



بسمه تعالی

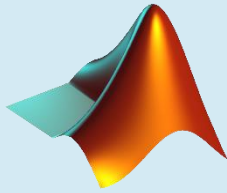
گام‌های تکمیل آزمایشگاه مخابرات دیجیتال

استاد: دکتر علی الفت

ویرایش ۱/۱ (آزمایشی)

فهرست مطالب آزمایشگاه مخابرات دیجیتال

آزمایش اول: آشنایی با نرم افزار MATLAB	۳
آزمایش دوم: پردازش سیگنال دیجیتال با نرم افزار MATLAB	۷
آزمایش سوم: رادیونرم افزار و پایش طیف سیگنال های رادیویی	۱۶
آزمایش چهارم: معرفی مدولاسیون دیجیتال خطی	۱۸
آزمایش پنجم: کاوش در مدولاسیون دیجیتال خطی	۲۱
آزمایش ششم: مدولاسیون FSK همدوس	۲۶
آزمایش هفتم: آشکارسازی ناهمدوس	۳۰
آزمایش هشتم: انتقال دیجیتال از درون کانال باند محدود AWGN	۳۴



آزمایش اول

آشنایی با نرم افزار MATLAB

شرح آزمایش



آزمایش ۱-۱: محاسبات جبری و گرافیک

۱. مقادیر زیر را محاسبه نماید.

ا.

تقریب کسرها:

2.645507812500000
2.645750000000000
2.645751633986928

عدد $\sqrt{7}$:

2.645751311064591

کمینه‌ی فاصله‌ی کسرها تا عدد $\sqrt{7}$:

3.229223373146795e-07

اندیس عدد با کمترین فاصله:

3

ب. عدد 3^{301} را یک بار به صورت یک عدد اعشاری تقریبی با ۱۵ رقم اعشار (نمایش به صورت نماد علمی) و بار دیگر

به صورت یک عدد صحیح دقیق محاسبه نمایید. (راهنمایی: از دستور **format** و دستور **vpa** با ورودی سمبلیک

استفاده نمایید.)

نمایش به صورت نماد علمی:

4.106744371757652e+143

نمایش به صورت عدد صحیح دقیق:

410674437175765127973978082146264947899391086876012309414440570235106991532
497229781400618467066824164751453321793982128440538198297087323698003

ت. 8.881784197001252e-16

ث. 0

۲.

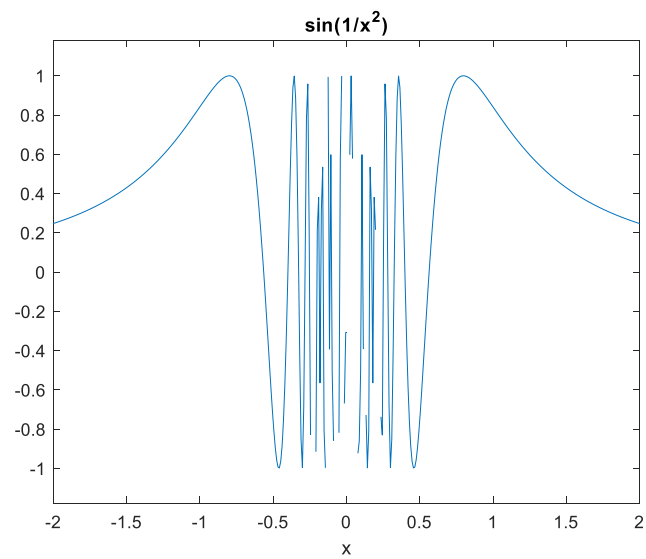
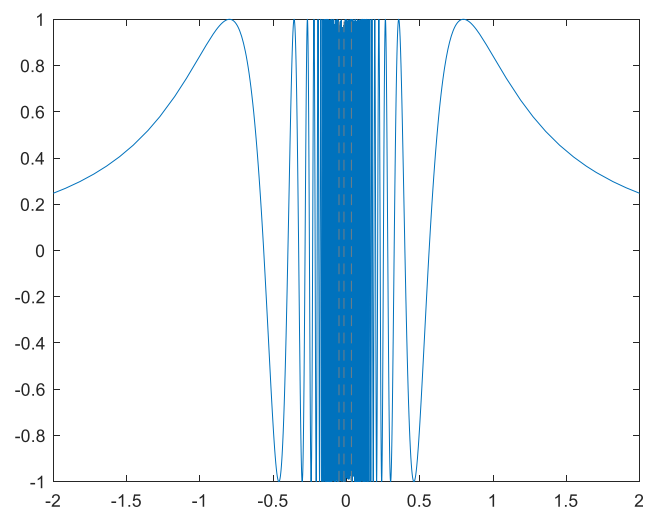
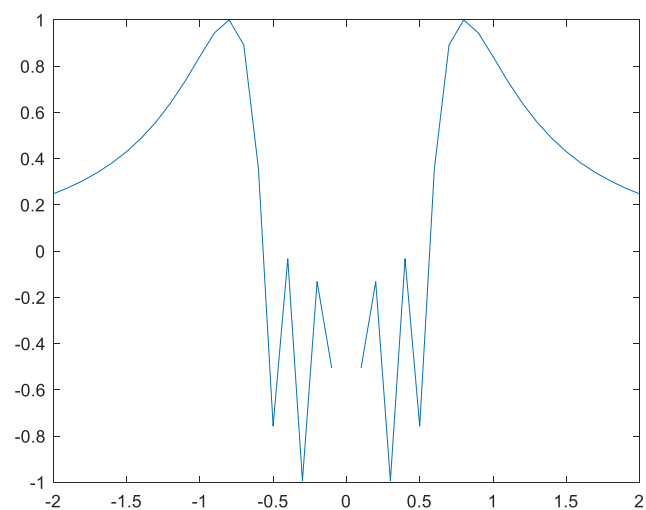
ا. 1.005004168055804

ب. 0.693147180559945

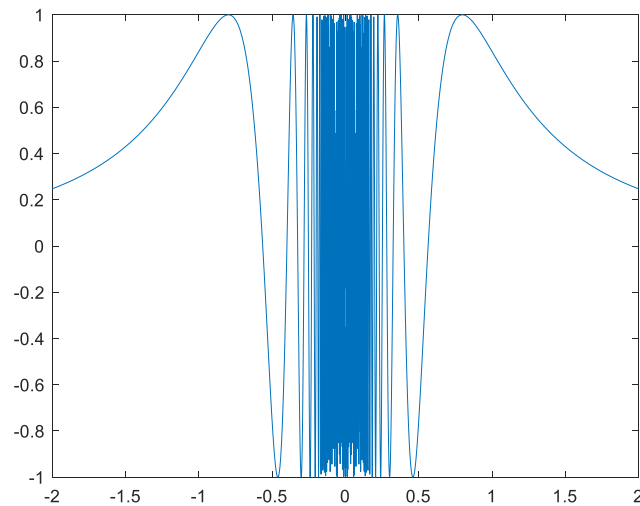
ت. 0.463647609000806

۳.

