به نام خدا

دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده مهندسی برق و کامپیوتر

اصول سیستم‌های مخابراتی

پاییز 1400

استاد: دکتر مریم صباغیان

تمرین کامپیوتری شماره 3

محمدمهدی عبدالحسینی

810 198 434

Communication Systems

فهرست مطالب

بخش اول : فرستنده (Transmitter)

مدولاسیون فاز (Phase Modulation)

Part 1: pm()

در مدولاسیون فاز (PM) ، مطابق رابطه 1.1 سیگنال پیام در فاز یک سیگنال حامل کسینوسی قرار میگیرد.

$$\text{Phase Modulation}: \quad x_c(t) = A_c \cos\left(2\pi f_c t + \phi_\Delta x(t)\right) \qquad 1.1$$

با توجه به رابطه 1.1 ، تابع pm() را در متلب پیاده‌سازی میکنیم.

```
function Xc = pm(x, Ac, phi_delta, fc, t)
    n = max(abs(x));
    phi_t = phi_delta * x/n;
    Xc = Ac*cos(2*pi*fc*t + phi_t);
end
```

Part 2: nbpm()

در مدولاسیون زاویه باند باریک (NarrowBand PM and FM) ، با فرض آنکه $|\phi(t)| \ll 1$ ، میتوان خروجی مدولاسیون را با تقریب بدست آورد. (رابطه 1.2)

$$x_c(t) = A_c \cos\left[W_c t + \phi(t)\right] = A_c \cos\phi(t) \cos W_c t - A_c \sin\phi(t) \sin W_c t$$

$$= A_c \left[1 - \frac{1}{2!}\phi^2(t) + \cdots\right] \cos W_c t - A_c \left[\phi(t) - \frac{1}{3!}\phi^3(t) + \cdots\right] \sin W_c t$$

$$\text{if } |\phi(t)| \ll 1 \Rightarrow x_c(t) \approx A_c \cos W_c t - A_c \phi(t) \sin W_c t \qquad 1.2$$

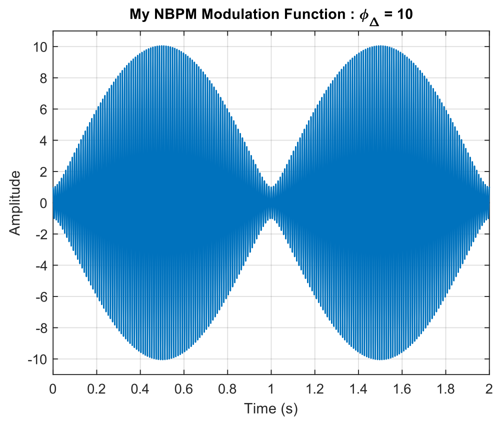
با توجه به رابطه 1.2 ، تابع nbpm() را در متلب پیاده‌سازی میکنیم.

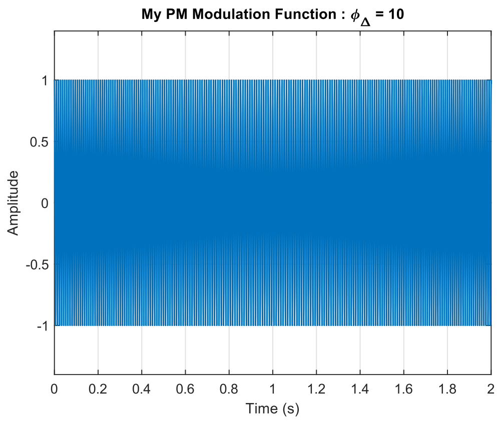
```
function Xc = nbpm(x, Ac, phi_delta, fc, t)
    n = max(abs(x));
    phi_t = phi_delta * x/n;
    Xc = Ac*cos(2*pi*fc*t) - Ac*phi_t.*sin(2*pi*fc*t);
end
```

Part 3:

مقادیر زیر را به عنوان ورودی، به دو تابع بالا میدهیم. خروجی‌ها را ترسیم میکنیم و در ادامه خطای روش NBPM را با استفاده از تابع immse() در متلب بدست می‌آوریم.

$$m(t) = \sin(\pi t) \;;\; t \in [0, 10] \;;\; f_s = 10\,kHz \;;\; A_c = 1 \;;\; f_c = 100\,Hz \;;\; \phi_\Delta = 10$$





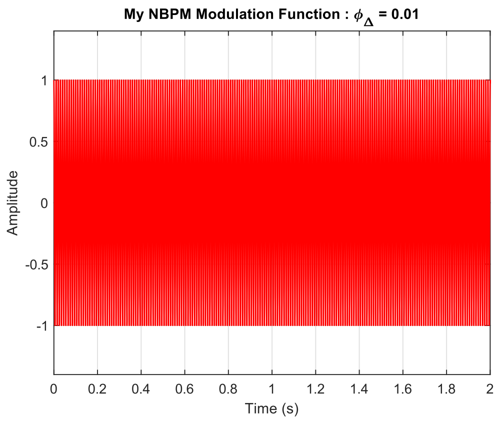
همانطور که انتظار داشتیم، با توجه به اینکه $|\phi(t)| \gg 1$ ، بنابراین تقریب NBPM خطای زیادی خواهد داشت.

