

بخش اول:

آزمایش‌های مربوط به پیاده‌سازی نرم‌افزاری

# آزمایش اول

## مقدمه

MATLAB یکی از نرم افزارهای انجام محاسبات ریاضی است. واژه MATLAB هم به معنی محیط محاسبات رقمی و هم به معنی خود زبان برنامه نویسی مربوطه است که از ترکیب دو واژه Matrix و Laboratory ایجاد شده است. این نام حاکی از رویکرد ماتریس محور برنامه است، که در آن حتی اعداد عادی هم به عنوان ماتریس در نظر گرفته می شوند.

با نرم افزار MATLAB می توان ماتریس ها را به راحتی تغییر داد، توابع یا داده ها را ترسیم کرد، الگوریتم ها را اجرا کرد و هم چنین صفحات رابط میان کاربر و رایانه ایجاد کرد.

برنامه های MATLAB اکثراً متن باز هستند و در واقع MATLAB مفسر است، نه کامپایلر. قدرت MATLAB از انعطاف پذیری و راحت بودن کار با آن ناشی می شود، هم چنین شرکت سازنده و گروه های مختلف، از جمله دانشگاه های سراسر دنیا و برخی شرکت های مهندسی هر ساله جعبه ابزارهای خاص کاربردی به آن می افزایند که باعث افزایش کارایی و محبوبیت آن شده است.

در این آزمایش شما با دستورات مقدماتی پردازش سیگنال در حوزه زمان و فرکانس آشنا می شوید که می توانید با مراجعه به Help نرم افزار ویژگی های پیشرفته تر و مثال های بیش تری را یاد بگیرید.

## مراحل آزمایش

### دستورات plot و stem

1-1) یک سیگنال سینوسی با طول 2 sec، طول گام 0.01 sec، دامنه ی 5 واحد و فرکانس 1 Hz بسازید و با دستورات plot و stem نمایش دهید. (توجه داشته باشید که نمودار باید دارای عنوان مناسب باشد و هم چنین محورها به درستی نام گذاری شوند، برای این کار می توانید از دستورات xlabel، ylabel و title استفاده کنید.)

### دستور subplot

1-2) اکنون یک سیگنال تصادفی با توزیع یکنواخت در بازه ی (0.5, -0.5) را با سیگنال بالا جمع کرده و هر دو سیگنال سینوسی نویزی و بدون نویز را با استفاده از دستور subplot در یک پنجره زیر هم نمایش دهید.

## دستور conv

3-1) با استفاده از تابع conv سیگنال نویزی را از یک سیستم moving average با طول 21 عبور دهید ( $\frac{1}{M_1+M_2+1} \sum_{k=-M_1}^{M_2} x[n-k]$ ) با مقادیر  $M_1 = 0$  و  $M_2 = 20$  و خروجی را در یک نمودار نمایش دهید. در مورد مشاهدات خود و علت تأخیر ایجاد شده در خروجی توضیح دهید. برای  $M_1 = M_2 = 10$  چگونه می‌توان از دستور conv استفاده کرد؟

## دستور filter

4-1) مرحله‌ی قبل را این بار با دستور filter تکرار نمایید. آیا خروجی با قسمت قبل یکسان است؟

## تعریف تابع

5-1) تابعی به نام  $\text{singen}(\omega, n)$  تعریف کنید که  $\omega$  و  $n$  به ترتیب فرکانس سینوسی و تعداد نمونه‌ها می‌باشد ( $0 < \omega < \pi$ ). می‌توانید از دستور filter استفاده نمایید.

$$x[n] = \sin(\omega_0 n) u_{-1}[n] \xrightarrow{z.T.} X(z) = \frac{\sin(\omega_0) z^{-1}}{1 - 2 \cos(\omega_0) z^{-1} + z^{-2}} \quad |z| > 1$$

## بررسی aliasing در حوزه زمان

6-1) سیگنال  $x(t) = \cos(2\pi t) + \cos(8\pi t) + \cos(12\pi t)$  را در نظر بگیرید، که دارای مولفه‌های فرکانسی 1 KHz، 4 KHz و 6 KHz می‌باشد (زمان به ms بیان شده است). این سیگنال را در بازه 0 تا 4 ms رسم کرده و سپس با نرخ 5 kHz نمونه برداری کنید و نمونه‌ها را با دایره روی نمودار مشخص کنید. اکنون سیگنال نمونه برداری شده را با فیلتر ایده‌آل بازسازی کرده و خروجی را روی همان نمودار قبل نشان دهید. انتظار دارید سیگنال بازسازی شده به چه شکلی دربیاید؟ حدس خود را اثبات کنید.

سوال: محدود کردن طول فیلتر ایده‌آل چه تاثیری در سیگنال بازسازی شده دارد؟

## بررسی aliasing در حوزه فرکانس

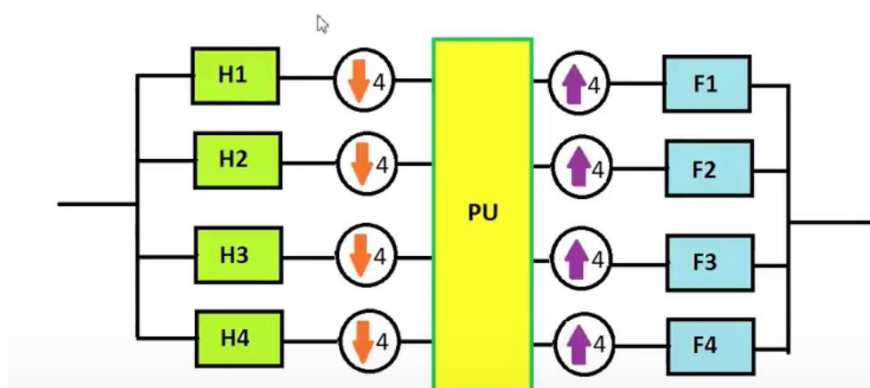
7-1) سیگنال  $x(t) = \text{sinc}^2(5t)$  را در بازه‌ی  $[-5, 5]$  با طول گام‌های 0.01 رسم کنید. سپس این سیگنال را با نرخ‌های 20 Hz، 10 Hz، 5 Hz و 4 Hz نمونه برداری کنید. طیف هر دو سیگنال را در هر حالت در یک پنجره در دو نمودار زیرهم رسم کرده و با هم مقایسه کنید. در مورد علت تفاوت احتمالی بحث کنید.

## بررسی aliasing در حوزه فرکانس

8-1) سیگنال  $x(t) = \text{sinc}(2t)$  را در بازه  $[-5, 5]$  در 256 نقطه نمونه برداری شده است. نرخ نمونه برداری را با نسبت  $1/2$ ،  $3$  و  $3/2$  تغییر داده و سیگنال اصلی و نمونه برداری شده و طیف نرمالیزه‌ی آن‌ها در هر حالت را در یک نمودار رسم کنید.

## بانک فیلتری چند نرخی

بلوک دیاگرام زیر را در نظر بگیرید:



این ساختار یک بانک فیلتری چهارکاناله با یک واحد پردازش را نشان می‌دهد. در این قسمت با یکی از کاربردهای ساده‌ی این ساختارها آشنا می‌شویم.

9-1-الف) سیگنال زیر را در نظر بگیرید که از چهار سینوسی با فرکانس‌های مختلف تشکیل شده است:

$$x(t) = \sum_{i=1}^4 \cos(2\pi f_i t), \quad f_1 = \frac{\pi}{16} \quad f_2 = \frac{5\pi}{16} \quad f_3 = \frac{9\pi}{16} \quad f_4 = \frac{13\pi}{16}$$

سیگنال  $x(t)$  را با استفاده از دستورهای `fft` و `fftshift` در حوزه فرکانس رسم کنید.

