مدولاسیون 4PSK است و نسبت سیگنال به نویز (E_b/N_0) برابر با 10dB است.

گام ۱. **اثر فاز اضافی کانال بر نرخ خطای بیت:** اختلاف فاز کانال را برابر ۳۰ درجه قرار دهید. ابتدا برای مدولاسیون 4PSK نرخ خطای بیت را به دست آورده و با حالت بدون اختلاف فاز مقایسه نمایید. منظومه سیگنالی هر دو حالت را نیز با یکدیگر مقایسه نمایید. علت مشاهدات خود را توضیح دهید.

گام ۲. **تخمین فاز اضافی کانال:** اگر به ابتدای بسته‌ی ارسالی هدر بخش قبل اضافه شود، با استفاده از همبستگی بردار **rx_smpl_noise** و نمونه‌های زمانی هدر **hdr_smpl**، فازی را که کانال به سیگنال اضافه کرده است را به دست آورید. (**راهنمایی:** به فاز در قله‌ی خروجی همبستگی دقت نمایید. قله از روی مقدار مطلق همبستگی به

دست می‌آید ولی مقدار خود همبستگی در این اندیس به دست آمده مورد استفاده قرار می‌گیرد. به عملکرد تابع **max** بر روی داده‌های مختلط توجه نمایید.)

گام ۳. **جبران‌سازی فاز کانال:** تلاش کنید با اعمال ضریب مناسب اثر فاز اضافی کانال را جبران نمایید. (**cmpnst_mode = 2**)

آزمایش ۵-۵: پیاده‌سازی سخت‌افزاری

در این بخش **flg_hrdwr_usg = 1** و **rx_alg = 1** می‌باشد.

۱. **ارسال نمونه‌ها با استفاده از ADALM-PLUTO:** بردار **tx_smpl** مربوط به مدولاسیون 4PSK (هدر اضافه شود) را با استفاده از دستور **transmitRepeat** به مدت زمان **stop_time** ثانیه به صورت پی‌درپی در فضا ارسال نمایید. این دستور داده‌ها را از طریق USB به رادیو نرم‌افزار ارسال می‌نماید و در حافظه‌ی سخت‌افزار ذخیره‌می‌نماید. فرستنده داده‌ها را از حافظه‌ی رادیونرم‌افزار قرایت کرده و مدام ارسال می‌نماید. برای این منظور ابتدا می‌بایست پس از پیکربندی رادیو نرم‌افزار یک شیء فرستنده ایجاد نموده و فرکانس مرکزی آن را بر روی 2400MHz تنظیم کرده و بهره‌ی آن را برابر با 0dB تنظیم نمایید.

۲. **دریافت سیگنال با استفاده از ADALM-PLUTO:** ابتدا شیء مربوط به گیرنده را تعریف نمایید. فرکانس مرکزی آن را بر روی 2400MHz تنظیم کرده و بهره‌ی آن را به صورت Manual و برابر 20dB تنظیم نمایید. به علت اطمینان از دریافت یک بسته‌ی کامل، تعداد نمونه‌های زمانی دریافتی را برابر با دو برابر تعداد نمونه‌های زمانی بسته درنظر بگیرید. به منظور دریافت داده، شیء گیرنده را فراخوانی نمایید و حاصل را درون متغیر **rx_smpl** قرار دهید.

۳. **اثر شکل پالس بر روی پهنای‌باند سیگنال ارسالی:** طیف فرکانسی سیگنال **rx_smpl** دریافتی از ADALM-PLUTO را برای مدولاسیون 4PSK و به ازای شکل پالس‌های مثلثی و root raised cosine رسم نماید. در مورد پهنای‌باند و سطح باندهای جانبی آن توضیحاتی ارایه دهید. برای این منظور از برنامه‌های قبلی استفاده نمایید.

۴. **دمدولاسیون و مشاهده‌ی منظومه‌ی سیگنالی:** مشابه قبل عمل دمدولاسیون را انجام داده و منظومه‌ی سیگنالی مربوط به **rx_sym** را رسم نمایید و مشاهدات خود را یادداشت نمایید. منظومه‌ی سیگنالی باید بتواند به صورت به‌لحظه به‌روز شود.

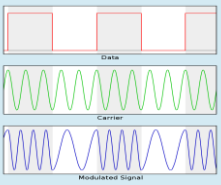
۵. **جبران‌سازی دامنه، فاز و تأخیر و مشاهده‌ی منظومه‌ی سیگنالی:** با یافتن ابتدای بسته‌ها و جبران فاز، دامنه و تأخیر، منظومه‌ی سیگنالی مربوط به **rx_sym** را رسم نمایید. احتمال خطای بیت را نیز در این حالت محاسبه نمایید.

مراجع

[1] J. G. Proakis and M. Salehi, *Digital Communications*. Boston: McGraw-Hill, 2008.
[2] M. Rice, *Digital Communications: A Discrete-Time Approach*. Upper Saddle River, NJ: Pearson Education, 2009.
[3] R. W. Stewart, K. Barlee, L. Crockett, and D. Atkinson, *Software Defined Radio using MATLAB & Simulink and the RTL-SDR*. 2015.