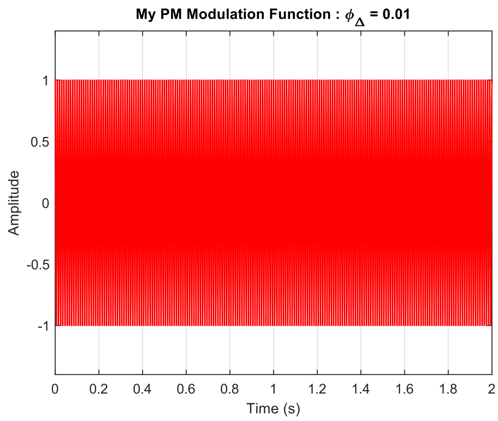
```
fs = 10000; % 10kHz
ts = 1 / fs;
t = 0 : ts : 10 - ts;
mt = sin(pi*t);
Ac = 1;
fc = 100; % 100Hz
phi_delta = 10;
mt_pm = pm(mt, Ac, phi_delta, fc, t);
plot(t, mt_pm, 'linewidth', 1);
title('My PM Modulation Function : \phi_\Delta = 10');
xlabel('Time (s)');
ylabel('Amplitude');
set(gca,'xtick', 0 : 0.2 : 10, 'xlim', [0,1]);
set(gca,'ytick', -2 : 0.5 : 2, 'ylim', [-1.4,1.4]);
grid on;
mt_nbpm = nbpm(mt, Ac, phi_delta, fc, t);
plot(t, mt_nbpm, 'linewidth', 1);
title('My NBPM Modulation Function : \phi_\Delta = 10');
xlabel('Time (s)');
ylabel('Amplitude');
set(gca,'xtick', 0 : 0.2 : 10, 'xlim', [0,1]);
set(gca,'ytick', -10 : 2 : 10, 'ylim', [-11,11]);
grid on;
error_NBPM = immse(mt_pm,mt_nbpm)
```

```
error_NBPM = 25.8112
```

Part 4:

قسمت 3 را به ازای $\phi_\Delta = 0.01$ تکرار میکنیم. بدلیل آنکه $|\phi(t)| \ll 1$ ، انتظار داریم تقریبِ روش NBPM ، خطای قابل قبولی داشته باشد.





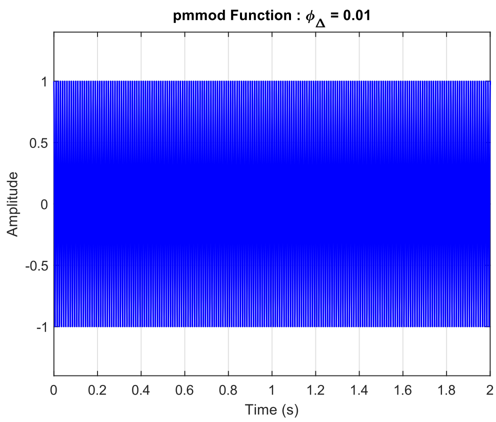

```
phi_delta = 0.01;
mt_pm = pm(mt, Ac, phi_delta, fc, t);
plot(t, mt_pm, 'r', 'linewidth', 1);
title('My PM Modulation Function : \phi_\Delta = 0.01');
xlabel('Time (s)');
ylabel('Amplitude');
set(gca,'xtick', 0 : 0.2 : 10, 'xlim', [0,2]);
set(gca,'ytick', -2 : 0.5 : 2, 'ylim', [-1.4,1.4]);
grid on;
mt_nbpm = nbpm(mt, Ac, phi_delta, fc, t);
plot(t, mt_nbpm, 'r', 'linewidth', 1);
title('My NBPM Modulation Function : \phi_\Delta = 0.01');
xlabel('Time (s)');
ylabel('Amplitude');
set(gca,'xtick', 0 : 0.2 : 10, 'xlim', [0,2]);
set(gca,'ytick', -2 : 0.5 : 2, 'ylim', [-1.4,1.4]);
grid on;
error_NBPM = immse(mt_pm,mt_nbpm)
```

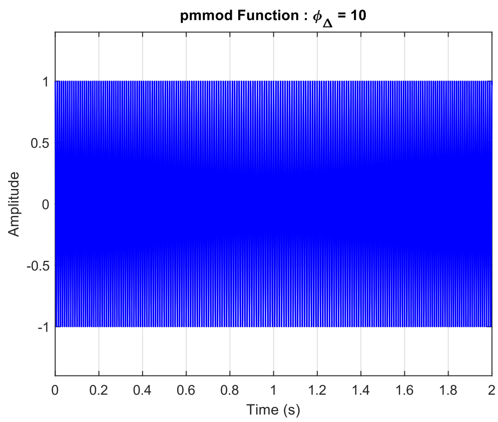
```
error_NBPM = 4.6875e-10
```

همانطور که انتظار داشتیم، با توجه به اینکه $1 \gg |\phi(t)|$ ، بنابراین تقریب NBPM خطای بسیار کمی دارد.