9-1-ب) با استفاده از دستور `xlsread` ضرایب فیلترهای تجزیه و ترکیب را `import` کنید.

توجه: ضرایب مربوط به فیلترهای تجزیه در sheet 1 و ضرایب مربوط به فیلترهای ترکیب در sheet 2 می‌باشد. هر کدام متشکل از یک ماتریس  $4 \times 32$  می‌باشد که هر سطر شامل ضرایب فیلتر متناظر خود است.

8-1-ج) با استفاده از بانک فیلتری مولفه‌ی فرکانسی اول را 2 برابر، دومی را حذف، سومی بدون تغییر و مولفه‌ی چهارم را با ضریب 0.5 تضعیف نمایید. نتیجه را با رنگی متفاوت روی نموداری که در قسمت 9-1-الف رسم کردید نمایش دهید.

# آزمایش دوم

## مقدمه

بسته به کاربرد و سخت‌افزار، عملیات فیلترینگ دیجیتال می‌تواند به صورت بلوکی یا نمونه به نمونه اعمال شود. در حالت پردازش بلوکی، سیگنال ورودی به صورت یک بلوک بزرگ از نمونه‌های سیگنال در نظر گرفته می‌شود. سیگنال فیلترشده، حاصل کانولوشن این بلوک و فیلتر است که خود یک بلوک از نمونه‌هاست. اگر سیگنال ورودی خیلی بزرگ و یا با طول بی‌نهایت باشد این روش به تغییراتی جزئی نیاز دارد، به طور مثال شکستن ورودی به چندین بلوک.

یک روش دیگر، پردازش نمونه به نمونه می‌باشد، بدین صورت که هرگاه یک نمونه ورودی می‌رسد فیلتر شده و نمونه خروجی متناظر تولید می‌شود. این روش در کاربردهای بلادرنگ که ورودی بسیار طولانی است سودمند می‌باشد. این روش هم‌چنین در فیلترینگ وفقی (adaptive) که خود فیلتر بعد از هر عمل فیلترینگ تغییر می‌کند کاربرد دارد.

در این آزمایش ابتدا روش فیلترینگ با کانولوشن بررسی شده و سپس مقدمه‌ای پردازش سیگنال در حوزه‌ی فرکانس مطرح خواهد شد.

## مراحل آزمایش

### کانولوشن

کانولوشن یک فیلتر  $h_n, n = 0, 1, 2, \dots, M$  با یک سیگنال علی زمان محدود  $x_n, n = 0, 1, 2, \dots, L$  با معادله‌ی زیر محاسبه می‌شود:

$$y_n = \sum_{m=0}^{\min(-L+n, M)} h_m x_{n-m} \quad n = 0, 1, 2, \dots, M + L - 1 \quad (1-2)$$

1-2-الف) تابعی به نام myconv (برای جلوگیری از ابهام با تابع conv داخلی متلب) بنویسید که معادله بالا را پیاده‌سازی کند. این تابع باید به صورت زیر باشد:

$$y = \text{myconv}(h, x) \quad (2-2)$$

که  $h$  و  $x$  ورودی‌ها و  $y$  خروجی فیلتر می‌باشد.

2-1-ب) یک فیلتر شبه‌انتگرال‌گیر را که با رابطه ورودی-خروجی زیر تعریف می‌شود، در نظر بگیرید:

$$y[n] = 0.1(x[n] + x[n-1] + x[n-2] + \dots + x[n-9]) \quad (3-2)$$

یک سیگنال ورودی موج‌مربعی را با طول  $L=200$  و دوره تناوب  $K=50$  نمونه را به آن اعمال کنید. با استفاده از تابع `myconv`، خروجی  $y[n]$  را محاسبه کرده و نمودار میله‌ای ورودی و خروجی را رسم کرده و در مورد مشاهدات خود توضیح دهید.

2-1-ج) قسمت قبل را برای فیلتر زیر تکرار نمایید.

$$h[n] = \begin{cases} 0.25(0.75)^n, & 0 \leq n \leq 14 \\ 0, & \text{otherwise} \end{cases} \quad (4-2)$$

2-1-د) قسمت (2-1-ب) را برای فیلتر با تابع تبدیل زیر تکرار نمایید:

$$H(z) = \frac{1}{5}(1 - z^{-1})^5 \quad (5-2)$$

این فیلتر به عنوان یک مشتق‌گیر مرتبه 5 عمل می‌کند.

### فیلترینگ سیگنال‌های نویزی

سیگنال  $x[n]$  مجموع یک سیگنال مطلوب  $s[n]$  و تداخل  $v[n]$  می‌باشد:

$$x[n] = s[n] + v[n] \quad (6-2)$$

$$s[n] = \sin(\omega_2 n) \quad (7-2)$$

$$v[n] = \sin(\omega_1 n) + \sin(\omega_3 n) \quad (8-2)$$

$$\omega_1 = 0.05\pi$$

$$\omega_2 = 0.20\pi$$

$$\omega_3 = 0.35\pi$$

به منظور حذف  $v[n]$ ، سیگنال  $x[n]$  با یک فیلتر میان‌گذر FIR فیلتر می‌شود، که به گونه‌ای طراحی شده‌است که فرکانس  $\omega_2$  را عبور داده و فرکانس‌های  $\omega_1$  و  $\omega_3$  را عبور ندهد.

یک نمونه از چنین فیلتری از مرتبه  $M=100$  می‌تواند با استفاده از پنجره Hamming طراحی شود که پاسخ ضربه زیر را داشته‌باشد:

$$h[n] = w[n] \frac{\sin\left(\omega_b\left(n - \frac{M}{2}\right)\right) - \sin\left(\omega_a\left(n - \frac{M}{2}\right)\right)}{\pi\left(n - \frac{M}{2}\right)} \quad 0 \leq n \leq M \quad (9-2)$$

که  $\omega_a = 0.15\pi$  و  $\omega_b = 0.25\pi$  و باند موثر  $[\omega_a, \omega_b] = [0.15\pi, 0.25\pi]$  است.  $w[n]$  پنجره Hamming می باشد که به صورت زیر تعریف می شود:

$$w[n] = \begin{cases} 0.54 - 0.46 \sin\left(\frac{2\pi n}{M}\right), & 0 \leq n \leq M \\ 0, & otherwise \end{cases} \quad (10-2)$$

برای جلوگیری از مشکلات محاسباتی در  $n = M/2$  می توانید از تابع sinc متلب استفاده کنید.

2-2-الف) در یک نمودار  $x[n]$  و  $s[n]$  را بر حسب  $n$  رسم کنید.

2-2-ب)  $x[n]$  را از طریق فیلتر  $h[n]$  با دستور filter متلب، فیلتر کرده و  $s[n]$  و خروجی فیلتر شده  $y[n]$  را در یک نمودار رسم کنید. جدای از یک تاخیر ناشی از فیلتر،  $y[n]$  باید تقریباً شبیه  $s[n]$  باشد.