بسمه تعالی

آزمایش ششم



مدولاسیون FSK همدوس

شرح آزمایش

آزمایش ۱-۶: مقداردهی‌های اولیه

در این آزمایش بنا داریم تعدادی بسته‌ی داده را برای ارسال از طریق مدولاسیون FSK آماده نماییم. از آن‌جا که در این آزمایش بناست این فرستنده با استفاده از رادیو نرم‌افزار ADALM-PLUTO پیاده‌سازی شود، می‌بایست با استفاده از پارامترها و ملاحظات عملی، این کار انجام شود.

پارامترهایی که در این پیاده‌سازی اهمیت دارد شامل نرخ نمونه‌برداری (**fs**)، تعداد نمونه‌های هر سمبل (**smpl_per_symbl**)، تعداد سمبل‌های ارسالی در بسته‌ی داده (**pkt_size**) و مدت زمان ارسال داده (**stop_time**) می‌باشد. در این آزمایش مقدار پارامترهای مورد نیاز به صورت جدول 3 می‌باشد.

جدول 3 پارامترهای آزمایش ۶

پارامتر	fs	smpl_per_symbl	pkt_size	stop_time
مقدار	10MHz	64	شبیه‌سازی: ۱۰۰۰۰۰ سخت‌افزار: ۱۰۰۰	۱۰۰ ثانیه

دقت نظر داشته باشید که در این آزمایش نیز تنها می‌خواهیم برنامه‌هایی که در آزمایش قبل نوشته شده است را تکمیل نماییم.

۱. **تولید M-File مربوط به مقدار دهی اولیه:** قالب کلی پارامترهای مورد نیاز در این آزمایش، مانند آزمایش قبل است و فایل **dcl_init.m** می بایست مدولاسیون fsk را نیز بشناسد. این فایل می‌بایست از برنامه‌ی اصلی فراخوانی نمایید.

آزمایش ۲-۶: پیاده‌سازی فرستنده مدولاسیون FSK

۱. **تولید بیت:** با استفاده از تابع **bit_gen** تعدادی بیت ۰ و ۱ را متناسب با طول بسته‌ی داده و با در نظر گرفتن پارامترهای مدولاسیون 4FSK تولید نموده و آن را درون ماتریس **b_tx** ذخیره نمایید. ماتریس **b_tx** یک ماتریس با اندازه‌ی pkt_size×k می‌باشد.

۲. **نگاشت بیت به شماره‌ی سمبل:** بردار جدیدی متناظر با ماتریس **b_tx** و با نام **sym_idx** تولید نمایید و متناظر با هر سطر ماتریس **b_tx** یک عدد صحیح معادل با آن بیت‌ها در **sym_idx** قرار دهید. دقت نمایید در مدولاسیون FSK نیازی به کدگذاری گری نیست. بنابراین در برنامه از پرچم **flag_gray_encode** برای اعمال یا عدم اعمال کدگزاری گری استفاده نمایید.

۳. **تولید نمونه‌های زمانی سمبل ارسالی:** با استفاده از تابع **pulse_modulation** ابتدا نمونه‌های زمانی متناظر با هر شماره‌ی سمبل را تولید و در نهایت نمونه‌های زمانی یک بسته‌ی داده را آماده‌ی ارسال کرده و حاصل را در بردار **tx_smpl** ذخیره نمایید. دقت نمایید که انرژی هر سمبل برابر با $E_s = 1$ باشد تا انرژی متوسط سمبل‌ها نیز برابر با $E_{s,avg} = 1$ شود.

توضیحات تکمیلی: در این آزمایش نمونه‌های سیگنال در باندپایه ایجاد می‌شود. چون سیگنال در باندمیانی تولید نمی‌شود نیازی به دانستن f_0 نیست. از این رو می‌بایست از فرمول معادل‌باندپایه‌ی سیگنال مدولاسیون FSK استفاده نمود. فرمول‌هایی که در بخش توضیحات این آزمایش نوشته شده مربوط به حالت سیگنال پیوسته است. در حالت گسسته ابتدا یک بردار زمان به صورت **t = (0:smpl_per_symbl-1)*ts** ایجاد نمود که در آن $t_s = 1/f_s$ زمان نمونه‌برداری می‌باشد. سپس با استفاده از این بردار که در واقع نقاط نمونه‌برداری از معادل پایین‌گذر است، شکل گسسته‌ی هر کدام از سمبل‌ها به دست می‌آید. ضریب مناسبی به جای $\sqrt{\frac{2E_s}{T}}$ می‌بایست گذاشته شود تا انرژی هر سمبل واحد شود. حال نمونه‌های سمبل‌های مختلف را پشت سر هم قرار می‌دهیم. در واقع اگر تعداد سمبل‌ها برابر ۱۰ باشد و `smpl_per_symbl` برابر با باشد، بردار **tx_smpl** یک بردار با ۶۴۰ سطر و یک ستون خواهد بود. با استفاده از یک حلقه‌ی for می‌توان پس از تولید هر یک از سمبل‌ها آن را در موقعیت مناسب در بردار **tx_smpl** قرار داد.

آزمایش ۳-۶: مدل‌سازی کانال

۱. **افزودن تأخیر در کانال:** برای شبیه‌سازی تأخیر در کانال به اندازه‌ی پارامتر **chnl_delay_in_smpl** به ابتدای بردار **tx_smpl** صفر اضافه نمایید. حاصل را درون بردار **tx_smpl_delayed** قرار دهید. دقت نمایید که طول بردار افزایش می‌یابد و می‌بایست در ادامه‌ی برنامه اثر این افزایش لحاظ شود. در این‌جا فعلا مقدار تأخیر برابر با صفر قرار دهید.