Part 5:6:

قسمت 3,4 را این بار با استفاده از تابع `pmmod()` در متلب، تکرار میکنیم.





```
phi_delta = 10;
mt_pmmod = pmmod(mt, fc, fs, phi_delta);
plot(t, mt_pmmod, 'b', 'linewidth', 1);
title('pmmod Function : \phi_\Delta = 10');
xlabel('Time (s)');
ylabel('Amplitude');
set(gca,'xtick', 0 : 0.2 : 10, 'xlim', [0,2]);
set(gca,'ytick', -2 : 0.5 : 2, 'ylim', [-1.4,1.4]);
grid on;
mt_pm = pm(mt, Ac, phi_delta, fc, t);
error_pmmod_pm = immse(mt_pmmod, mt_pm)

    error_pmmod_pm = 0

% ===================================================================
phi_delta = 0.01;
mt_pmmod = pmmod(mt, fc, fs, phi_delta);
plot(t, mt_pmmod, 'b', 'linewidth', 1);
title('pmmod Function : \phi_\Delta = 0.01');
xlabel('Time (s)');
ylabel('Amplitude');
set(gca,'xtick', 0 : 0.2 : 10, 'xlim', [0,2]);
set(gca,'ytick', -2 : 0.5 : 2, 'ylim', [-1.4,1.4]);
grid on;
mt_pm = pm(mt, Ac, phi_delta, fc, t);
error_pmmod_pm = immse(mt_pmmod, mt_pm)

    error_pmmod_pm = 0
```

با محاسبه تفاوت خروجی توابع `pm()` و `pmmod()` ، مشخص شد که خروجی دو تابع دقیقاً یکسان است.

مدولاسیون فرکانس (Frequency Modulation)

Part 7: fm()

در مدولاسیون فرکانس (FM) ، مطابق رابطه 1.3 ، سیگنال پیام در سیگنال حامل کسینوسی قرار میگیرد.

$$\text{Frequency Modulation :} \quad x_c(t) = A_c \cos\left[\omega_c t + 2\pi f_\Delta \int x(\lambda)\, d\lambda\right] \qquad 1.3$$

با توجه به رابطه 1.3 ، تابع `fm()` را در متلب پیاده‌سازی میکنیم.

```
function Xc = fm(x, Ac, f_delta, fc, t)
    n = max(abs(x));
    phi_t = 2 * pi * f_delta * cumtrapz(t, x/n);
    Xc = Ac*cos(2*pi*fc*t + phi_t);
end
```

Part 7: nbfm()

مطابق آنچه در قسمت 2 دیدیم، در مدولاسیون زاویه باند باریک (NarrowBand PM and FM) ، با فرض آنکه $|\phi(t)| \ll 1$ ، میتوان خروجی مدولاسیون را با تقریب بدست آورد. (رابطه 1.2)

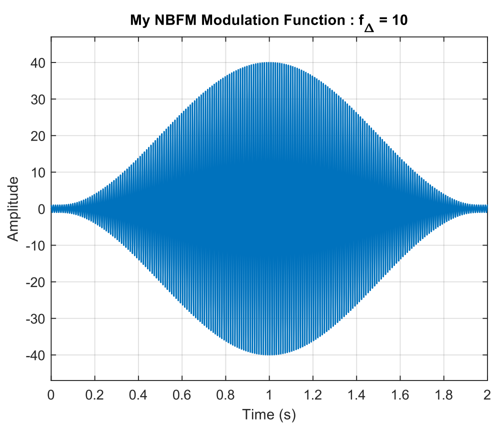
با توجه به رابطه 1.2 ، تابع nbfm() را در متلب پیاده‌سازی میکنیم.