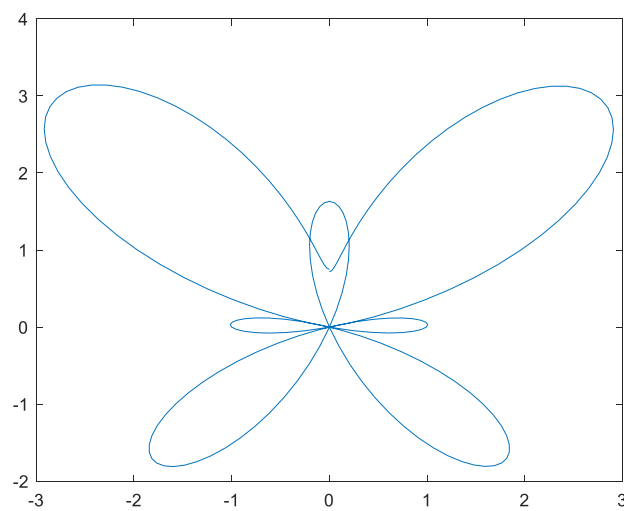
۴. ا.

استفاده از دستور `ezplot`:استفاده از دستور `fplot`:استفاده از دستور `plot` با گام 0.1:

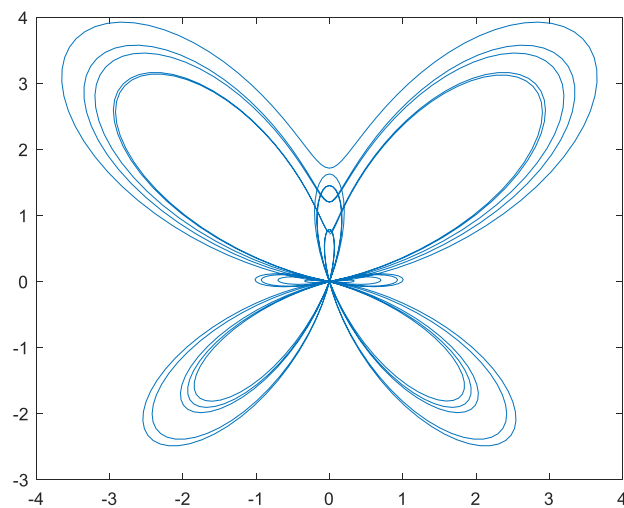
استفاده از دستور plot با گام 0.001:



ب. منحنی پروانه را با استفاده از معادله های پارامتری زیر رسم نمایید.
منحنی پروانه برای $0 \leq t \leq 2\pi$:



منحنی پروانه برای $0 \leq t \leq 10\pi$:



آزمایش ۲-۱: ریاضیات و برنامه نویسی

۱.

ا. 0
ب. 1
ت. 1
ث. 0

۲.

1	0	0	0	0	0	0
1	1	0	0	0	0	0
1	2	1	0	0	0	0
1	3	3	1	0	0	0
1	4	6	4	1	0	0
1	5	10	10	5	1	0
1	6	15	20	15	6	1

۳. 2.3769

۴. اگر اول کد خود بنویسید `rng(1)` خروجی شما به صورت زیر خواهد شد.

18 26 10 16 30 16 24 28 18 28 18 24 14 28 10 24 18 10 12 14

The Vector Generated After 3 Iteration(s).

۵.

خروجی برای `mylcm(4, 5, 6)` 60

خروجی برای `mylcm([6 7 12 15])` 420

خروجی برای `mylcm(4.5, 6)`

Error using mylcm (line)
Arguments must be positive integers.



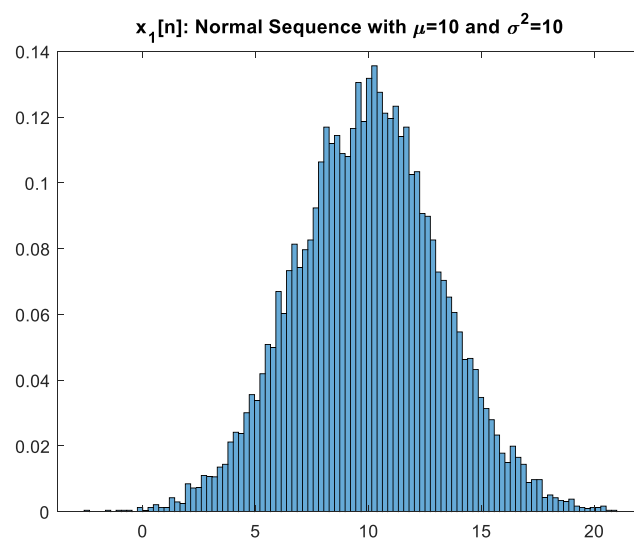
آزمایش دوم

پردازش سیگنال دیجیتال با نرم افزار MATLAB

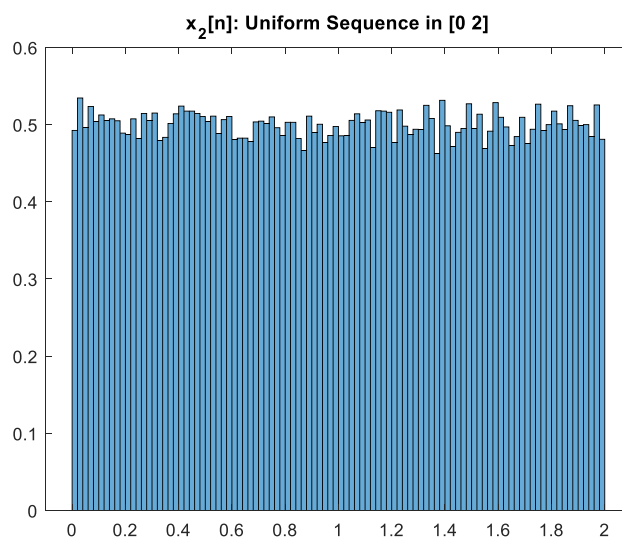
شرح آزمایش 

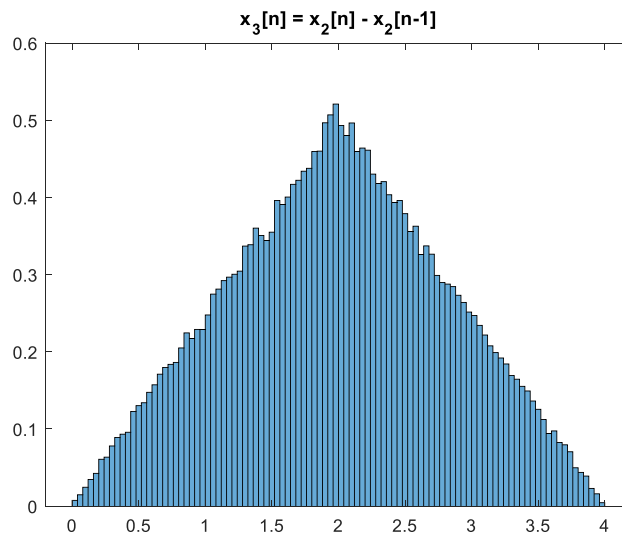
آزمایش ۱-۲: تولید سیگنال دیجیتال و عملیات بر روی آن

۱. دنباله‌های تصادفی: در ابتدای تولید همه‌ی بخش‌ها $\text{rng}(1)$ قرار دهید تا نمودارها یکسان باشد.
ا.