۲. **اعمال اختلاف فاز در کانال:** برای اعمال اختلاف فاز در کانال می‌بایست بردار **tx_smpl_delayed** را در **exp(1i*chnl_phase_offset)** ضرب شود. حاصل را در بردار **rx_smpl** قرار دهید. در این‌جا نیز آفست فاز را برابر با صفر قرار دهید.

۳. **شبیه‌سازی کانال با نویز سفید گاوسی**

گام ۳. **تعیین واریانس نویز بر اساس نسبت سیگنال به نویز (E_b/N_0):** با توجه به تولید هر سمبل با انرژی واحد، مقدار انرژی متوسط بیت را به دست آورده و برابر متغیر **Eb** قرار دهید. مقدار E_b/N_0 را برابر با 10dB در نظر بگیرید (به عبارتی پارامترهای **snr_min** و **snr_max** برابر 10 تنظیم می‌شوند). سپس واریانس نویز را بر اساس نسبت E_b/N_0 به دست آورید و آن را درون متغیر **var_noise** قرار دهید.

گام ۴. **افزودن نویز به سیگنال:** با استفاده از تابع **randn** یک بردار نویز مختلط با واریانس **var_noise** و ابعاد برابر با **tx_smpl** تولید نمایید و آن را **noise_smpl** نامگذاری نمایید. سپس این بردار را با بردار **rx_smpl** جمع نموده و حاصل را **rx_smpl_noise** بنامید.

آزمایش ۴-۶: پیاده‌سازی گیرنده‌ی مدولاسیون FSK

در این‌جا می‌خواهیم گیرنده‌ی مدولاسیون FSK خطی را پیاده‌سازی نماییم.

۱. **دمدولاسیون یا آشکارسازی پالس:** نمونه‌های ارسال شده به همراه نویز (**tx_smpl_noise**) را به عنوان نمونه‌های دریافتی درون بردار **rx_smpl** قرار دهید. با استفاده از تابع **pulse_demodulation** با ورودی‌های نمونه‌های سیگنال دریافتی (**rx_smpl**)، نام مدولاسیون (**modulation**)، مرتبه‌ی مدولاسیون (**M**)، نرخ نمونه‌برداری (**fs**)، تعداد نمونه‌های هر سمبل (**smpl_per_symbl**)، نام تابع شکل‌دهنده‌ی پالس (**pulse_name**) و روش تشخیص قدرت سیگنال دریافتی در راستای سیگنال‌های پایه (**mode**)، خروجی سمبل‌های خروجی (**rx_sym**) و اندیس سمبل‌های آشکارشده (**det_sym_idx**) را به دست آورید. (در این‌جا به پارامترهای ورودی **pulse_name**، **mode** نیازی نیست.)

۲. **محاسبه‌ی خطای سمبل:** با مقایسه‌ی تعداد اختلاف‌های بردارهای **det_sym_idx** و **sym_idx** و محاسبه نسبت اختلاف این دو بردار به تعداد کل سمبل‌ها، خطای سمبل را به دست آورده و آن را در متغیر **ser** قرار دهید.

۳. **تبدیل سمبل به بیت و محاسبه‌ی خطای بیت:** بیت‌های متناظر با سطرهای **det_sym_idx** را به دست آورید و آن را درون متغیر **det_bit** قرار دهید. با مقایسه‌ی تعداد اختلاف ماتریس‌های **det_bit** و **b_tx** و محاسبه نسبت تعداد اختلاف این دو ماتریس به تعداد کل بیت‌ها، خطای بیت را به دست آورده و آن را در متغیر **ber** قرار دهید.

آزمایش ۵-۶: خواسته‌های کلی

۱. **رسم نمودار نرخ خطای بیت:** با تغییر نسبت سیگنال به نویز بین 0 تا 10dB نرخ خطای بیت را برای مدولاسیون‌های 2FSK، 4FSK و 8FSK را به دست آورده و در یک نمودار رسم نمایید و آن را با خروجی تابع **berawgn** مقایسه نمایید.

۲. **اثر تأخیر در کانال:**

در این‌جا فرض نمایید مدولاسیون 2FSK است و نسبت سیگنال به نویز (E_b/N_0) را از 0dB تا 10dB تغییر دهید.

گام ۱. **اثر تأخیر در کانال بر روی نرخ خطای بیت و منظومه‌ی سیگنالی:** تأخیر کانال را برابر با گرد شده‌ی 0.1، 0.5 و 0.8 برابر طول یک سمبل قرار دهید (یک عدد صحیح معادل تعداد نمونه‌های زمانی تأخیر یافته). نمودار نرخ خطای بیت را به دست آورد و با حالت بدون تأخیر مقایسه نمایید. منظومه سیگنالی هر دو حالت فوق را نیز با یکدیگر مقایسه نمایید. علت مشاهدات خود را توضیح دهید.