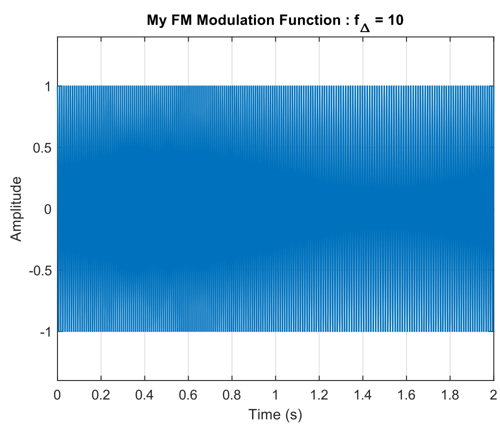
```
function Xc = nbfm(x, Ac, f_delta, fc, t)
    n = max(abs(x));
    phi_t = 2 * pi * f_delta * cumtrapz(t, x/n);
    Xc = Ac*cos(2*pi*fc*t) - Ac*phi_t.*sin(2*pi*fc*t);
end
```

Part 8:

مقادیر را مشابه با آنچه در قسمت 3 داشتیم ($f_\Delta = 10$) ، به عنوان ورودی، به دو تابع بالا میدهیم. خروجی‌ها را ترسیم میکنیم و در ادامه، خطای روش NBFM را با استفاده از تابع immse() در متلب بدست می‌آوریم.

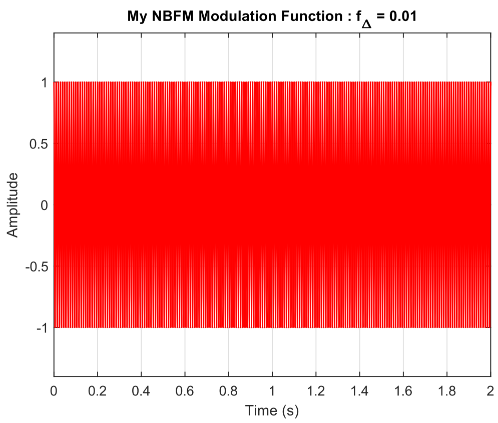
```
f_delta = 10;
mt_fm = fm(mt, Ac, f_delta, fc, t);
plot(t, mt_fm, 'linewidth', 1);
title('My FM Modulation Function : f_\Delta = 10');
xlabel('Time (s)');
ylabel('Amplitude');
set(gca,'xtick', 0 : 0.2 : 10, 'xlim', [0,2]);
set(gca,'ytick', -2 : 0.5 : 2, 'ylim', [-1.4,1.4]);
grid on;
mt_nbfm = nbfm(mt, Ac, f_delta, fc, t);
plot(t, mt_nbfm, 'linewidth', 1);
title('My NBFM Modulation Function : f_\Delta = 10');
xlabel('Time (s)');
ylabel('Amplitude');
set(gca,'xtick', 0 : 0.2 : 10, 'xlim', [0,2]);
set(gca,'ytick', -50 : 10 : 50, 'ylim', [-47,47]);
grid on;
error_NBFM = immse(mt_fm,mt_nbfm)

    error_NBFM = 297.3367
```

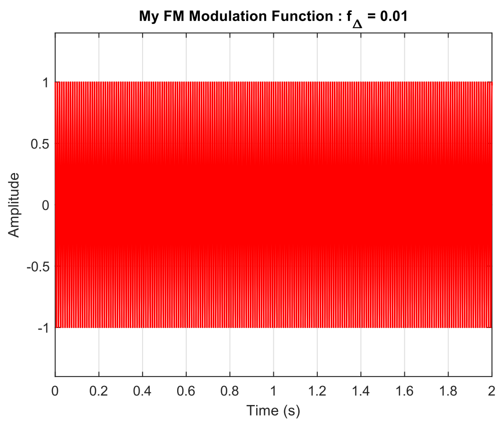




Part 9:

قسمت 8 را به ازای $f_{\Delta} = 0.01$ تکرار میکنیم. بدلیل آنکه $|\phi(t)| \ll 1$ ، انتظار داریم تقریبِ روش NBFM ، خطای قابل قبولی داشته باشد.