2-2-ج) با استفاده از جعبه ابزار fdatool یک فیلتر میان گذر با ویژگی های زیر طراحی کرده و قسمت قبل را تکرار نمایید.

$$\omega_{s1} = 0.1$$

$$\omega_{p1} = 0.15$$

$$\omega_{p2} = 0.25$$

$$\omega_{s2} = 0.3,$$

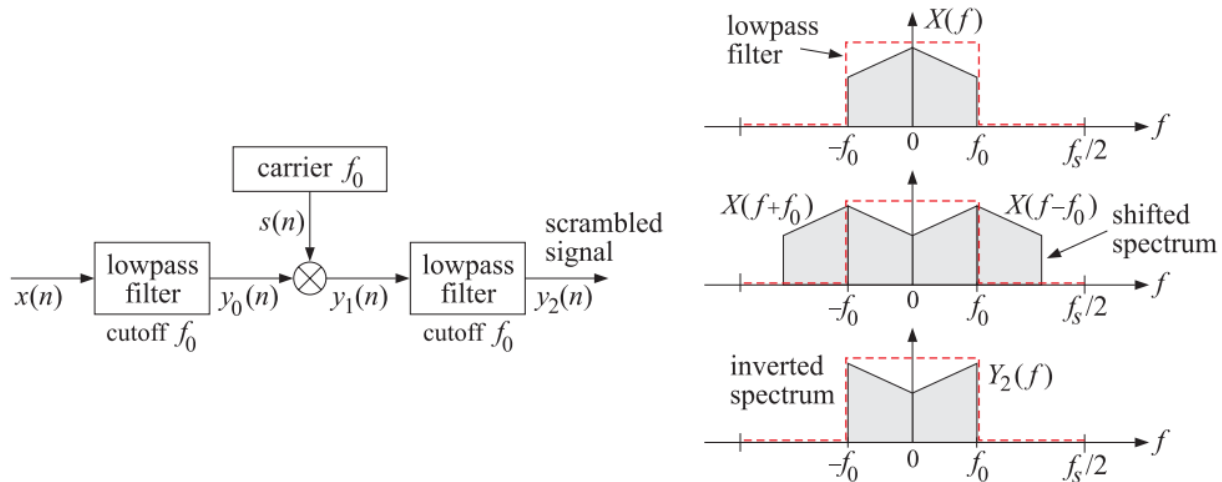
$$A_{s1} = 60dB$$

$$A_p = 1dB$$

$$A_{s2} = 60dB$$

## درهم سازی صوت (voice scrambling)

یک درهم ساز صوت ساده از طریق معکوس کردن طیف کار می کند. اگرچه این روش امن ترین راه کد کردن نیست اما این روش را به عنوان کاربردی از فیلترهای پایین گذر و مدولاسیون AM در نظر می گیریم. عملیات کلی در زیر نشان داده شده است:



ابتدا سیگنال صدای نمونه‌برداری شده‌ی  $x[n]$  با فیلتر پایین گذر  $h[n]$  می‌شود (فرکانس قطع آن  $f_0$  به اندازه‌ی کافی بالا هست تا اعوجاج ایجاد نکند). نرخ نمونه‌برداری  $f_s$  به گونه‌ای انتخاب می‌شود که  $f_s > 4f_0$ . عمل فیلترینگ می‌تواند با رابطه‌ی کانولوشن نمایش داده شود:

$$y_0[n] = \sum_m h[m]x[n-m] \quad (11-2)$$

سپس خروجی فیلتر با یک حامل کسینوسی با فرکانس  $f_0$  مدوله می‌شود:

$$y_1[n] = s[n]y_0[n], \quad s[n] = 2\cos\left(\frac{2\pi f_0}{f_s}n\right) \quad (12-2)$$

این ضرب در سیگنال حامل باعث می‌شود که طیف سیگنال شیف‌ت پیدا کند و مرکز آن  $\pm f_0$  شود. نهایتاً سیگنال مدوله شده  $y_1[n]$  دوباره از همان فیلتر پایین گذر عبور داده می‌شود که مولفه‌های طیف با  $|f| > f_0$  را حذف می‌کند. نتیجه سیگنال  $y_2[n]$  با طیف معکوس نسبت به  $y_0[n]$  می‌شود. عملیات آخر مطابق زیر است:

$$y_2[n] = \sum_m h[m]y_1[n-m] \quad (13-2)$$

برای بازیابی صدای اصلی از صدای درهم‌ریخته می‌توان عملیات بالا را دوباره روی آن اعمال کرد. چون طیف معکوس شده دوباره معکوس می‌شود و می‌توانیم طیف اصلی را بازیابی کنیم.

3-2-الف) با استفاده از دستور audioread متلب سیگنال صوت با نام 'Audio01.wav' را بارگذاری کنید.

3-2-ب) با استفاده از fdatool متلب یک فیلتر پایین‌گذر با مشخصات زیر طراحی کنید:

$$f_p = 10000, f_s = 12000, A_p = 0.5, A_s = -60$$



و سپس ضرایب آن را در قالب MAT-File و نام متغیر filter ذخیره کنید.

2-3-ج) فیلتر طراحی شده در قسمت قبل را به سیگنال صوت اعمال کرده و سیگنال حاصله را به سیگنال حامل ضرب کنید. سپس دوباره از همان فیلتر عبور دهید. برای شنیدن نتیجه می‌توانید از دستور sound استفاده کنید.

2-3-د) با تکرار عملیات فوق سیگنال صوت اصلی را بازیابی کرده و به آن گوش دهید. آیا با سیگنال صدای اصلی مطابقت دارد؟

## تبدیل فوریه‌ی گسسته و تبدیل فوریه‌ی زمان کوتاه

سیگنالی در نظر بگیرید که جمع یک سینوسی، یک سیگنال Chirp و یک ضربه است. فرکانس سیگنال سینوسی 100 Hz است که در بازه‌ی [0, 2] با فرکانس 500 Hz نمونه‌برداری شده‌است. فرکانس سیگنال Chirp هم به‌صورت خطی در بازه‌ی [0, 2] از 400 Hz به 200 Hz تغییر می‌کند. سیگنال ضربه در نقطه‌ی 250 ام با دامنه‌ی 50 است.

2-4-الف) طیف این سیگنال را برحسب فرکانس با استفاده از دستور fft رسم کنید.

2-4-ب) با استفاده از دستور spectrogram تبدیل فوریه‌ی زمان کوتاه آن را رسم کنید. برای این کار از پنجره‌ی Hamming استفاده کنید. طول پنجره را 256 و 512 قرار دهید. با توجه به مشاهدات خود تفاوت تبدیل فوریه گسسته و تبدیل فوریه‌ی زمان کوتاه را شرح دهید. با تغییر طول پنجره رزولوشن زمانی و رزولوشن فرکانسی چه تغییری می‌کنند؟