گام ۲. **اضافه کردن هدر:** برای یافتن نقطه‌ی شروع درست داده‌ها، تعدادی داده‌ی مشخص به عنوان هدر در ابتدای رشته بیت‌های ارسالی قرار دهید. نمونه‌های زمانی مربوط به هدر را از بردار **tx_smpl** جدا کرده و در متغیر **hdr_smpl** ذخیره نمایید. بیت‌های هدر می‌بایست به صورت زیر باشد. این رشته بیت خواص خودهمبستگی خوبی دارد.

```
hder_bit = repmat(de2bi(hex2dec('1C6387FF5DA4FA325C895958DC5')),1,k);
```

گام ۳. **تخمین زمان شروع بسته و جبران اثر تأخیر کانال:** با استفاده از همبستگی بردار **tx_smpl** و نمونه‌های زمانی هدر **hdr_smpl**، نقطه‌ی شروع بسته‌ی داده را تخمین بزنید. با مقایسه‌ی نرخ خطای بیت حالت بدون تأخیر و حالتی که نقطه‌ی شروع داده‌ها تخمین زده می‌شود، از عملکرد برنامه‌ی خود اطمینان حاصل نمایید.

۳. **اعمال فاز در کانال و تخمین فاز:**

در این‌جا فرض نمایید مدولاسیون 2FSK است و نسبت سیگنال به نویز (E_b/N_0) را از 0dB تا 10dB تغییر دهید.

گام ۱. **اثر فاز اضافی کانال بر نرخ خطای بیت:** اختلاف فاز کانال را برابر ۳۰ درجه قرار دهید. ابتدا نمودار نرخ خطای بیت را به دست آورده و با حالت بدون اختلاف فاز مقایسه نمایید. منظومه سیگنالی هر دو حالت فوق را نیز با یکدیگر مقایسه نمایید. علت مشاهدات خود را توضیح دهید.

گام ۲. **تخمین فاز اضافی کانال:** اگر به ابتدای بسته‌ی ارسالی هدر بخش قبل اضافه شود، با استفاده از همبستگی بردار **rx_smpl_noise** و نمونه‌های زمانی هدر **hdr_smpl**، فازی را که کانال به سیگنال اضافه کرده است را به دست آورید. (**راهنمایی:** به فاز در قله‌ی خروجی همبستگی دقت نمایید.)

گام ۳. **جبران‌سازی فاز کانال:** تلاش کنید با اعمال ضریب مناسب اثر فاز اضافی کانال را جبران نمایید. (**cmpnst_mode = 2**)

آزمایش ۷-۶: پیاده‌سازی سخت‌افزاری گیرنده و فرستنده‌ی FSK

در این بخش **flg_hrdwr_usg = 1** و **rx_alg = 1** می‌باشد.

۱. **ارسال نمونه‌ها با استفاده از ADALM-PLUTO:** بردار **tx_smpl** مربوط به مدولاسیون 2FSK (هدر اضافه شود) را با استفاده از دستور **transmitRepeat** به مدت زمان **stop_time** ثانیه به صورت پی‌درپی در فضا ارسال نمایید. این دستور داده‌ها را از طریق USB به رادیو نرم‌افزار ارسال می‌نماید و در حافظه‌ی سخت‌افزار ذخیره‌می‌نماید. فرستنده داده‌ها را از حافظه‌ی رادیونرم‌افزار قرایت کرده و مدام ارسال می‌نماید. برای این منظور ابتدا می‌بایست پس از پیکربندی رادیو نرم‌افزار یک شیء فرستنده ایجاد نموده و فرکانس مرکزی آن را بر روی 2400MHz تنظیم کرده و بهره‌ی آن را برابر با 0dB تنظیم نمایید.

۲. **دریافت سیگنال با استفاده از ADALM-PLUTO:** ابتدا شیء مربوط به گیرنده را تعریف نمایید. فرکانس مرکزی آن را بر روی 2400MHz تنظیم کرده و بهره‌ی آن را به صورت Manual و برابر 20dB تنظیم نمایید. به علت اطمینان از دریافت یک بسته‌ی کامل، تعداد نمونه‌های زمانی دریافتی را برابر با دو برابر تعداد نمونه‌های زمانی بسته درنظر بگیرید. به منظور دریافت داده، شیء گیرنده را فراخوانی نمایید و حاصل را درون متغیر **rx_smpl** قرار دهید.

۳. **طیف سیگنال ارسالی:** طیف فرکانسی سیگنال **rx_smpl** دریافتی از ADALM-PLUTO را برای مدولاسیون 2FSK را رسم نماید. در مورد پهنای‌باند و سطح باندهای جانبی آن توضیحاتی ارایه دهید. برای این منظور از برنامه‌های قبلی استفاده نمایید.

۴. **دمدولاسیون و مشاهده‌ی منظومه‌ی سیگنالی:**

گام ۱. مشابه قبل عمل دمدولاسیون را انجام داده و منظومه‌ی سیگنالی مربوط به **rx_sym** را رسم نمایید. منظومه‌ی سیگنالی باید بتواند به صورت به‌لحظه به‌روز شود.

گام ۲. علت تفاوت منظومه‌ی سیگنالی در هنگام کار با سخت‌افزار را با حالت شبیه‌سازی را با مشاهده‌ی تفاوت شکل موج سیگنال دریافتی در حالت کار با سخت‌افزار و شبیه‌سازی توضیح دهید.