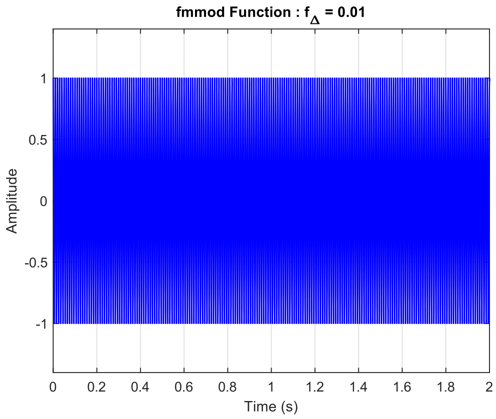


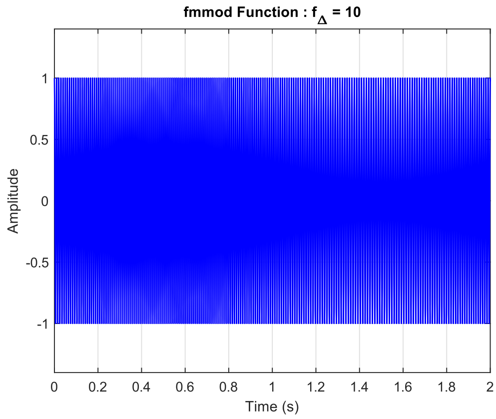

```
f_delta = 0.01;
mt_fm = fm(mt, Ac, f_delta, fc, t);
plot(t, mt_fm, 'r', 'linewidth', 1);
title('My FM Modulation Function : f_\Delta = 0.01');
xlabel('Time (s)');
ylabel('Amplitude');
set(gca,'xtick', 0 : 0.2 : 10, 'xlim', [0,2]);
set(gca,'ytick', -2 : 0.5 : 2, 'ylim', [-1.4,1.4]);
grid on;
mt_nbfm = nbfm(mt, Ac, f_delta, fc, t);
plot(t, mt_nbfm, 'r', 'linewidth', 1);
title('My NBFM Modulation Function : f_\Delta = 0.01');
xlabel('Time (s)');
ylabel('Amplitude');
set(gca,'xtick', 0 : 0.2 : 10, 'xlim', [0,2]);
set(gca,'ytick', -2 : 0.5 : 2, 'ylim', [-1.4,1.4]);
grid on;
error_NBFM = immse(mt_fm,mt_nbfm)

    error_NBFM = 8.7494e-08
```

همانطور که انتظار داشتیم، با توجه به اینکه $|\phi(t)| \ll 1$ ، بنابراین تقریب NBFM خطای بسیار کمی دارد.

Part 10:11:

قسمت 8,9 را این بار با استفاده از تابع `fmmod()` در متلب، تکرار میکنیم.

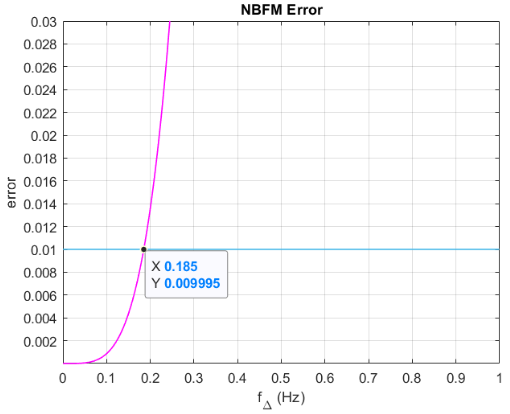


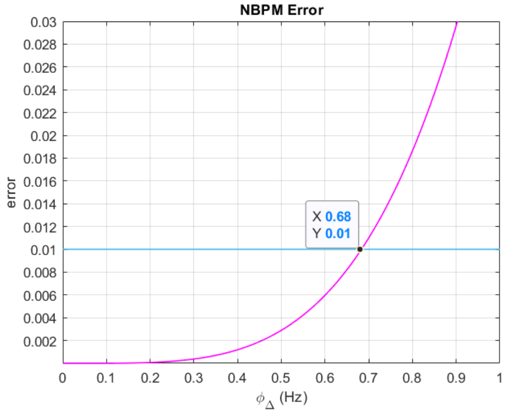



```
f_delta = 10;
mt_fmmod = fmmod(mt, fc, fs, f_delta);
plot(t, mt_fmmod, 'r', 'linewidth', 1);
title('fmmod Function : f_\Delta = 10');
xlabel('Time (s)');
ylabel('Amplitude');
set(gca,'xtick', 0 : 0.2 : 10, 'xlim', [0,2]);
set(gca,'ytick', -2 : 0.5 : 2, 'ylim', [-1.4,1.4]);
grid on;
mt_fm = pm(mt, Ac, f_delta, fc, t);
error_fmmod_fm = immse(mt_fmmod, mt_fm)

        error_fmmod_fm = 2.4674e-06
% ==========================================================================
f_delta = 0.01;
mt_fmmod = fmmod(mt, fc, fs, f_delta);
plot(t, mt_fmmod, 'r', 'linewidth', 1);
title('fmmod Function : f_\Delta = 0.01');
xlabel('Time (s)');
ylabel('Amplitude');
set(gca,'xtick', 0 : 0.2 : 10, 'xlim', [0,2]);
set(gca,'ytick', -2 : 0.5 : 2, 'ylim', [-1.4,1.4]);
grid on;
mt_fm = pm(mt, Ac, f_delta, fc, t);
error_fmmod_fm = immse(mt_fmmod, mt_fm)

        error_fmmod_fm = 2.4674e-12
```

Part 12:

در این قسمت میخواهیم خطا را برای دو مدولاسیون عادی و باند باریک به ازای ϕ_Δ و f_Δ مختلف بدست آوریم تا به بیشینه ϕ_Δ و f_Δ برای اینکه حداکثر خطا 1 درصد باشد، برسیم.