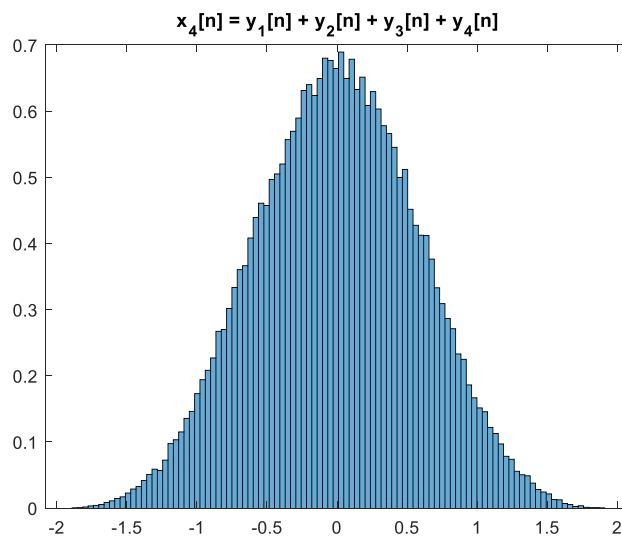


ب.





ت.

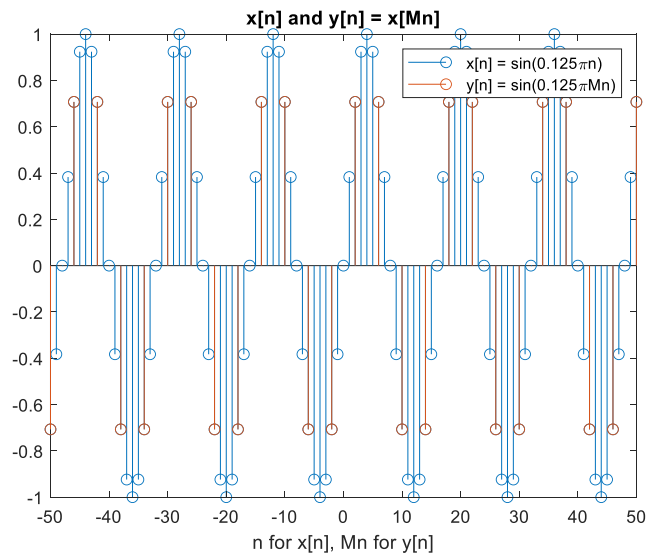


۲. کاهش نرخ نمونه برداری:

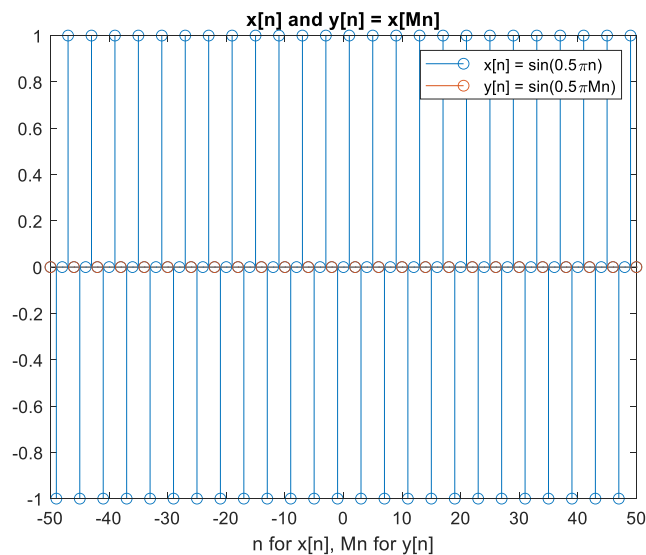
ا. خروجی برای مثال صورت سوال:

$y = 4 \times 1$
-2
3
5
8

ب.



ت. عملیات فوق را برای تابع $x[n] = \sin[0.5\pi n]$ و در بازه $-50 \leq n \leq 50$ تکرار نمایید.



۳. افزایش نرخ نمونه برداری:

ا.

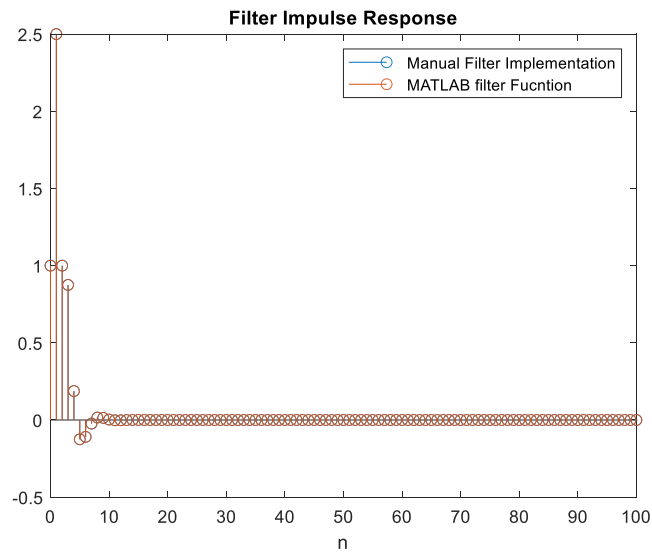
ب.

y = 7×1
-2
0
3
0
5
0
8

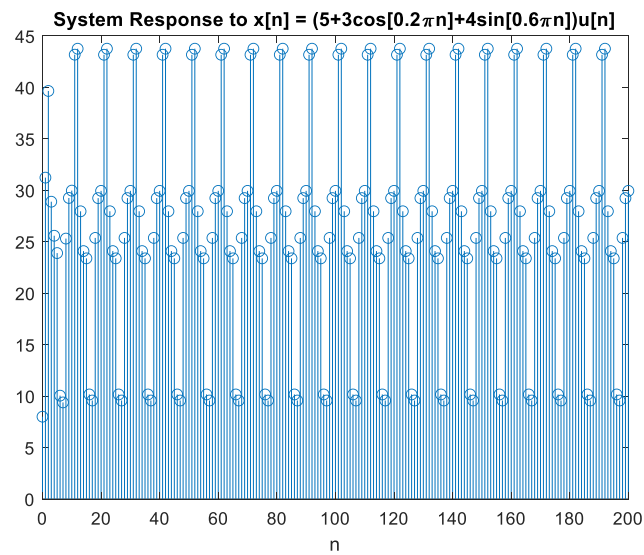
آزمایش ۲-۲: عبور سیگنال گسسته از یک سیستم

۱. حل معادله تفاضلی:

ا.