گام ۳. با تغییر ویژگی‌های زیر برای سخت‌افزار مجدد مشاهدات خود را بیان نمایید. (توضیح بیشتر با جستجوی عبارت DC Offset Tracking در راهنمای نرم‌افزار MATLAB)

```
rx.ShowAdvancedProperties = true;
rx.EnableBasebandDCCorrection = false;
```

۵. **جبران‌سازی دامنه، فاز و تأخیر و مشاهده‌ی منظومه‌ی سیگنالی:** با یافتن ابتدای بسته‌ها و جبران فاز، دامنه و تأخیر، منظومه‌ی سیگنالی مربوط به **rx_sym** را رسم نمایید. احتمال خطای بیت را نیز در این حالت محاسبه نمایید.

۶. **اثر Δf بر روی پهنای‌باند ارسالی:** طیف فرکانسی سیگنال **rx_smpl** دریافتی از ADALM-PLUTO را برای مدولاسیون 4FSK رسم نماید. در مورد پهنای‌باند و سطح باندهای جانبی آن توضیحاتی ارایه دهید. با تغییر Δf از $\frac{1}{2T}$ به $\frac{2}{T}$ پهنای‌باند سیگنال دریافتی هر حالت را به دست آورده و با یکدیگر مقایسه نمایید.

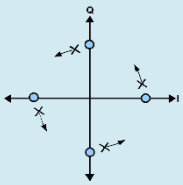
مراجع

[1] J. G. Proakis and M. Salehi, *Digital Communications*. Boston: McGraw-Hill, 2008.

[2] R. W. Stewart, K. Barlee, L. Crockett, and D. Atkinson, *Software Defined Radio using MATLAB & Simulink and the RTL-SDR*. 2015.

بسمه تعالی

آزمایش هفتم



آشکارسازی ناهمدوس

شرح آزمایش

آزمایش ۱-۷: مقداردهی‌های اولیه

در این آزمایش بنا داریم تعدادی بسته‌ی داده را برای ارسال از طریق مدولاسیون FSK ناهمدوس و DBPSK آماده نماییم. از آن‌جا که در این آزمایش بناست این فرستنده با استفاده از رادیو نرم‌افزار ADALM-PLUTO پیاده‌سازی شود، می‌بایست با استفاده از پارامترها و ملاحظات عملی، این کار انجام شود.

پارامترهایی که در این پیاده‌سازی اهمیت دارد شامل نرخ نمونه‌برداری (**fs**)، تعداد نمونه‌های هر سمبل (**smpl_per_symbl**)، تعداد سمبل‌های ارسالی در بسته‌ی داده (**pkt_size**) و مدت زمان ارسال داده (**stop_time**) می‌باشد. در این آزمایش مقدار پارامترهای مورد نیاز به صورت جدول 3 می‌باشد.

جدول 4 پارامترهای آزمایش ۷

پارامتر	fs	smpl_per_symbl	pkt_size	stop_time
مقدار	10MHz	FSK: 64 DBPSK: 8	شبیه‌سازی: ۱۰۰۰۰۰ سخت‌افزار: ۱۰۰۰	۱۰۰ ثانیه

دقت نظر داشته باشید که در این آزمایش نیز تنها می‌خواهیم برنامه‌هایی که در آزمایش قبل نوشته شده است را تکمیل نماییم.

۱. **تولید M-File مربوط به مقدار دهی اولیه:** قالب کلی پارامترهای مورد نیاز در این آزمایش، مانند آزمایش قبل است و فایل **dcl_init.m** می بایست پارامتری به نام **mod_det_opt** داشته باشد که نوع آشکارسازی را که می‌تواند همدوس (**coherent**) یا ناهمدوس (**noncoherent**) باشد را مشخص می‌نماید. برنامه با توجه به این گزینه نوع گیرنده و نحوه‌ی ارسال فرستنده مدیریت می‌شود. هم‌چنین پارامتر **phase_amb_opt** برای نوع اعمال ابهام فاز در نظر گرفته شود. این فایل می‌بایست از برنامه‌ی اصلی فراخوانی نمایید.

آزمایش ۲-۷: پیاده‌سازی مدولاسیون FSK ناهمدوس

۱. **شبیه‌سازی کانال با فاز تصادفی:** برای شبیه‌سازی کانال با فاز تصادفی می‌بایست بردار **tx_smpl_delayed** را در **exp(1i*chnl_phase_offset_amb)** ضرب شود. حاصل را در بردار **tx_smpl** قرار دهید. **chnl_phase_offset_amb** را به صورت یک بردار تصادفی بین 0 و 2π تولید کنید. طول این بردار برابر با طول بردار **tx_smpl** می‌باشد. دقت کنید که فاز اضافه شده در طول هر سمبل ثابت باشد. در این حالت **phase_amb_opt** برابر با 0 است. (**راهنمایی:** از دستور **kron** می‌توان برای این کار استفاده نمود.)

۲. **شبیه‌سازی مدولاسیون FSK ناهمدوس:** برای پیاده‌سازی این مدولاسیون، همانند FSK همدوس که در آزمایش قبل پیاده شد عمل نمایید. نمودار احتمال خطای بیت را برای مدولاسیون 4FSK و 2FSK به دست آورده و نتیجه‌ی را با دستور **berawgn** صحت‌سنجی نمایید. دقت نمایید در حالت آشکارسازی ناهمدوس باید حداقل فاصله‌ی فرکانسی $1/T_s$ باشد. آشکارساز این مدولاسیون نیز به صورت $\arg\max|\underline{r}_\ell . \underline{S}_{\ell,m}|$ می‌باشد.