## تبدیل موجک (Wavelet Transform)

از دیدگاه تئوری سیگنال، تبدیل موجک تصویر کردن سیگنال روی مجموعه‌ای از توابع پایه به نام موجک‌هاست. تبدیل موجک به‌عنوان جایگزینی برای STFT ارائه شد تا بر مشکلات مربوط به رزولوشن زمانی و فرکانسی آن غلبه کند. برخلاف STFT که رزولوشن زمانی یکنواختی برای همه‌ی فرکانس‌ها دارد، تبدیل موجک برای فرکانس‌های بالا، رزولوشن زمانی بالا و رزولوشن فرکانسی پایین و برای فرکانس‌های پایین، رزولوشن زمانی پایین و رزولوشن فرکانسی بالا فراهم می‌کند. در حالت عمومی، موجک یک تابع با میانگین صفر است:

$$\int_{-\infty}^{+\infty} \psi(t) dt = 0$$

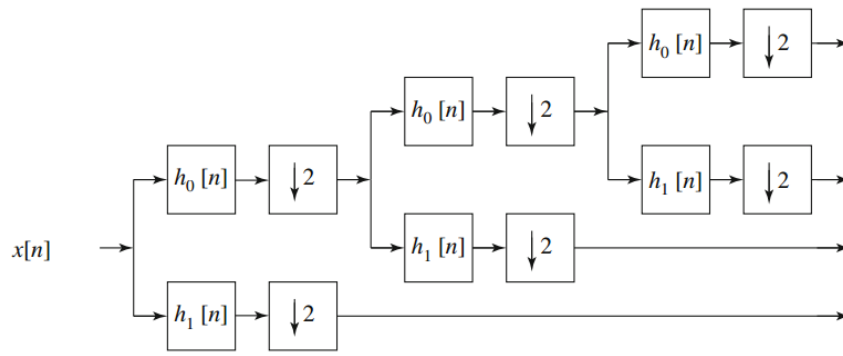
که شروطی را ارضا می‌کند تا اطمینان حاصل شود که می‌تواند در تجزیه‌ی چند رزولوشن استفاده شود. می‌توان با جابجایی و تغییر مقیاس یک موجک مادر  $\psi$  خانواده‌ای از موجک‌ها داشته‌باشیم:

$$\psi_{s,u}(t) = \frac{1}{\sqrt{s}} \psi\left(\frac{t-u}{s}\right)$$

تبدیل موجک تابع  $f$  در زمان  $u$  و مقیاس  $s$  به صورت زیر تعریف می شود:

$$\mathcal{W}(f, s, u) = \int_{-\infty}^{+\infty} f(t) \psi_{s,u}(t) dt$$

موجک های گسسته هم از جابجایی و تغییر مقیاس با گام های گسسته از موجک مادر حاصل می شوند. اصول تبدیل موجک گسسته مشابه روش Subband Coding است. بلوک دیاگرام زیر تبدیل موجک سه سطحی را نشان می دهد.



همان طور که مشاهده می کنید سیگنال  $x[n]$  به سه سطح تجزیه شده است. پروسه ی بالا یک بانک فیلتری (filter bank) با سه سطح است. تبدیل موجک یک سیگنال گسسته مطابق زیر اجرا می شود:

ابتدا  $x[n]$  از هر دو فیلتر پایین گذر  $h_0[n]$  و بالاگذر  $h_1[n]$  عبور می کند. سپس سیگنال های حاصله با فاکتور 2، downsample می شوند. سیگنال های حاصله یا دو مجموعه ضرایب، تبدیل سطح 1 سیگنال نامیده می شوند. اگر این پروسه مکرراً روی حاصل خط پایین گذر اعمال شود، تبدیل های سطح بالاتر به دست می آیند.

2-5) یکی از کاربردهای تبدیل موجک Denoising است. با استفاده از تابع wnoise از جعبه ابزار موجک، سیگنال نویزی ایجاد کرده و با استفاده از دستور dwt چند سطح از آن تبدیل موجک بگیرید. مشاهدات خود را شرح دهید.

# آزمایش سوم

## مقدمه

قسمت اول این آزمایش اصول بهبود سیگنال و کاهش نویز را با یک مثال ساده فیلتر IIR نشان می‌دهد. قسمت دوم فعل و انفعال بین پاسخ گذرا و مانا و trade off بین ثابت زمانی و تیزی فیلتر را بیان می‌کند. قسمت سوم نگاهی است بر مثال‌هایی از پردازنده‌های دینامیک برای سیگنال‌های صوتی، مانند فشرده سازها، محدودکننده‌ها، منبسط کننده‌ها و گیت‌های نویز.

## مراحل آزمایش

### بهبود سیگنال و کاهش نویز (Signal Enhancement and Noise Reduction)

یک سیگنال سینوسی نویزی با فرکانس  $f_0 = 500 \text{ Hz}$  را در نظر بگیرید که با نرخ  $f_s = 10 \text{ KHz}$  نمونه‌برداری شده است:

$$x[n] = \cos(\omega_0 n) + v[n] \quad (1 - 3)$$

که  $\omega_0 = 2\pi f_0 / f_s$  فرکانس دیجیتال و  $v[n]$  نویز می‌باشد.

مطلوب، طراحی فیلتر  $H(j\omega)$  به منظور استخراج سیگنال مطلوب  $s[n]$  از سیگنال نویزی  $x[n]$  می‌باشد. چنین فیلتری باید دو ویژگی داشته باشد: اولاً باید مولفه نویز را تا حد امکان حذف کند، ثانیاً باید به سیگنال مطلوب اجازه دهد که بدون تغییر عبور کند، به جز احتمالاً با یک تاخیر زمانی.

خواسته دوم با طراحی یک فیلتر میان‌گذر که باند عبورش با سیگنال مطلوب تلاقی دارد برآورده می‌شود. مولفه‌ی نویز معمولاً یک نویز سفید است که توان آن به صورت یکنواخت روی کل محور فرکانسی توزیع شده است. بعد از فیلترینگ، توان مجموع نویز کاهش خواهد یافت زیرا تنها، توانی که در بلند عبور قرار گرفته باقی خواهد ماند.

در مثال ما باند عبور سیگنال مطلوب فقط فرکانس  $\omega_0$  می‌باشد. بنابراین می‌بایست فیلتر میانگذری طراحی کنیم که در فرکانس  $\omega_0$  بهره واحد داشته‌باشد، یعنی  $|H(j\omega_0)| = 1$ . ساده‌ترین انتخاب ممکن برای چنین فیلتری تشدیدکننده دوطبقه با قطب‌هایی در  $\text{Re}^{\pm j\omega_0}$  می‌باشد که  $R$  باید برای پایداری به گونه‌ای انتخاب شود که  $0 < R < 1$ . تابع تبدیل، به فرم زیر خواهد بود:

$$H(z) = \frac{G}{(1 - Re^{j\omega_0}z^{-1})(1 - Re^{-j\omega_0}z^{-1})} = \frac{G}{1 + a_1z^{-1} + a_2z^{-2}} \quad (2-3)$$

که  $a_1 = -2R\cos(\omega_0)$  و  $a_2 = R^2$  می‌باشد. بهره  $G$  برای اطمینان از  $|H(\omega_0)| = 1$  زیر می‌باشد:

$$G = (1 - R)(1 - 2R\cos(2\omega_0) + R^2)^{\frac{1}{2}} \quad (3-3)$$

در حوزه زمان این فیلتر با معادله دیفرانس زیر توصیف می‌شود:

$$y[n] = -a_1y[n-1] - a_2y[n-2] + Gx[n] \quad (4-3)$$

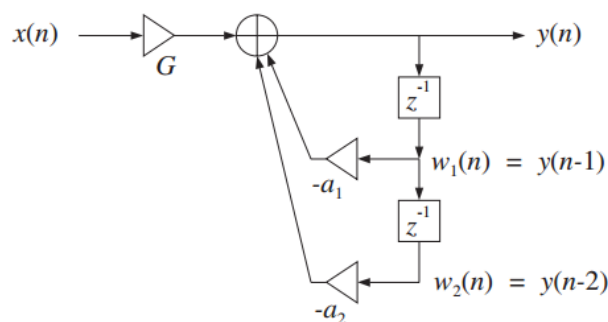
پاسخ ضربه این فیلتر می‌تواند به صورت زیر نشان داده شود:

$$h[n] = \frac{G}{\sin(\omega_0)} R^n \sin(\omega_0 n + \omega_0), \quad n = 0, 1, 2, \dots \quad (5-3)$$

در نهایت توان دوم پاسخ دامنه می‌تواند به صورت زیر نشان داده شود:

$$|H(j\omega)|^2 = \frac{G^2}{[1 - 2R\cos(\omega - \omega_0) + R^2][1 - 2R\cos(\omega + \omega_0) + R^2]} \quad (6-3)$$

یک بلوک‌دیگرام برای پیاده‌سازی معادله دیفرانس و تابع تبدیل بالا در زیر نشان داده شده‌است:



شکل 4-3

ابتدا روابط بالا را اثبات کنید. سپس به ازای هر کدام از مقادیر  $R=0.8$ ،  $R=0.90$  و  $R=0.99$  موارد زیر را انجام دهید:

3-1 الف) توان دوم پاسخ دامنه  $|H(e^{j\omega})|^2$  را در بازه‌ی فرکانسی  $0 < \omega < \pi$  رسم کنید.