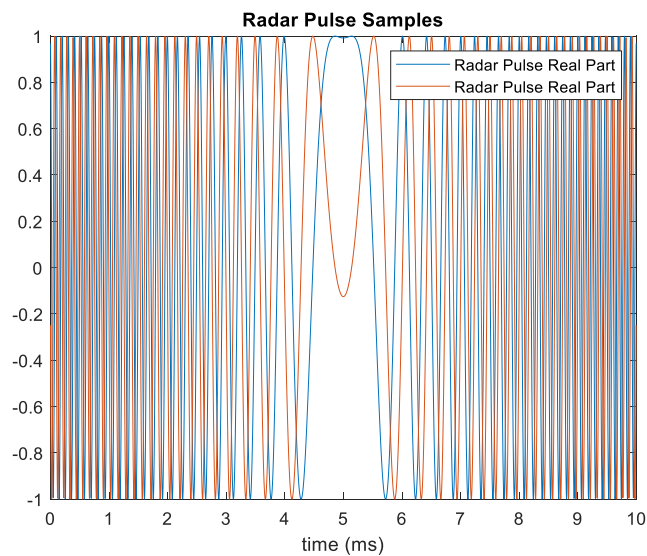
ب.

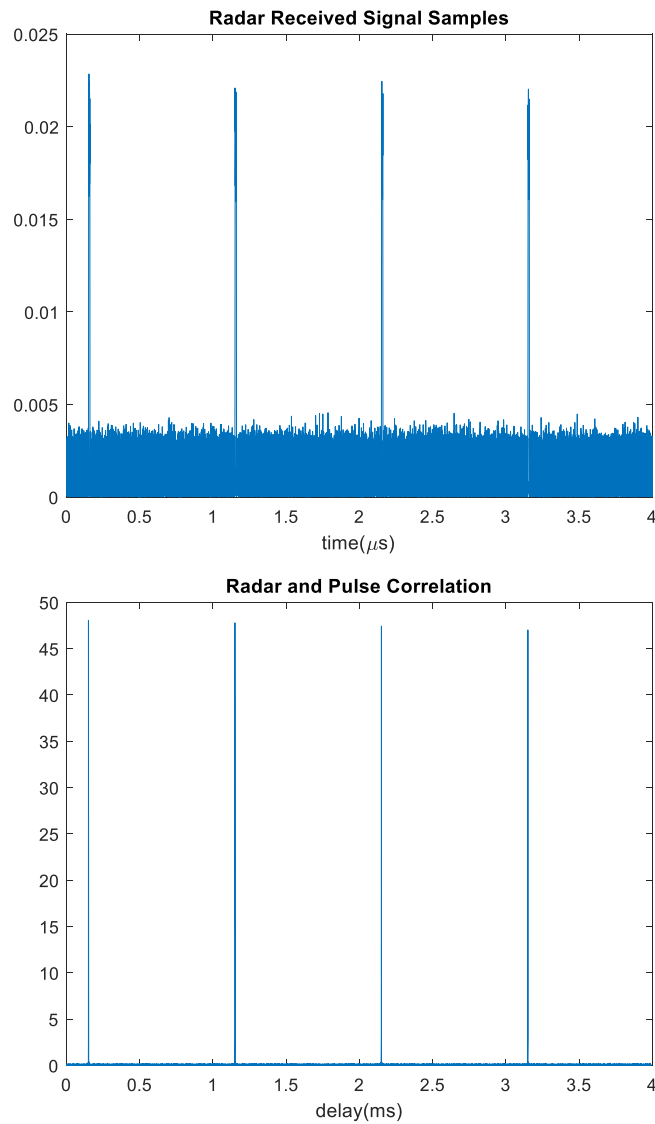


۲. تابع همبستگی و کاربرد آن:

ا.

ب.





distance: 22.50km

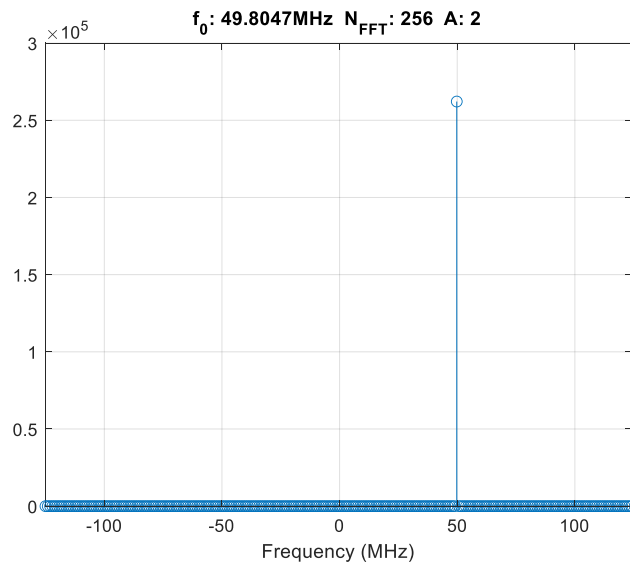
period: 1.00ms

نیازی نیست برای به دست آوردن فاصله و دوره تناوب تکرار پالس از توابع MATLAB برای محاسبه‌ی پیک استفاده کنید. با بزرگ‌نمایی نمودار سوم و قرایت نقطه‌ی قله می‌توان فاصله و دوره تناوب تکرار پالس را به دست آورد.

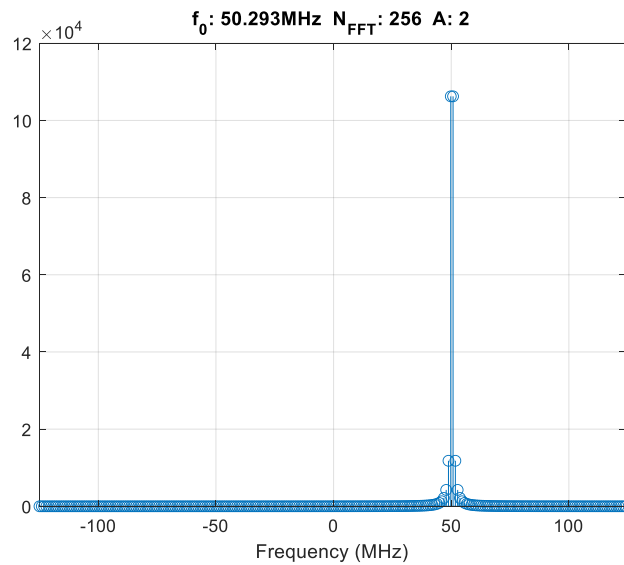
آزمایش ۳-۲: تحلیل حوزه‌ی فرکانس

۱. محاسبه‌ی طیف سیگنال:

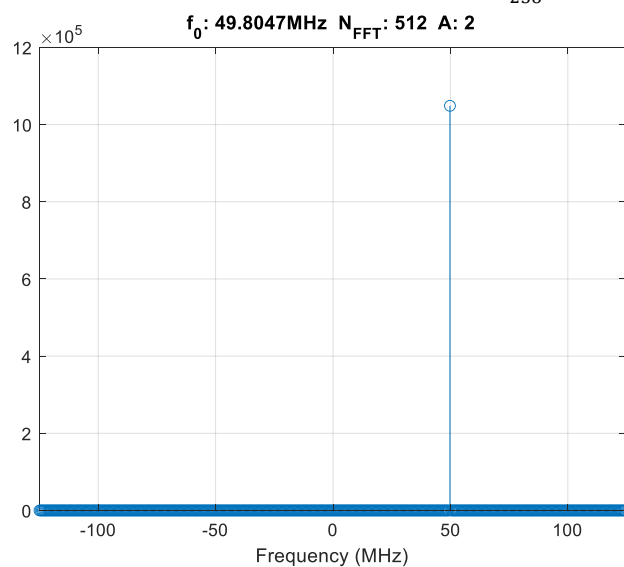
أ. $f_0 = \frac{250 \times 51}{256} \text{ MHz}$, $A = 2$ و تعداد نقاط FFT برابر ۲۵۶ نقطه است.