۳. **پیاده‌سازی سخت‌افزاری مدولاسیون FSK ناهمدوس:** با فرض این‌که توان فرستنده‌ی ADALM-PLUTO برابر با 0dBm و بهره‌ی گیرنده برابر با 20dB باشد و از آنتن به منظور ارتباط فرستنده و گیرنده استفاده می‌شود، خطای آشکارسازی با استفاده از آشکارسازی ناهمدوس را برای مدولاسیون 2FSK محاسبه نمایید. هم‌چنین در این حالت منظومه‌ی سیگنالی این مدولاسیون را نیز رسم نمایید. در این‌جا از هدر تنها برای به دست آوردن ابتدای بسته‌ی ارسالی استفاده می‌شود و هیچ گونه جبران‌سازی فازی صورت نمی‌پذیرد.

آزمایش ۳-۷: پیاده‌سازی مدولاسیون DBPSK

۱. **تولید بیت:** با استفاده از تابع **bit_gen** تعدادی بیت ۰ و ۱ را متناسب با طول بسته‌ی داده و با در نظر گرفتن پارامترهای مدولاسیون DBPSK تولید نموده و آن را درون ماتریس **b_tx** ذخیره نمایید. ماتریس **b_tx** یک ماتریس با اندازه‌ی pkt_size×1 می‌باشد.

۲. **کد کردن بیت‌ها:** با استفاده از **Error! Reference source not found.** رشته بیت‌های e_n را تولید نمایید و آن را د رون بردار **enc_b_tx** قرار دهید.

۳. **نگاشت بیت به سمبل:**

گام ۱. **کدگذاری گری بیت‌های تولیدی:** با وجود این‌که در این‌جا نیازی به کدگذاری گری نیست ولی به منظور کلی بودن برنامه این مرحله نیز انجام می‌شود. با استفاده از تابع **gray_code**، ماتریس مربوط به کدگذاری گری مدولاسیون DBPSK را تولید کنید و آن را در ماتریس **b_gray** ذخیره کنید. بردار جدیدی متناظر با ماتریس **enc_b_tx** و با نام **sym_idx** تولید نمایید و نگاشتی یک‌به‌یک بین سطرهای ماتریس **enc_b_tx** و ماتریس **b_gray** برقرار کنید. به عبارتی سطر iام بردار **sym_idx**، شماره‌ی سطری از **b_gray** را نشان می‌دهد که برابر با سطر iام ماتریس **enc_b_tx** باشد.

گام ۲. **تولید سمبل‌های ارسالی:** با استفاده از تابع **constellation**، تمامی سمبل‌های ارسالی مدولاسیون DBPSK را تولید کرده و سمبل‌های متناظر با هر سطر بردار **sym_idx** را تولید کنید. این سمبل‌ها را در بردار **mod_sym** ذخیره نمایید.

۴. **شکل‌دهی پالس ارسالی:** با استفاده از تابع **pulse_modulation** نمونه‌های ارسالی مربوط به یک بسته‌ی داده را آماده‌ی ارسال کرده و حاصل را در بردار **tx_smpl** ذخیره نمایید. تابع شکل‌دهی پالس را مثلثی در نظر بگیرید.

آزمایش ۴-۷: مدل‌سازی کانال

۱. **افزودن تأخیر در کانال:** برای شبیه‌سازی تأخیر در کانال به اندازه‌ی پارامتر **chnl_delay_in_smpl** به ابتدای بردار **tx_smpl** صفر اضافه نمایید. حاصل را درون بردار **tx_smpl_delayed** قرار دهید. دقت نمایید که طول بردار افزایش می‌یابد و می‌بایست در ادامه‌ی برنامه اثر این افزایش لحاظ شود. در این‌جا فعلا مقدار تأخیر برابر با صفر قرار دهید.

۲. **اعمال ابهام فاز کانال:** مدولاسیون DPSK به تغییرات شدید فاز حساس است و بنابراین باید ابهام فاز ایجاد شده خیلی زیاد نباشد. ابهام فاز ایجاد شده برای سمبل‌ها را به صورت زیر مدل می‌کنیم

```
chnl_phase_offset_amb = 0:phase_step:(pkt_size-1)*phase_step
```

پارامتر **phase_step**، مقدار ابهام فاز ایجاد شده برای یک سمبل نسبت به سمبل قبلی را مشخص می‌کند. حال با استفاده از دستور **kron**، فاز تولید شده را برای کل نمونه‌های هر سمبل تکرار کنید. با ضرب کردن عبارت **exp(1j* chnl_phase_offset_amb)** در **tx_smpl_delayed**، ابهام فاز ایجاد شده را به سینگال ارسالی اضافه نمایید و حاصل را در **tx_smpl** ذخیره نمایید. در این‌جا **phase_step** را برابر با ۱۰ درجه قرار دهید.