پیشنهاد: برای رسم می‌توانید  $|H(e^{j\omega})|^2$  را در 500 نقطه با فاصله‌های مساوی در بازه مذکور محاسبه کنید.

$$\omega = \frac{2\pi f}{f_s} \quad \text{توجه:}$$

3-1-ب) پاسخ دامنه  $h[n]$  را با ارسال ضربه واحد در ورودی معادله دیفرانس و تکرار در زمان (با شرایط اولیه صفر) محاسبه کنید. مقادیر محاسبه شده را با مقادیری که از فرمول 3-5 به دست آمده مقایسه کنید. مقادیر  $h[n]/G$  را به ازای  $0 \leq n \leq 300$  رسم کنید.

3-1-ج) با استفاده از تابع تولید عدد تصادفی متلب، randn، 301 نمونه سیگنال تصادفی تولید کنید. سپس با استفاده از الگوریتم پردازش نمونه زیر  $x[n]$  را با  $H(z)$  فیلتر کرده و سیگنال خروجی  $y[n]$  را به ازای  $n$  در بازه  $0 \leq n \leq 300$  محاسبه و رسم کنید. در همان نمودار سیگنال بدون نویز  $s[n]$  را قرار دهید. بدهستان بین تیزی فیلتر میان گذر و سرعت پاسخ (سرعت رسیدن به حالت مانا) را مورد بحث قرار دهید.

For each input sample  $x$  do:

$$y = -a_1 w_1 - a_2 w_2 + Gx$$

$$w_2 = w_1$$

$$w_1 = y$$

3-1-د) سیگنال نویز  $v[n]$  را جداگانه از این فیلتر میان گذر عبور داده و نویز خروجی فیلتر شده متناظر  $y_v[n]$  را حساب کنید. در دو نمودار مجزا  $v[n]$  و  $y_v[n]$  را برجسب  $n$  رسم کنید. توضیح دهید که چرا نویز فیلتر شده بیشتر شبیه سینوسی است تا نویز.

3-1-ه) می توان نشان داد که برای یک ورودی نویز سفید میانگین صفر  $v[n]$  با واریانس  $\sigma_v^2$  که از یک فیلتر پایدار علی  $h[n]$  عبور می کند، سیگنال نویز خروجی  $y_v[n]$  واریانس  $\sigma_{y_v}^2$  را خواهد داشت که:

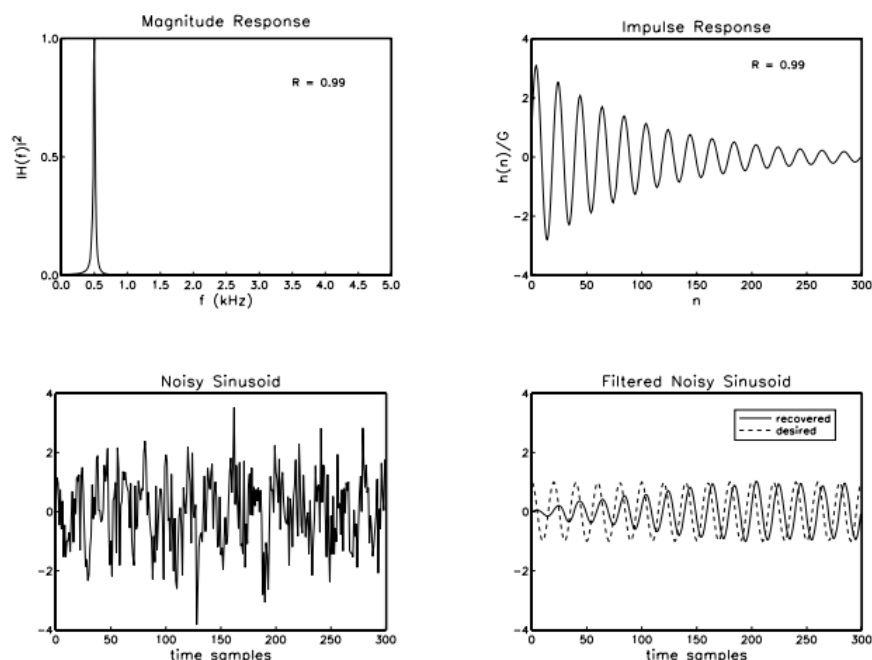
$$\frac{\sigma_{y_v}^2}{\sigma_v^2} = \sum_{n=0}^{\infty} h^2[n] \quad (7-3)$$

سمت چپ معادله نسبت کاهش نویز (NRR) می باشد، زیرا تاثیر فیلتر را بر نویز ورودی تعیین می کند. برای پاسخ ضربه خصوصی 3-5 اثبات کنید NRR مطابق رابطه زیر است:

$$\sum_{n=0}^{\infty} h^2[n] = \frac{1 + R^2}{(1 + R)(1 + 2R\cos(\omega_0) + R^2)} \quad (8-3)$$

(راهنمایی:  $h[n] = A_1 p_1^n + A_2 p_2^n$  را به فرم  $h[n] = A_1 p_1^n + A_2 p_2^n$  بنویسید که  $p_1$  و  $p_2$  قطبهای فیلتر هستند.)

با استفاده از سری‌های  $v[n]$  و  $y_v[n]$  که در قسمت 3-1 د تولید کردید و با تابع متلب std یک مقدار تقریبی برای سمت چپ معادله 3-7 حساب کنید و آن را با مقدار تئوری 3-8 مقایسه کنید.



شکل 3-2

## ویژگی‌های حالت گذرا و حالت مانا

پاسخ سینوسی فیلتر مرتبه‌ی دوم با قطب‌های  $p_1$  و  $p_2$  فرم دقیق زیر را خواهد داشت:

$$x[n] = \cos(\omega_0 n) \Rightarrow y[n] = |H(\omega_0)| \cos(\omega_0 n + \theta_0) + B_1 p_1^n + B_2 p_2^n \quad (9-3)$$

که شیفت فاز  $\theta_0$  مقدار فاز پاسخ فیلتر در فرکانس  $\omega_0$  می‌باشد و  $B_1$  و  $B_2$  به پاسخ‌های خصوصی تابع تبدیل بستگی دارد. در این بخش شما خواهید دید که ترم مانا به خوبی خروجی یک سیگنال زمان کوتاه را تعیین می‌کند و تاثیر ترم‌های گذرا بر ثابت زمانی فیلتر را مشاهده خواهید کرد.