بنابراین بیشینه ϕ_Δ و f_Δ برای اینکه حداکثر خطا 0.01 باشد، به ازای $\phi_\Delta = 0.68$ و $f_\Delta = 0.185$ خواهد بود.

```
delta = 0 : 0.01 : 10 - 0.01;
error_NBPM = [];
error_NBFM = [];
for idelta = delta
    mti_pm = pm(mt, Ac, idelta, fc, t);
    mti_nbpm = nbpm(mt, Ac, idelta, fc, t);
    error_NBPM = [error_NBPM, immse(mti_pm, mti_nbpm)];
    mti_fm = fm(mt, Ac, idelta, fc, t);
    mti_nbfm = nbfm(mt, Ac, idelta, fc, t);
    error_NBFM = [error_NBFM, immse(mti_fm, mti_nbfm)];
end
plot(delta, 0.01*ones(length(delta)), delta, error_NBPM, 'm', 'linewidth', 1);
title('NBPM Error');
xlabel('\phi_\Delta (Hz)');
ylabel('error');
set(gca,'xtick', 0 : 0.1 : 1 , 'xlim', [0, 1]);
set(gca,'ytick', 0.002 : 0.002 : 0.03 , 'ylim', [0,0.03]);
grid on;
plot(delta, 0.01*ones(length(delta)), delta, error_NBFM, 'm', 'linewidth', 1);
title('NBFM Error');
xlabel('f_\Delta (Hz)');
ylabel('error');
set(gca,'xtick', 0 : 1 : 10 , 'xlim', [0, 10]);
set(gca,'ytick', 0.002 : 0.002 : 0.03 , 'ylim', [0,0.03]);
grid on;
```

بخش دوم : گیرنده (Receiver)

دمدولاسیون فرکانس (Frequency DeModulation)

Part 1:

FM Signal : $x_c(t) = A_c \cos\left(2\pi f_c t + 2\pi f_\Delta \int x(\lambda)\,d\lambda\right) \longrightarrow \boxed{d/dt} \longrightarrow y_1(t)$

$y_1(t) = A_c\left[2\pi f_c + 2\pi f_\Delta x(t)\right]\cos\left(2\pi f_c t + 2\pi f_\Delta \int x(\lambda)\,d\lambda + \frac{\pi}{2}\right) \longrightarrow \boxed{\text{Envelop Detector}} \longrightarrow y_2(t)$

$y_2(t) = A_c 2\pi f_c + A_c 2\pi f_\Delta x(t) \longrightarrow \boxed{\text{Remove DC}} \longrightarrow y_3(t)$

$y_3(t) = 2\pi f_\Delta A_c x(t) = k\, x(t) \longrightarrow k = 2\pi f_\Delta A_c$

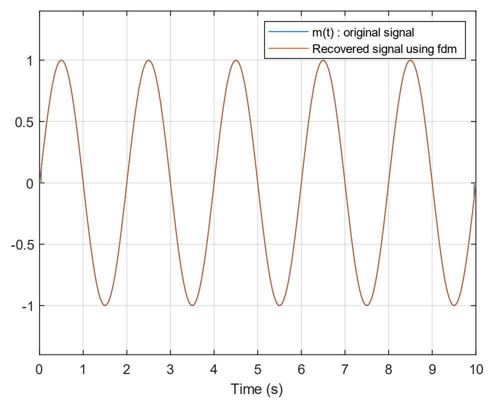
Part 2: fdm()

با توجه به نتایج قسمت 1 ، تابع fdm() را در متلب بصورت زیر پیاده‌سازی میکنیم.

```
function x = fdm(Xc, Ac, f_delta, fs)
    dXc = fs * gradient(Xc); % d/dt
    [EdXc,] = envelope(dXc,1000);   % Envelope Detector
    removeDC = EdXc - mean(EdXc);
    k = 2 * pi * f_delta * Ac;
    x = removeDC / k;

    % for smoother edges and error reduction
    x(1, 1:round(fs/25)) = 0;
    x(1, length(x) - round(fs/25) : length(x)) = 0;
end
```

Part 3:4:

در این قسمت میخواهیم خروجی تابع fm() در قسمت 8 از بخش فرستنده را به تابع fdm() بدهیم تا سیگنال پیام اولیه را بازیابی کنیم. سپس این کار را با استفاده از تابع fmdemod() در متلب تکرار میکنیم.