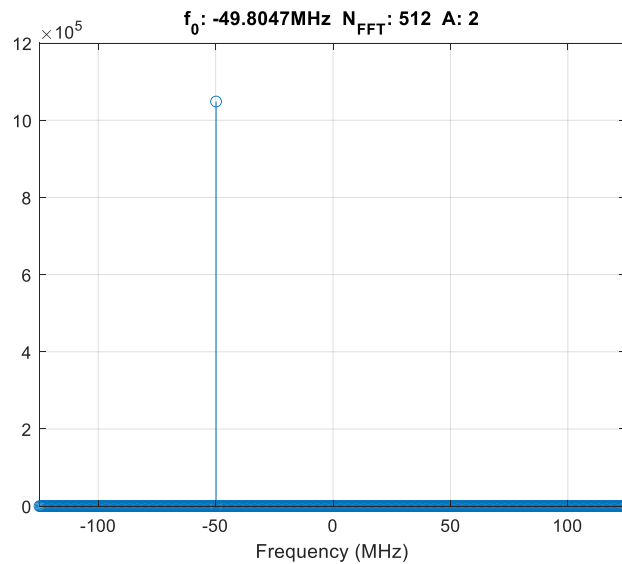
ب. $f_0 = \frac{250 \times 51.5}{256} \text{MHz}$, $A = 2$ و تعداد نقاط FFT برابر ۲۵۶ نقطه است.



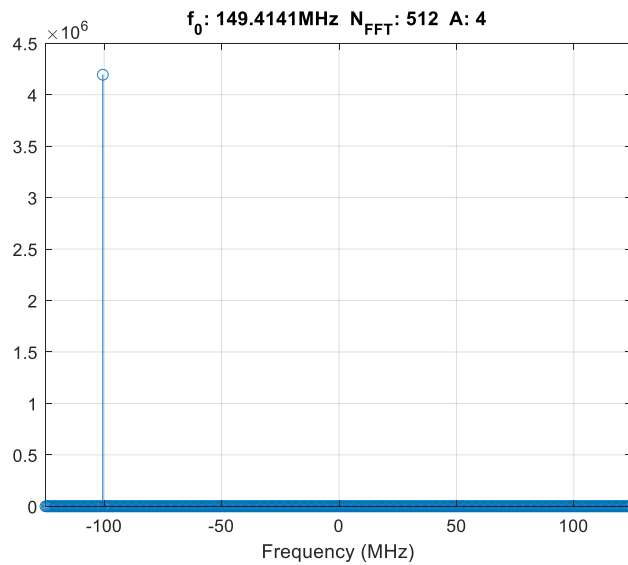
ت. $f_0 = \frac{250 \times 51}{256} \text{MHz}$, $A = 2$ و تعداد نقاط FFT برابر ۵۱۲ نقطه است.



ث. $f_0 = -\frac{250 \times 51}{256} \text{MHz}$, $A = 2$ و تعداد نقاط FFT برابر ۵۱۲ نقطه است.



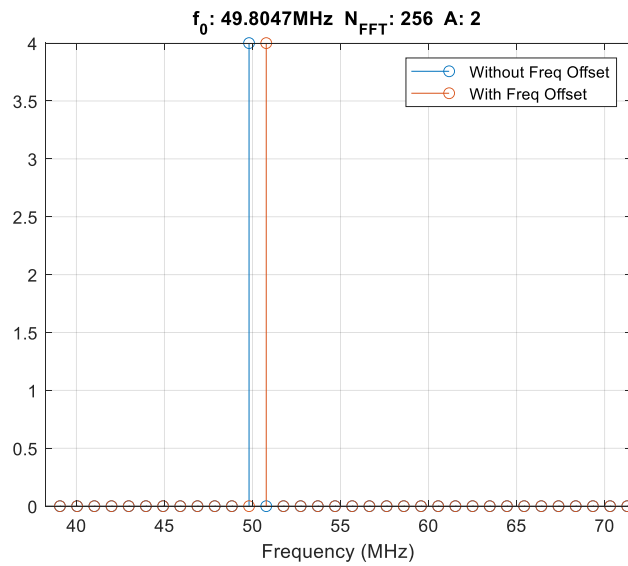
ج. $f_0 = \frac{250 \times 153}{256} \text{ MHz}$, $A = 4$ و تعداد نقاط FFT برابر ۵۱۲ نقطه است.



ح. نسبت طول زمانی یک فریم FFT ($N_{\text{FFT}} T_s$) به دوره تناوب سیگنال ($1/f_0$) برای بخش های قبل

```
ratio1 = 51
ratio2 = 51.5
ratio3 = 102
ratio4 = -102
ratio5 = 306
```

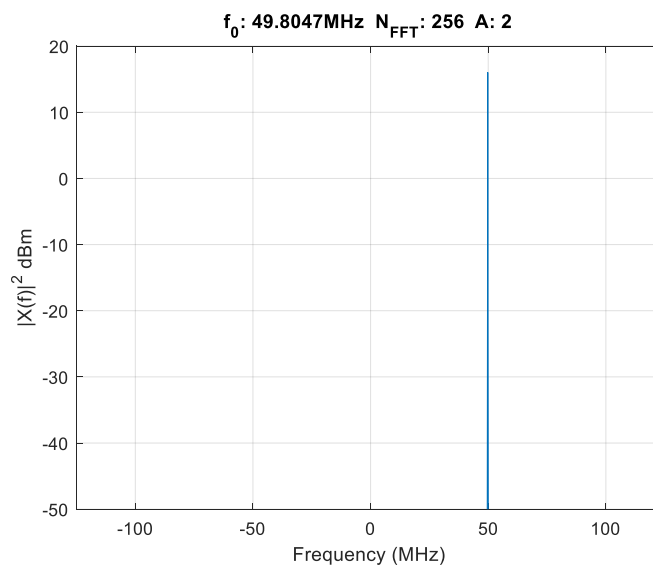
خ. در این جا بخش الف در نظر گرفته شده است. فرکانس بخش الف اگر به اندازه ی رزولوشن جابه جا شود بر روی خانه ی FFT بعد می افتد و هیچ نشتی نیز وجود ندارد. نمودار بزرگ نمایی شده ی آن در زیر آمده است. دقت نمایید ضریبی که در بخش خ نیز خواسته شده یود در این جا اعمال شده است.