سیگنال چندضابطه‌ای زیر را در نظر بگیرید:

$$x(t) = \begin{cases} \cos(2\pi f_1 t), & 0 \leq t < 2\text{sec} \\ \cos(2\pi f_2 t), & 2 \leq t < 4\text{sec} \\ \cos(2\pi f_3 t), & 4 \leq t < 6\text{sec} \end{cases} \quad (10-3)$$

که  $f_1 = 4\text{Hz}$ ,  $f_2 = 8\text{Hz}$  و  $f_3 = 12\text{Hz}$  می‌باشد.

این سیگنال با نرخ  $f_s = 400 \frac{\text{sample}}{\text{second}}$  نمونه برداری می شود. دو فیلتر زیر که در نرخ  $f_s$  عمل می کنند notch filter می باشند که در فرکانس  $f_2 = 8\text{Hz}$  یک شکاف دارند. بنابراین مشخصه میانی  $x(t)$  را از بین می برند:

$$H_1(z) = \frac{0.969531 - 1.923772z^{-1} + 0.969531z^{-2}}{1 - 1.923772z^{-1} + 0.939063z^{-2}} \quad (11 - 3)$$

$$H_2(z) = \frac{0.996088 - 1.976468z^{-1} + 0.996088z^{-2}}{1 - 1.976468z^{-1} + 0.992177z^{-2}} \quad (12 - 3)$$

فیلتر اول پهنای باند 3db برابر  $\Delta f = 4\text{Hz}$  و دومی پهنای  $\Delta f = 0.5\text{Hz}$  را دارد. پاسخ های دامنه  $|H(f)|$  در دو صفحه بعد نشان داده شده اند.

در این آزمایش ما رابطه ی بین پهنای شکاف و ثابت زمانی حالت گذرا را بررسی خواهیم کرد. فیلتر اول پهنای زیاد و ثلثت زمانی کوتاه دارد و فیلتر دوم پهنای کم و ثلثت زمانی طولانی دارد. Notch filter های مورد بحث ما، فرم کلی زیر را دارند:

$$H(z) = \frac{1}{1 + \beta} \cdot \frac{1 - 2 \cos(\omega_0) z^{-1} + z^{-2}}{1 - \frac{2 \cos(\omega_0)}{1 + \beta} z^{-1} + \frac{1 - \beta}{1 + \beta} z^{-2}} \quad (13 - 3)$$

که فرکانس شکاف و پهنای شکاف به ترتیب  $f_0$  و  $\Delta f_0$  می باشند و:

$$\omega_0 = \frac{2\pi f_0}{f_s} \quad \Delta\omega = \frac{2\pi \Delta f}{f_s} \quad \beta = \tan\left(\frac{\Delta\omega}{2}\right)$$

3-2-الف) تحقق کانونی هر فیلتر را رسم کنید و الگوریتم پردازش نمونه را با بافرهای خط تاخیر بنویسید.

3-2-ب) زمان نشست 1% هر دو فیلتر را حساب کنید.

3-2-ج) اگر  $x(t_n)$  ورودی نمونه برداری شده ی  $x(t)$  باشد،  $x(t_n)$  را بر حسب  $t_n$  روی یک دوره ی 6 sec رسم کنید. این ورودی را با فیلتر  $H_1(z)$  فیلتر کنید و خروجی  $y(t_n)$  را بر حسب  $t_n$  رسم کنید.

3-2-د) دقت کنید که با چه سرعتی مشخصه ی میانی  $x(t_n)$  برداشته می شود. هم چنین توجه کنید که مشخصه های  $f_1$  و  $f_3$  دیگر دامنه ی واحد ندارند. ثابت کنید که دامنه (های) حالت گذرا به ترتیب با اعداد پاسخ دامنه  $|H(f_1)|$  و  $|H(f_2)|$  داده شده اند. (شما می توانید این کار را با ارتفاع های  $|H(f_1)|$  و  $|H(f_3)|$  روی

مشخصات سیگنال متناظر انجام دهید یا با استفاده از تابع max بیشترین مقدار دو مشخصه را معلوم کرده و سپس آن‌ها را با مقادیر پاسخ محاسبه شده، مقایسه کنید.)

آیا یک شیفت فاز را مشاهده می‌کنید؟ آیا پاسخ گذرای مشاهده‌شده با ثابت زمانی قسمت 2-3-ب مطابقت دارد؟

2-3-ه) سوال‌های 2-3-ج و 2-3-د را برای فیلتر دوم  $H_2(z)$  تکرار نمایید.

2-3-و) روی دو نمودار مجزا پاسخ دامنه‌ی  $H_1(f)$  و  $H_2(f)$  را برحسب  $f$  در بازه‌ی  $0 < f < 20$  رسم کنید. نمودارهای مطلوب در صفحه‌ی بعد نشان داده شده‌اند. مقادیر  $f_1$  و  $f_3$  نیز روی نمودارها مشخص شده‌اند.

2-3-ز) برای هر کدام از فیلترها فرکانس 3db پایین و بالا را حساب کرده،  $f_L$  و  $f_H$  بنامید و آن را روی نمودار قسمت 2-3-و نشان دهید. ثابت کنید که اختلاف  $f_L$  و  $f_H$  به ترتیب برابر 4 Hz و 0.5 Hz می‌باشد.

2-3-ح) دو peak filter زیر را در نظر بگیرید که مکمل notch filter های بالا هستند. این‌ها یک قله در  $f_2$  و همان پهنای باند 4 Hz و 0.5 Hz را دارند:

$$H_1(z) = \frac{0.030469(1 - z^{-2})}{1 - 1.923772z^{-1} + 0.939063z^{-2}} \quad (3 - 14 - \text{الف})$$

$$H_2(z) = \frac{0.003912(1 - z^{-2})}{1 - 1.976468z^{-1} + 0.992177z^{-2}} \quad (3 - 14 - \text{ب})$$