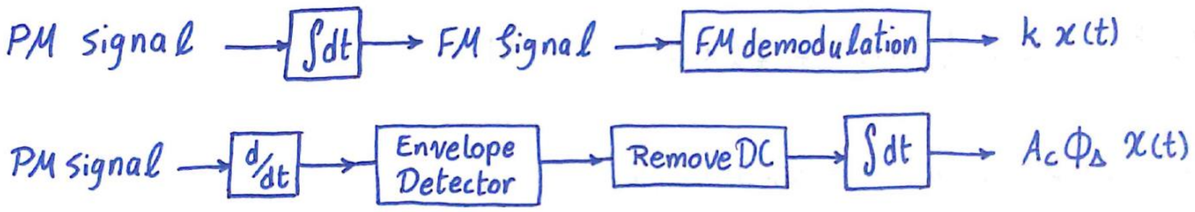

```
f_delta = 10;
```

```
mt_fm = fm(mt, Ac, f_delta, fc, t);
mt_fdm = fdm(mt_fm, Ac, f_delta, fs);
plot(t, mt, t, mt_fdm);
legend('m(t) : original signal', 'Recovered signal using fdm');
xlabel('Time (s)');
set(gca,'xtick', 0 : 1 : 10, 'xlim', [0,10]);
set(gca,'ytick', -2 : 0.5 : 2, 'ylim', [-1.4,1.4]);
grid on;
error_fdm = immse(mt_fdm, mt)
mt_fmdemod = fmdemod(mt_fm,fc,fs,f_delta);
error_fmdemod_fdm = immse(mt_fdm, mt_fmdemod)
```

```
error_fdm = 4.4507e-05
error_fmdemod_fdm = 4.4543e-05
```

دمدولاسیون فاز (Phase DeModulation)

Part 5:



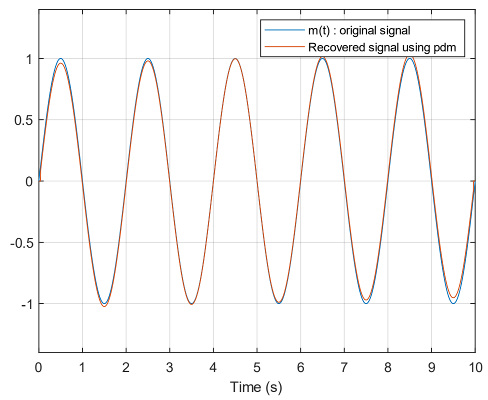
Part 6: pdm()

با توجه به نتایج قسمت 5 ، تابع ()pdm را در متلب بصورت زیر پیاده‌سازی میکنیم.

```
function x = pdm(Xc, Ac, phi_delta, fs, t)
    dXc = fs * gradient(Xc); % d/dt
    [EdXc,] = envelope(dXc,1000);   % Envelope Detector
    removeDC = EdXc - mean(EdXc);
    k = phi_delta * Ac;
    x = removeDC / k;
    x = cumtrapz(t, x);   % integral

    % for smoother edges and error reduction
    x(1, 1:round(fs/25)) = 0;
    x(1, length(x) - round(fs/25) : length(x)) = 0;
end
```

Part 7:8

در این قسمت میخواهیم خروجی تابع ()pm در قسمت 3 از بخش فرستنده را به تابع ()pdm بدهیم تا سیگنال پیام اولیه را بازیابی کنیم. سپس این کار را با استفاده از تابع ()pmdemod در متلب تکرار میکنیم.




```
phi_delta = 3;
mt_pm = pm(mt, Ac, phi_delta, fc, t);
mt_pdm = pdm(mt_pm, Ac, phi_delta, fs, t);
plot(t, mt, t, mt_pdm);
legend('m(t) : original signal', 'Recovered signal using pdm');
xlabel('Time (s)');
set(gca,'xtick', 0 : 1 : 10, 'xlim', [0,10]);
set(gca,'ytick', -2 : 0.5 : 2, 'ylim', [-1.4,1.4]);
grid on;
error_pdm = immse(mt_pdm, mt)
mt_pmdemod = pmdemod(mt_pm, fc, fs, phi_delta);
error_pmdemod_pdm = immse(mt_pdm, mt_pmdemod)
```

```
error_pdm = 7.2883e-04
error_pmdemod_pdm = 7.2883e-04
```

بخش سوم : مدولاسیون تک تن (Single Tone Modulation)

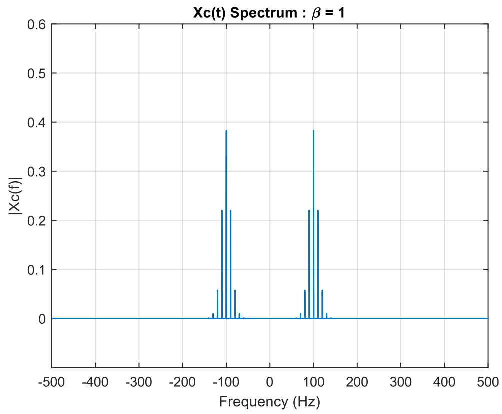
Part 1:

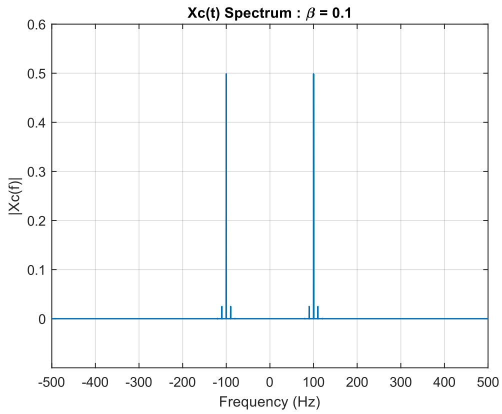
$$x_c(t) = A_c \cos\left(\omega_c t + \beta \sin(\omega_m t)\right) = \sum_{n=-\infty}^{+\infty} A_c J_n(\beta) \cos\left(2\pi(f_c + n f_m)t\right)$$

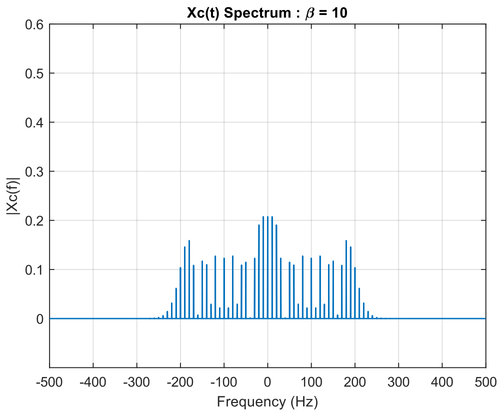
$$\mathcal{F}\{x_c(t)\} = \sum_{n=-\infty}^{+\infty} A_c J_n(\beta) \frac{1}{2}\left[\delta(f + f_c + n f_m) + \delta(f - f_c - n f_m)\right]$$

با توجه به روابط ریاضی نوشته شده ، مشخص است که تبدیل فوریه سیگنال مدوله شده ، قطاری از ضربه‌ها حول فرکانس موج حامل fc میباشد که ضرایب آن وابسته به β و تابع بسل است. اگر β به اندازه کافی کوچک باشد ، توابع بسل مرتبه‌های بالاتر را میتوان برابر با صفر در نظر گرفت. اما با افزایش β ، توابع بسل مرتبه‌های بالاتر نیز مقادیر خواهند داشت. پس در نتیجه در طیف سیگنال مدوله شده ، ضربه‌های بیشتری ظاهر میشود و به اصطلاح پهنای فرکانسی گسترده‌تر خواهد شد.

Part 2:








```
beta = 0.1;
mt = sin(20*pi*t);
xc = Ac * cos(2*pi*fc*t + beta*mt);
Xcf = fftshift(fft(xc));
l_Xcf = length(Xcf);
abs_Xcf = abs(Xcf/l_Xcf);
f1 = -fs/2 : fs/l_Xcf : fs/2 - fs/l_Xcf;
plot(f1, abs_Xcf, 'linewidth', 1);
title('Xc(t) Spectrum : \beta = 0.1');
xlabel('Frequency (Hz)');
ylabel('|Xc(f)|');
set(gca,'xtick', -500 : 100 : 500, 'xlim', [-500,500]);
set(gca,'ytick', 0 : 0.1 : 1, 'ylim', [-0.1,0.6]);
grid on;
beta = 1;
xc = Ac * cos(2*pi*fc*t + beta*mt);
Xcf = fftshift(fft(xc));
l_Xcf = length(Xcf);
abs_Xcf = abs(Xcf/l_Xcf);
f1 = -fs/2 : fs/l_Xcf : fs/2 - fs/l_Xcf;
plot(f1, abs_Xcf, 'linewidth', 1);
title('Xc(t) Spectrum : \beta = 1');
xlabel('Frequency (Hz)');
ylabel('|Xc(f)|');
set(gca,'xtick', -500 : 100 : 500, 'xlim', [-500,500]);
```

```
set(gca,'ytick', 0 : 0.1 : 1, 'ylim', [-0.1,0.6]);
grid on;
beta = 10;
xc = Ac * cos(2*pi*fc*t + beta*mt);
Xcf = fftshift(fft(xc));
l_Xcf = length(Xcf);
abs_Xcf = abs(Xcf/l_Xcf);
f1 = -fs/2 : fs/l_Xcf : fs/2 - fs/l_Xcf;
plot(f1, abs_Xcf, 'linewidth', 1);
title('Xc(t) Spectrum : \beta = 10');
xlabel('Frequency (Hz)');
ylabel('|Xc(f)|');
set(gca,'xtick', -500 : 100 : 500, 'xlim', [-500,500]);
set(gca,'ytick', 0 : 0.1 : 1, 'ylim', [-0.1,0.6]);
grid on;
```

تعداد ساعات تأخیر شناور (Grace) :

این تکلیف با 23 ساعت تاخیر آپلود شد. که تمام آن از Grace کسر شود.

تعداد ساعات Grace باقی مانده : 45 = 23 - 28 - 96