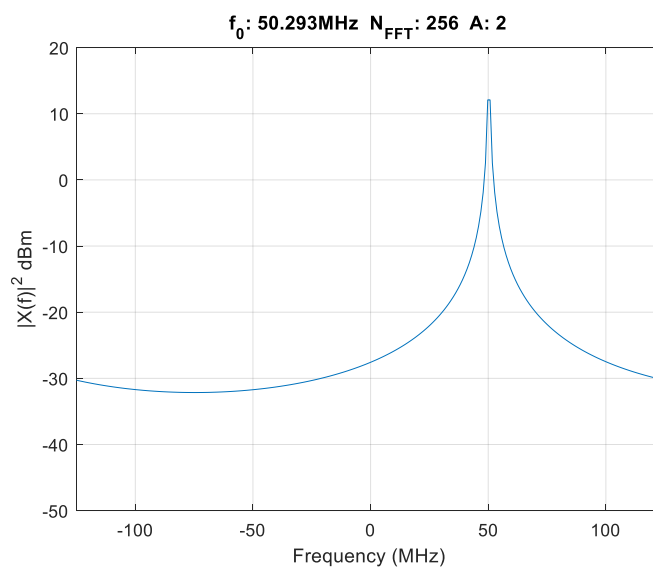
د. توان

```
time_power = 4
freq_power = 4
```

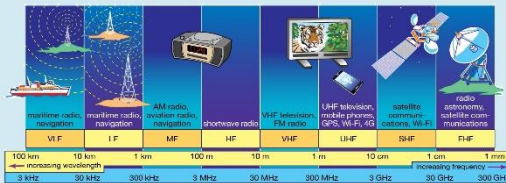
د. نمودار مربوط به بخش الف



نمودار مربوط به بخش ب







آزمایش سوم

رادیونرم افزار و پایش سیگنال های رادیویی

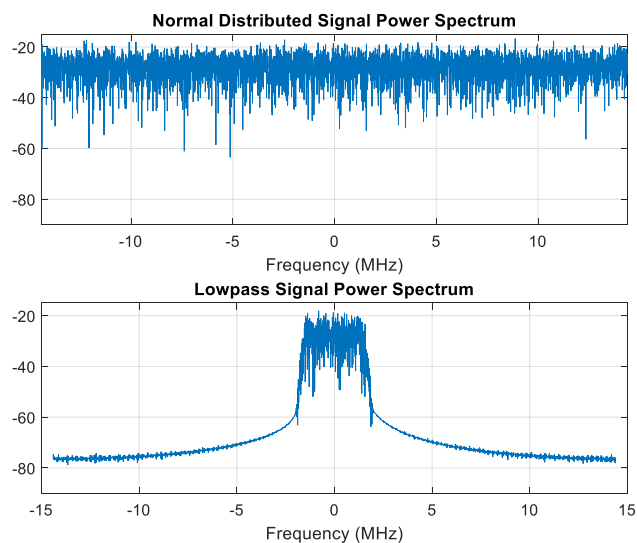
شرح آزمایش



آزمایش ۱-۳: معادل پایین گذر و میان گذر سیگنال

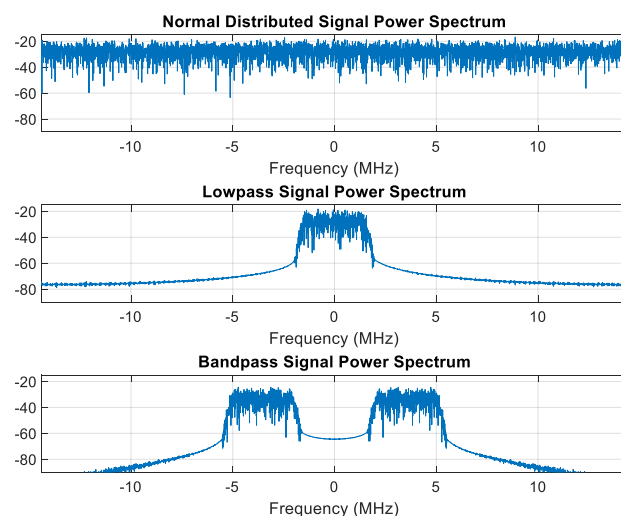
۱. تولید فرآیند میان گذر و پایین گذر:

أ. تولید فرآیند پایین گذر:



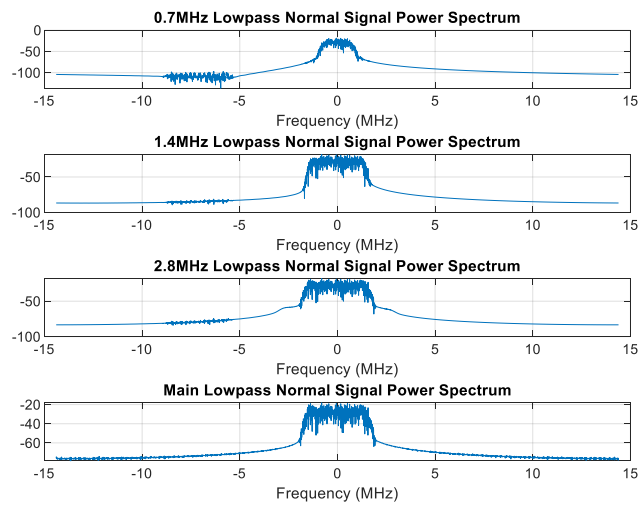
```
x_power = 1.0122
x_baseband_power = 0.1139
bw_noise = 3.3081e6
```

ب. تولید فرآیند میان گذر معادل یک فرآیند پایین گذر:

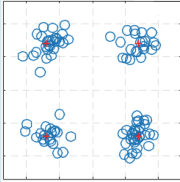


```
x_power = 1.0122
x_baseband_power = 0.1139
x_bandpass_power = 0.0570
```


ت. تولید فرآیند پایین گذر از یک سیگنال میان گذر:



```
x_baseband_power = 0.1139
x_baseband_0p7_power = 0.0455
x_baseband_1p4_power = 0.0954
x_baseband_1p4_power = 0.0954
```



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 آن جا 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_symbol**)، 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_symbol	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 $\text{pkt_size} \times k$ (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 **sym_idx** بردار **sym_idx**، شماره‌ی سطری از **b_gray** را نشان می‌دهد که برابر با سطر **sym_idx** 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_symbol`)، 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_symbol, pulse_name, pulse_shape_mode, varargin)
pulse_shape_mode
```

توجه داشته باشید که گام ۳ سوال قبل را مجدد در قالب یک تابع منسجم انجام داده‌ایم. در این جا `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_symbol, 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 `rx_sym` بردار سمبل‌های آشکارشده‌ی `det_sym`، نزدیکترین مقدار `constellation` به المان `rx_sym` خواهد بود. سطر اول این تابع به صورت زیر می‌باشد.

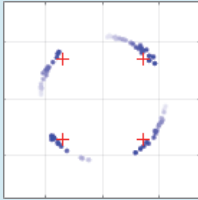
```
function [det_sym] = min_dist_detector(rx_sym, constellation);
```

۳. دمدولاسیون یا آشکارسازی پالس: نمونه‌های ارسال شده (`tx_smpl`) را به عنوان نمونه‌های دریافتی درون بردار `rx_smpl` قرار دهید. در این جا 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_symbol`)، 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_symbol, pulse_name, rx_mode, varargin)
```



- [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.