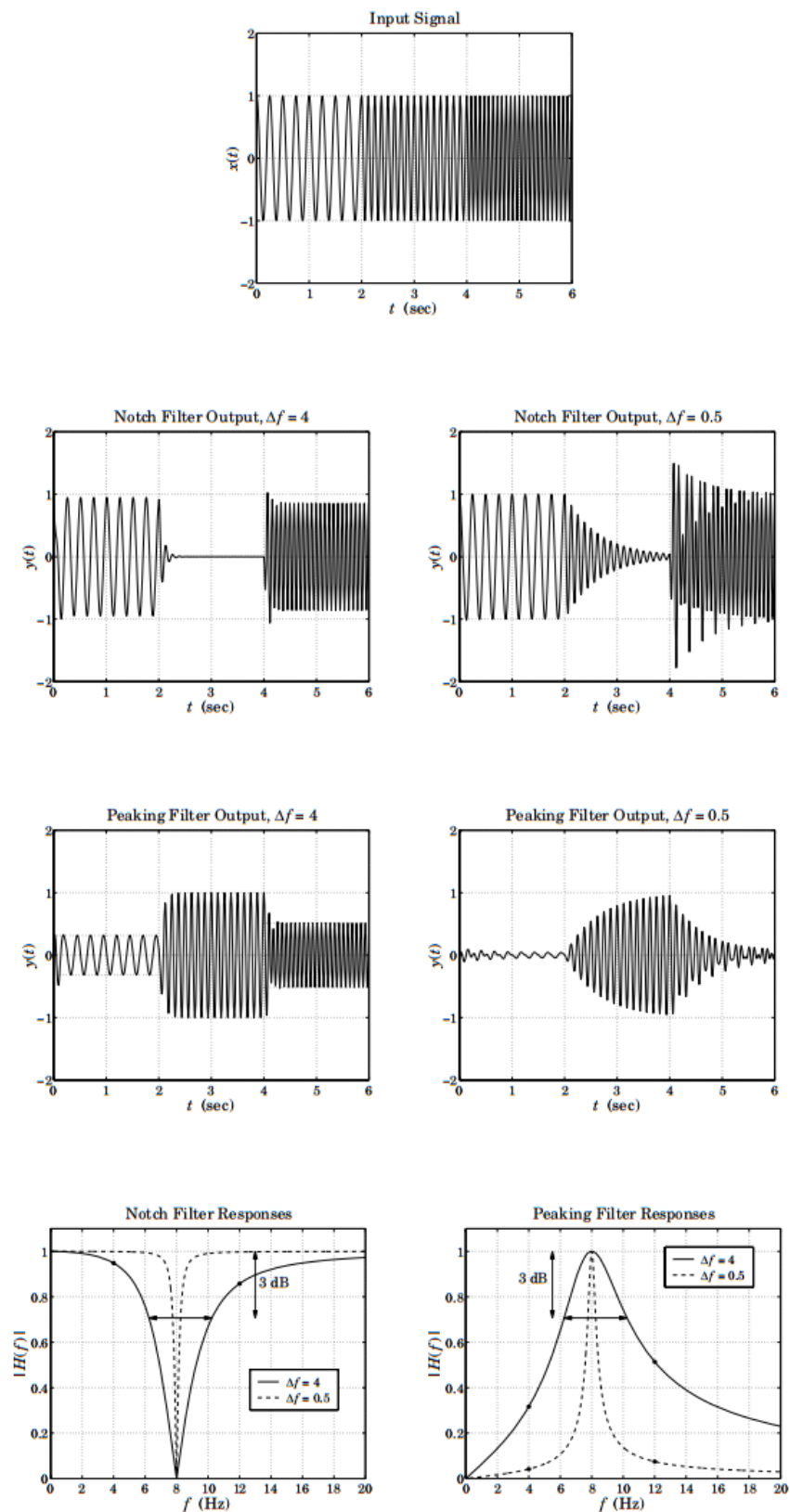
معادله‌ی کلی برای طراحی این فیلترها مطابق زیر است که  $f_0$  و  $\Delta f$  فرکانس قله و پهنای 3db قله می‌باشند:

$$H(z) = \frac{\beta}{1 + \beta} \cdot \frac{1 - z^{-2}}{1 - 2 \frac{\cos(\omega_0)}{1 + \beta} + \frac{1 - \beta}{1 + \beta} z^{-2}} \quad (3 - 15)$$

$$\omega_0 = \frac{2\pi f_0}{f_s} \quad \Delta\omega = \frac{2\pi\Delta f}{f_s} \quad \beta = \tan(\Delta\omega)$$

بخش‌های 2-3-الف تا 2-3-ز را برای این فیلترها تکرار کنید. انتظار داریم که peak filter مشخصه‌ی میانی ورودی را استخراج کرده و مشخصه‌های  $f_1$  و  $f_3$  را حذف کند.





شکل 3-3

## فشرده‌سازها، محدودکننده‌ها، منبسط‌کننده‌ها و گیت‌های نویز

فشرده‌سازها، محدودکننده‌ها، منبسط‌کننده‌ها و گیت‌های نویز کاربردهای وسیعی در پردازش صوت دارند. فشرده‌سازها سیگنال‌های قوی را تضعیف می‌کنند و چون روی بازه‌ی دینامیکی سیگنال‌ها اثر می‌گذارند به آن‌ها پردازنده‌های دینامیکی می‌گویند.

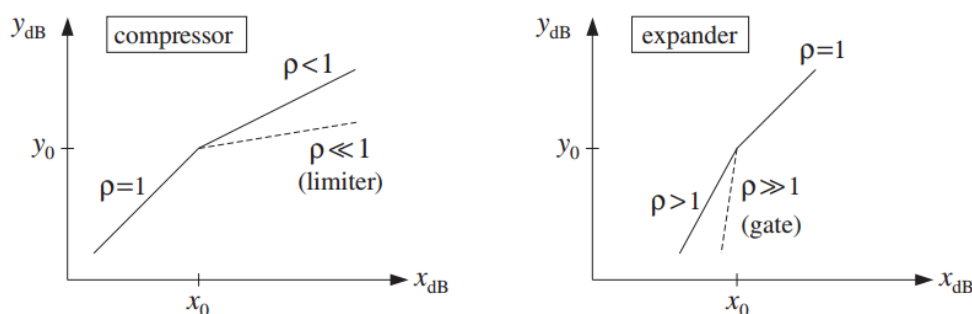
فشرده‌سازها غالباً به منظور کاهش بازه‌ی دینامیکی به کار می‌روند تا بازه‌ی دینامیکی پخش یا سیستم رسانه را مناسب کنند، مثلاً ضبط صدا روی یک نوار صوتی. چندین کاربرد دیگر هم وجود دارد، مانند موسیقی پس‌زمینه‌ی پنهان‌ساز گزارشگرها، de-essing برای حذف صفیر بیش از حد میکروفون و اثرات خاص دیگر.

منبسط‌کننده‌ها برای افزایش بازه‌ی دینامیکی سیگنال‌ها به منظور کاهش نویز و اثرات خاص گوناگون به کار می‌روند.

یک رابطه‌ی نوعی ورودی-خروجی حالت مانا برای فشرده‌سازها و منبسط‌کننده‌ها برحسب اندازه و دامنه، مطابق زیر است:

$$y = y_0 \left( \frac{x}{x_0} \right)^\rho \Rightarrow 20 \log_{10} \left( \frac{y}{y_0} \right) = \rho 20 \log_{10} \left( \frac{x}{x_0} \right) \quad (16-3)$$

که  $x$  در این‌جا یک ورودی ثابت و  $x_0$  یک مقدار آستانه‌ی دلخواه می‌باشد و هم‌چنین  $\rho$  نسبت انبساط یا فشرده‌سازی را تعریف می‌کند. یک فشرده‌ساز زمانی اثر می‌گذارد که  $x \geq x_0$  باشد و  $\rho < 1$  است. درحالی‌که برای منبسط‌کننده‌ها باید  $x \leq x_0$  بوده و  $\rho > 1$  می‌باشد. شکل 3-4 این روابط را به دسی‌بل نشان می‌دهد، به این ترتیب که 1db تغییر در ورودی باعث  $\rho$  dB تغییر در خروجی می‌شود که  $\rho$  شیب خط مستقیم بین ورودی و خروجی است.



شکل 3-4

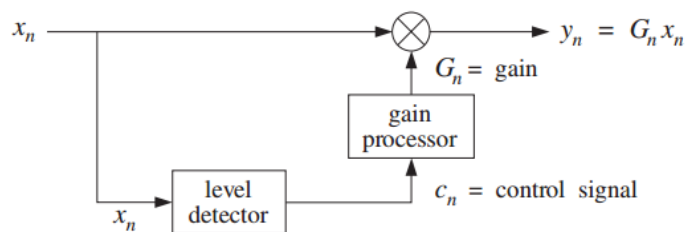
مقادیر عملی نوعی برای فشرده‌ساز بین  $\rho = (1/4 \text{ تا } 1/2)$  و برای منبسط‌کننده‌ها بین  $\rho = (2 \text{ تا } 4)$  می‌باشد. محدود کننده‌ها فرم‌های اکستریم فشرده‌سازها هستند که از این که سیگنال از یک آستانه حداکثر فراتر رود جلوگیری می‌کنند و شیب خیلی کمی دارند (مثلا 0.1). گیت‌های نویز حالت‌های اکستریم منبسط کننده‌ها هستند که سیگنال‌های ضعیف را بی‌نهایت تضعیف می‌کنند و بنابراین می‌توانند برای حذف نویز پس‌زمینه به کار روند و شیب خیلی زیادی دارند، مثلا (10).