۳. **شبیه‌سازی کانال با نویز سفید گاوسی**

گام ۵. **تعیین واریانس نویز بر اساس نسبت سیگنال به نویز (E_b/N_0)**: ابتدا بر اساس خروجی تابع **constellation**، مقدار متوسط انرژی سمبل را به دست آورید (**Es_avg**). حال با استفاده از متوسط انرژی سمبل، مقدار انرژی متوسط بیت را به دست آورده و برابر متغیر **Eb** قرار دهید. مقدار E_b/N_0 را برابر با 10dB در نظر بگیرید (به عبارتی پارامترهای **snr_min** و **snr_max** برابر 10 تنظیم می‌شوند). سپس واریانس نویز را بر اساس نسبت E_b/N_0 به دست آورید و آن را درون متغیر **var_noise** قرار دهید.

گام ۶. **افزودن نویز به سیگنال**: با استفاده از تابع **randn** یک بردار نویز مختلط با واریانس **var_noise** و ابعاد برابر با **tx_smpl** تولید نمایید و آن را **noise_smpl** نامگذاری نمایید. سپس این بردار را با بردار **tx_smpl** جمع نموده و حاصل را **tx_smpl_noise** بنامید.

آزمایش ۵-۷: پیاده‌سازی گیرنده‌ی مدولاسیون DBPSK

در ادامه‌ی فایل آزمایش قبل می‌خواهیم گیرنده‌ی مدولاسیون DBPSK را پیاده‌سازی نماییم.

۱. **دمدولاسیون یا آشکارسازی پالس**: نمونه‌های ارسال شده به همراه نویز (**tx_smpl_noise**) را به عنوان نمونه‌های دریافتی درون بردار **rx_smpl** قرار دهید. با استفاده از تابع **pulse_demodulation** با ورودی‌های نمونه‌های سیگنال دریافتی (**rx_smpl**)، نام مدولاسیون (**modulation**)، مرتبه‌ی مدولاسیون (**M**)، نرخ نمونه‌برداری (**fs**)، تعداد نمونه‌های هر سمبل (**smpl_per_symbl**)، نام تابع شکل‌دهنده‌ی پالس (**pulse_name**)، روش تشخیص قدرت سیگنال دریافتی در راستای سیگنال‌های پایه (**mode**) و شیوه‌ی آشکارسازی (**det_opt**)، سمبل‌های دریافتی (**rx_sym**) و اندیس سمبل‌های آشکارشده (**det_sym_idx**) را به دست آورید.

۲. **محاسبه‌ی خطای سمبل**: با مقایسه‌ی تعداد اختلاف‌های بردارهای **det_sym_idx** و **sym_idx** و محاسبه نسبت اختلاف این دو بردار به تعداد کل سمبل‌ها، خطای سمبل را به دست آورده و آن را در متغیر **ser** قرار دهید.

۳. **تبدیل سمبل و بیت و تبدیل از کدگذاری گری به کدگذاری باینری**: بیت‌های متناظر با سطرهای **det_sym_idx** را به دست آورید و آن را درون متغیر **det_bit_gray** قرار دهید. با وجود این‌که در این‌جا عکس تبدیل گری مطرح نیست به منظور کلی ماندن برنامه بیت‌های گری را به کدگذاری باینری تبدیل نماید و آن را درون بردار **det_bit** قرار دهید.

۴. **محاسبه‌ی خطای بیت**: با مقایسه‌ی تعداد اختلاف ماتریس‌های **det_bit** و **b_tx** و محاسبه نسبت تعداد اختلاف این دو ماتریس به تعداد کل بیت‌ها، خطای بیت را به دست آورده و آن را در متغیر **ber** قرار دهید.

آزمایش ۶-۷: خواسته‌های کلی

۱. **رسم نمودار نرخ خطای بیت**: با تغییر نسبت سیگنال به نویز بین 0 تا 10dB نرخ خطای بیت را برای مدولاسیون‌های DBPSK به دست آورده و در یک نمودار رسم نمایید و آن را با خروجی تابع **berawgn** مقایسه نمایید. این کار را برای شکل موج مثلثی انجام دهید.

۲. **اثر تغییرات فاز بر روی عملکرد**: با تغییر پارامتر **phase_step**، تأثیر تغییرات فاز را بر روی عملکرد این سامانه ارزیابی نمایید.

آزمایش ۷-۷: پیاده‌سازی سخت‌افزاری مدولاسیون DBPSK

۱. **ارسال نمونه‌ها با استفاده از ADALM-PLUTO:** بردار **`tx_smpl`** مربوط به مدولاسیون DBPSK (هدر اضافه شود) را با استفاده از دستور **`transmitRepeat`** به مدت زمان **`stop_time`** ثانیه به صورت پی‌درپی در فضا ارسال نمایید. این دستور داده‌ها را از طریق USB به رادیو نرم‌افزار ارسال می‌نماید و در حافظه‌ی سخت‌افزار ذخیره‌می‌نماید. فرستنده داده‌ها را از حافظه‌ی رادیونرم‌افزار قرایت کرده و مدام ارسال می‌نماید. برای این منظور ابتدا می‌بایست پس از پیکربندی رادیو نرم‌افزار یک شیء فرستنده ایجاد نموده و فرکانس مرکزی آن را بر روی 2400MHz تنظیم کرده و بهره‌ی آن را برابر با 0dB تنظیم نمایید.