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 (f_s)، ضریب افزایش نرخ نمونه‌برداری سمبل‌ها یا تعداد نمونه‌های هر سمبل (smp1_per_syml)، 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 ۱ می‌باشد.

جدول ۲ پارامترهای آزمایش ۵

پارامتر	f_s	smp1_per_syml	pkt_size	stop_time
مقدار	10MHz	۸	شبیه‌سازی: ۱۰۰۰۰۰ 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; % Truncation 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 Transmitter Center Frequency (AD9363: 325-
3800MHz) (AD9364: 70-6000MHz)
tx_gain = 0; % Set Transmitter Attenuation As a Negative Gain
(-89.75 to 0 dB)
tx_address = 'usb:0'; % Set Transmitter 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 $\text{pkt_size} \times k$ 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 سطرهاى ماتریس `b_tx` و ماتریس `b_gray` برقرار کنید. به عبارتی سطر `i`ام بردار `sym_idx`، شماره‌ی سطر `i` از `b_gray` را نشان می‌دهد که برابر با سطر `i`ام ماتریس `b_tx` باشد. (راهنمایی: بهتر است سطرهاى ماتریس `b_gray` را به یک عدد صحیح تبدیل نمایید.)

گام ۲. کدگذاری طبیعی بیت‌های تولیدی: با استفاده از فلگ `flg_gray_encode`، قابلیتى را به برنامه‌ی خود اضافه نمایید که بتوان کدگذاری گری را اعمال نمود.

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 فعلاً مثلثی در نظر بگیرید.

آزمایش ۳-۵: مدل‌سازی کانال

- 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 این جا فعلاً مقدار تأخیر برابر با صفر قرار دهید.
- اعمال اختلاف فاز در کانال: برای اعمال اختلاف فاز در کانال می‌بایست بردار `tx_smpl_delayed` را در $\exp(1i \cdot \text{chnl_phase_offset})$ ضرب شود. حاصل را در بردار `rx_smpl` قرار دهید. در این جا نیز آفست فاز را برابر با صفر قرار دهید.

۳. شبیه‌سازی کانال با نویز سفید گاوسی

- 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.
- 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.

- 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 آشکار شده (`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 کدگذاری 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 کدگذاری گری و کدگذاری باینری را نیز بر روی نمودار نرخ خطای بیت نشان دهید.

۲. اثر تأخیر در کانال:

در این جا 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_smp1` جدا کرده و در متغیر `hdr_smp1` ذخیره نمایید. بیت‌های هدر می‌بایست به صورت زیر باشد. این رشته بیت خواص خودهمبستگی خوبی دارد.

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

گام ۳. تخمین زمان شروع بسته و جبران اثر تأخیر کانال: با استفاده از همبستگی بردار `rx_smp1_noise` و نمونه‌های زمانی هدر `hdr_smp1`، 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 این جا 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_smp1_noise` و نمونه‌های زمانی هدر `hdr_smp1`، 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_smp1** 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 ذخیره می‌نماید. فرستنده داده‌ها را از حافظه‌ی رادیو نرم‌افزار قرائت 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 در نظر بگیرید. به منظور دریافت داده، شیء گیرنده را فراخوانی نمایید و حاصل را درون متغیر **rx_smp1** قرار دهید.

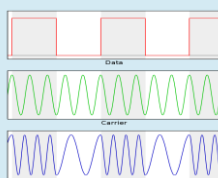
۳. اثر شکل پالس بر روی پهنای باند سیگنال ارسالی: طیف فرکانسی سیگنال **rx_smp1** دریافتی از ADALM-PLUTO را برای مدولاسیون 4PSK و به ازای شکل پالس‌های مثلثی و root raised cosine رسم نمایید. در مورد پهنای باند و سطح باندهای جانبی آن توضیحاتی ارائه دهید. برای این منظور از برنامه‌های قبلی استفاده نمایید.

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.



- [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.



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 (f_s)، تعداد نمونه‌های هر سمبل ($smpl_per_syml$)، 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 ۳ می‌باشد.

جدول ۳ پارامترهای آزمایش ۶

پارامتر	f_s	$smpl_per_syml$	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` می‌بایست مدولاسیون FSK را نیز بشناسد. این فایل می‌بایست از برنامه‌ی اصلی فراخوانی نماید.

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 $pkt_size \times k$ 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 کدگذاری 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_smp1** 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 باند میانی تولید نمی‌شود نیازی به دانستن f_0 نیست. از این رو می‌بایست از فرمول معادل باندپایه‌ی 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 $t = (0:smp1_per_symbol-1)*ts$ ایجاد نمود که در آن $ts = 1/f_s$ 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 **smp1_per_symbol** برابر با باشد، بردار **tx_smp1** 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_smp1** قرار داد.

آزمایش ۳-۶: مدل‌سازی کانال

۱. افزودن تأخیر در کانال: برای شبیه‌سازی تأخیر در کانال به اندازه‌ی پارامتر **chnl_delay_in_smp1** به ابتدای بردار **tx_smp1** صفر اضافه نمایید. حاصل را درون بردار **tx_smp1_delayed** قرار دهید. دقت نمایید که طول بردار افزایش می‌یابد و می‌بایست در ادامه‌ی برنامه اثر این افزایش لحاظ شود. در این جا فعلاً مقدار تأخیر برابر با صفر قرار دهید.

۲. اعمال اختلاف فاز در کانال: برای اعمال اختلاف فاز در کانال می‌بایست بردار **tx_smp1_delayed** را در $\exp(1i*chnl_phase_offset)$ ضرب شود. حاصل را در بردار **rx_smp1** قرار دهید. در این جا 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_smp1** تولید نمایید و آن را **noise_smp1** نامگذاری نمایید. سپس این بردار را با بردار **rx_smp1** جمع نموده و حاصل را **rx_smp1_noise** بنامید.

آزمایش ۴-۶: پیاده‌سازی گیرنده‌ی مدولاسیون FSK

در این جا می‌خواهیم گیرنده‌ی مدولاسیون FSK خطی را پیاده‌سازی نماییم.

۱. دمدولاسیون یا آشکارسازی پالس: نمونه‌های ارسال شده به همراه نویز (**tx_smp1_noise**) را به عنوان نمونه‌های دریافتی درون بردار **rx_smp1** قرار دهید. با استفاده از تابع **pulse_demodulation** با ورودی‌های نمونه‌های سیگنال دریافتی (**rx_smp1**)، نام مدولاسیون (**modulation**)، مرتبه‌ی مدولاسیون (**M**)، نرخ نمونه‌برداری (**fs**)، تعداد نمونه‌های هر سمبل (**smp1_per_symbol**)، 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 آشکار شده (**det_sym_idx**) را به دست آورید. (در این جا به پارامترهای ورودی **mode**, **pulse_name** 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 این جا 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_smp1 جدا کرده و در متغیر hdr_smp1 ذخیره نمایید. بیت‌های هدر می‌بایست به صورت زیر باشد. این رشته بیت خواص خودهمبستگی خوبی دارد.

$hdr_bit = repmat(de2bi(hex2dec('1C6387FF5DA4FA325C895958DC5'))', 1, k);$
گام ۳. تخمین زمان شروع بسته و جبران اثر تأخیر کانال: با استفاده از همبستگی بردار tx_smp1_noise و نمونه‌های زمانی هدر hdr_smp1 ، 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 این جا 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_smp1_noise و نمونه‌های زمانی هدر hdr_smp1 ، 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 $rx_alg = 1$ و $flg_hrdwr_usg = 1$ می‌باشد.

۱. ارسال نمونه‌ها با استفاده از ADALM-PLUTO: بردار `tx_smp1` 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 ذخیره می‌نماید. فرستنده داده‌ها را از حافظه‌ی رادیو نرم‌افزار قرائت 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 در نظر بگیرید. به منظور دریافت داده، شیء گیرنده را فراخوانی نمایید و حاصل را درون متغیر `rx_smp1` قرار دهید.

۳. طیف سیگنال ارسالی: طیف فرکانسی سیگنال `rx_smp1` دریافتی از ADALM-PLUTO را برای مدولاسیون 2FSK را رسم نماید. در مورد پهنای باند و سطح باندهای جانبی آن توضیحاتی ارائه 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 به لحظه به روز 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;
```

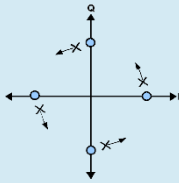
۵. جبران‌سازی دامنه، فاز و تأخیر و مشاهده‌ی منظومه‌ی سیگنالی: با یافتن ابتدای بسته‌ها و جبران فاز، دامنه و تأخیر، منظومه‌ی سیگنالی مربوط به `rx_sym` را رسم نمایید. احتمال خطای بیت را نیز در این حالت محاسبه نمایید.

۶. اثر Δf بر روی پهنای باند ارسالی: طیف فرکانسی سیگنال `rx_smp1` دریافتی از ADALM-PLUTO را برای مدولاسیون 4FSK رسم نماید. در مورد پهنای باند و سطح باندهای جانبی آن توضیحاتی ارائه دهید. با تغییر Δf از $\frac{1}{2T}$ به $\frac{2}{T}$ پهنای باند سیگنال دریافتی هر حالت را به دست آورده و با یکدیگر مقایسه نمایید.



- [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.





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 آن جا که در این آزمایش بناست این فرستنده با استفاده از رادیو نرم افزار 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 (f_s)، تعداد نمونه‌های هر سمبل (smp1_per_syml)، 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	f_s	smp1_per_syml	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.

- 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` می‌بایست 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 نماید.

آزمایش ۲-۷: پیاده‌سازی مدولاسیون FSK ناهمدوس

- شبیه‌سازی کانال با فاز تصادفی: برای شبیه‌سازی کانال با فاز تصادفی می‌بایست بردار `tx_smp1_delayed` را در `exp(1i*chnl_phase_offset_amb)` ضرب شود. حاصل را در بردار `tx_smp1` قرار دهید. `chnl_phase_offset_amb` را به صورت یک بردار تصادفی بین ۰ و 2π تولید کنید. طول این بردار برابر با طول بردار `tx_smp1` می‌باشد. دقت کنید که فاز اضافه شده در طول هر سمبل ثابت باشد. در این حالت `phase_amb_opt` برابر با ۰ 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 $\arg \max |r_{\ell} \cdot \underline{S}_{\ell,m}|$ می‌باشد.

۳. پیاده‌سازی سخت‌افزاری مدولاسیون FSK ناهمدوس: با فرض این که 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 این جا 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 در راون بردار `enc_b_tx` قرار دهید.

۳. نگاشت بیت به سمبل:

گام ۱. کدگذاری گری بیت‌های تولیدی: با وجود این که در این جا 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 `sym_idx` بردار `sym_idx`، شماره‌ی سطری از `b_gray` را نشان می‌دهد که برابر با سطر `sym_idx` 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_smp1` ذخیره نمایید. تابع شکل‌دهی پالس را مثلی در نظر بگیرید.

آزمایش ۴-۷: مدل‌سازی کانال

۱. افزودن تأخیر در کانال: برای شبیه‌سازی تأخیر در کانال به اندازه‌ی پارامتر `chnl_delay_in_smp1` به ابتدای بردار `tx_smp1` صفر اضافه نمایید. حاصل را درون بردار `tx_smp1_delayed` قرار دهید. دقت نمایید که طول بردار افزایش می‌یابد و می‌بایست در ادامه‌ی برنامه اثر این افزایش لحاظ شود. در این جا فعلاً 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_smp1_delayed`، ابهام فاز ایجاد شده را به سیگنال ارسالی اضافه نمایید و حاصل را در `tx_smp1` ذخیره نمایید. در این جا `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_smp1` تولید نمایید و آن را `noise_smp1` نامگذاری نمایید. سپس این بردار را با بردار `tx_smp1` جمع نموده و حاصل را `tx_smp1_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_smp1_noise`) را به عنوان نمونه‌های دریافتی درون بردار `rx_smp1` قرار دهید. با استفاده از تابع `pulse_demodulation` با ورودی‌های نمونه‌های سیگنال دریافتی (`rx_smp1`)، نام مدولاسیون (`modulation`)، مرتبه‌ی مدولاسیون (`M`)، نرخ نمونه‌برداری (`fs`)، تعداد نمونه‌های هر سمبل (`smp1_per_symb1`)، نام تابع شکل دهنده‌ی 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 آشکار شده (`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 این که در این جا 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 مثلی 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_smp1` 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 ذخیره می‌نماید. فرستنده داده‌ها را از حافظه‌ی رادیو نرم‌افزار قرائت 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 در نظر بگیرید. به منظور دریافت داده، شیء گیرنده را فراخوانی نمایید و حاصل را درون متغیر `rx_smp1` 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.



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





آزمایش هشتم

انتقال دیجیتال از درون کانال باندمحدود AWGN

شرح آزمایش



آزمایش ۱-۸: مقداردهی‌های اولیه

در این آزمایش بنا داریم تعدادی بسته‌ی داده را برای ارسال از طریق مدولاسیون 2PAM آماده نماییم. از آن جا 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 ($smpl_per_syml$)، 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	$smpl_per_syml$	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 ۲-۸: 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 $span_in_syml = 6$ 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 ۳-۸: مدل‌سازی کانال باندمباریک

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

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

آزمایش ۵-۸: خواسته‌های کلی

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



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