۲. **دریافت سیگنال با استفاده از ADALM-PLUTO:** ابتدا شیء مربوط به گیرنده را تعریف نمایید. فرکانس مرکزی آن را بر روی 2400MHz تنظیم کرده و بهره‌ی آن را به صورت Manual و برابر 20dB تنظیم نمایید. به علت اطمینان از دریافت یک بسته‌ی کامل، تعداد نمونه‌های زمانی دریافتی را برابر با دو برابر تعداد نمونه‌های زمانی بسته درنظر بگیرید. به منظور دریافت داده، شیء گیرنده را فراخوانی نمایید و حاصل را درون متغیر **`rx_smpl`** قرار دهید.

۳. **دمدولاسیون و مشاهده‌ی منظومه‌ی سیگنالی:** مشابه قبل عمل دمدولاسیون را انجام داده و منظومه‌ی سیگنالی مربوط به **`rx_sym`** را رسم نمایید و مشاهدات خود را یادداشت نمایید. منظومه‌ی سیگنالی باید بتواند به صورت به‌لحظه به‌روز شود.

مراجع

[3] [1] R. W. Stewart, K. Barlee, L. Crockett, and D. Atkinson, *Software Defined Radio using MATLAB & Simulink and the RTL-SDR*. 2015.

بسمه تعالی

آزمایش هشتم

انتقال دیجیتال از درون کانال باندمحدود AWGN

شرح آزمایش

آزمایش ۸-۱: مقداردهی‌های اولیه

در این آزمایش بنا داریم تعدادی بسته‌ی داده را برای ارسال از طریق مدولاسیون 2PAM آماده نماییم. از آن‌جا که در این آزمایش بناست این فرستنده با استفاده از رادیو نرم‌افزار ADALM-PLUTO پیاده‌سازی شود، می‌بایست با استفاده از پارامترها و ملاحظات عملی، این کار انجام شود.

پارامترهایی که در این پیاده‌سازی اهمیت دارد شامل نرخ نمونه‌برداری (**fs**)، تعداد نمونه‌های هر سمبل (**smpl_per_symbl**)، تعداد سمبل‌های ارسالی در بسته‌ی داده (**pkt_size**) و مدت زمان ارسال داده (**stop_time**) می‌باشد. در این آزمایش مقدار پارامترهای مورد نیاز به صورت جدول 3 می‌باشد.

جدول 5 پارامترهای آزمایش ۷

پارامتر	fs	smpl_per_symbl	pkt_size	stop_time
مقدار	10MHz	32	شبیه‌سازی: ۱۰۰۰۰۰ سخت‌افزار: ۱۰۰۰	۱۰۰ ثانیه

دقت نظر داشته باشید که در این آزمایش نیز تنها می‌خواهیم برنامه‌هایی که در آزمایش قبل نوشته شده است را تکمیل نماییم.

آزمایش ۸-۲: پیاده‌سازی مدولاسیون 2PAM

۱. **شبیه‌سازی مدولاسیون 2PAM:** برای پیاده‌سازی این مدولاسیون، همانند آزمایش ۴ یا ۵ عمل نمایید. احتمال خطای بیت را برای نسبت $\frac{E_b}{N_0}$ برابر با 10dB به دست آورید. این عمل را برای دو شکل پالس مستطیلی و Root Raised Cosine ($\beta = 0.9$ و **span_in_symbl = 6**) انجام دهید. (حالت عملکردی دمدولاتور را بر روی فیلتر منطبق تنظیم نمایید.)

۲. **رسم نمودار چشمی:** با استفاده از دستور **eyediagram** نرم‌افزار MATLAB نمودار چشمی مدولاسیون 2PAM را برای دو شکل پالس گفته شده و در دو نسبت E_b/N_0 برابر با 10dB و 40dB رسم نمایید. پارامترهای بر روی **Error! Reference source not found.** را به دست آورید.

آزمایش ۸-۳: مدل‌سازی کانال باندباریک

۱. **تولید یک فیلتر FIR:** با استفاده از دستور **fir1** نرم‌افزار MATLAB یک فیلتر با پهنای‌باند 300kHz و با تعداد ۱۰۰ تپ تولید نمایید. پاسخ فرکانسی و پهنای‌باند نویز فیلتر را به دست آورید و با پهنای‌باند سیگنال ارسالی مقایسه نمایید.

۲. **کانال باند محدود:** پس از اعمال تأخیر و اعمال ابهام فاز، سیگنال ارسالی حاصل را با استفاده از عمل کانولوشن از فیلتر FIR مرحله‌ی قبل عبور دهید. مجدد نمودار چشمی را در دو نسبت E_b/N_0 برابر با 10dB و 40dB رسم نمایید. احتمال خطای بیت را نیز به دست آورید. این عمل برای هر دو شکل پالس گفته شده انجام شود.

آزمایش ۴-۸: پیاده‌سازی کانال باندمحدود

۱. **کار با ADALM-PLUTO:** موارد گفته شده را با استفاده از رادیونرم‌افزار ADALM-PLUTO نیز انجام دهید.

آزمایش ۵-۸: خواسته‌های کلی

۱. **اثر طول شکل پالس root raised cosine:** با تغییر پارامتر span_in_symbl تأثیر آن را بر روی نمودار چشمی و خطای بیت مدولاسیون 2PAM با شکل پالس root raised cosine را مشاهده نمایید.

مراجع

[1] R. W. Stewart, K. Barlee, L. Crockett, and D. Atkinson, *Software Defined Radio using MATLAB & Simulink and the RTL-SDR*. 2015.