رابطه‌ی ورودی-خروجی 3-16 تنها برای سیگنال‌های ثابت مناسب است. به ازای  $y = Gx$  مشاهده می‌کنیم که بهره‌ی موثر فشرده‌ساز تابعی غیر خطی از ورودی به فرم  $G = G_0 x^{p-1}$  می‌باشد. برای سیگنال‌های متغیر با زمان، بهره باید از یک میانگین محلی سیگنال حساب شود که بیانگر سطح سیگنال است.

مدلی از یک فشرده‌ساز/منبسط‌کننده در شکل 3-5 نشان داده شده‌است. تعیین‌کننده‌ی سطح، یک سیگنال کنترلی  $c_n$  تولید می‌کند که بهره‌ی ضرب‌کننده  $G_n$  را از طریق یک پردازنده‌ی بهره‌ی غیرخطی کنترل می‌کند. بسته به نوع فشرده‌ساز سیگنال کنترلی می‌تواند (1) مقدار قله‌ی لحظه‌ای  $|x_n|$ ، (2) پوش  $x_n$  و یا (3) مقدار موثر  $x_n$  (RMS) باشد. یک مدل ساده از آشکارساز پوش مطابق زیر است:

$$c_n = \lambda c_{n-1} + (1 - \lambda) |x_n| \quad (17 - 3)$$

معادله‌ی دیفرانس  $c_n$  به عنوان یکسوسازی که به دنبال آن آن یک فیلتر پایین‌گذر است عمل می‌کند. زمان خیز به یک سطح بالای آستانه (جایی که فشرده‌ساز فعال است) ثابت زمانی حمله (attack) نامیده می‌شود. این زمان می‌تواند با یک تاخیر  $D$  در ورودی آشکارساز بیش‌تر شود که  $|x_{n-D}|$  می‌باشد. زمان افت به یک سطح زیر آستانه (جایی که فشرده‌ساز غیرفعال می‌شود) زمان آزادسازی نامیده می‌شود.



شکل 3-5

به ازای  $\lambda = 0$  معادله‌ی 3-17 یک آشکارساز قله‌ی لحظه‌ای می‌شود. این حالت زمانی کاربرد دارد که از فشرده‌ساز به عنوان یک محدودکننده استفاده کنیم. اگر در معادله‌ی 3-17 به جای قدر مطلق  $|x_n|$  مربع آن  $|x_n|^2$  قرار بگیرد سیگنال کنترلی مقدار موثر ورودی را دنبال خواهد کرد.

پردازنده‌ی بهره یک تابع غیرخطی از سیگنال کنترلی است که از رابطه‌ی ورودی-خروجی 3-16 پیروی می‌کند. برای یک فشرده‌ساز می‌توانیم تابع بهره را این‌گونه تعریف کنیم:

$$f(c) = \begin{cases} \left(\frac{c}{c_0}\right)^{\rho-1}, & c \geq c_0 \\ 1, & c < c_0 \end{cases} \quad (18-3)$$

که  $c_0$  آستانه‌ی دلخواه و  $\rho < 1$  می‌باشد. برای یک منبسط‌کننده  $\rho > 1$  است و :

$$f(c) = \begin{cases} \left(\frac{c_0}{c}\right)^{\rho-1}, & c < c_0 \\ 1, & c \geq c_0 \end{cases} \quad (19-3)$$

بنابراین بهره‌ی  $G_n$  و سیگنال خروجی نهایی  $y_n$  مطابق زیر محاسبه می‌شوند:

$$G_n = f(c_n) \quad (20-3)$$

$$y_n = G_n x_n \quad (21-3)$$

فشرده‌سازها/منبسط‌کننده‌ها مثال‌هایی از سیستم‌های پردازش سیگنال وفقی هستند که ضرایب فیلتر (در این جا بهره‌ی  $G_n$ ) وابسته به زمان هستند و خودشان را با طبیعت سیگنال ورودی وفق می‌دهند. آشکارساز سطح 3-17 به عنوان معادله‌ی adaption عمل می‌کند. ثابت زمانی‌های حمله و آزادسازی، ثابت زمانی‌های آموزش سیستم وفقی هستند. پارامتر  $\lambda$  فاکتور فراموشی سیستم نامیده می‌شود.

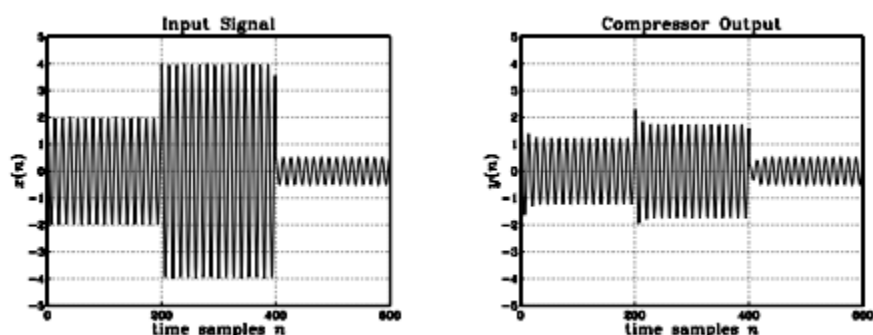
به عنوان یک مثال شبیه‌سازی یک سینوسی با فرکانس  $\omega_0 = 0.15\pi \text{ rad/sec}$  را در نظر بگیرید که دامنه‌اش هر 200 نمونه به سه مقدار  $A_1 = 2$ ،  $A_2 = 4$  و  $A_3 = 0.5$  تغییر می‌کند. همان‌طور که در شکل 3-6 نشان داده شده‌است که  $x_n = A_n \cos(\omega_0 n)$  با:

$$A_n = A_1(u_n - u_{n-200}) + A_2(u_{n-200} - u_{n-400}) + A_3(u_{n-400} - u_{n-600}) \quad (22-3)$$

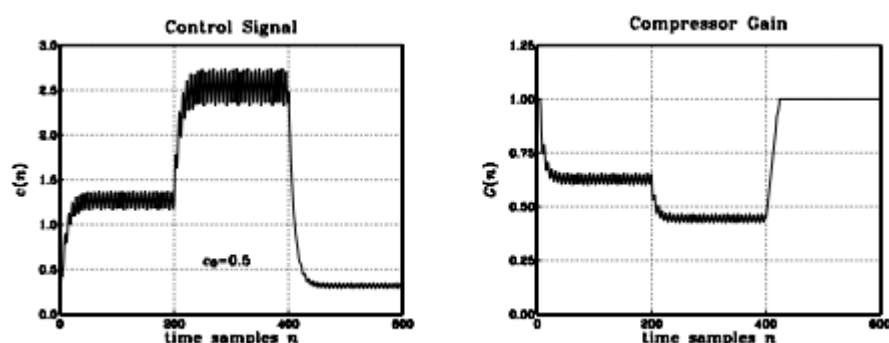
از یک فشرده‌ساز با پارامترهای  $\lambda = 0.9$ ،  $c_0 = 0.5$  و  $\rho = 0.2$  استفاده می‌شود. خروجی  $y_n$  در شکل 3-6 و سیگنال کنترلی  $c_n$  و بهره‌ی  $G_n$  در شکل 3-7 نشان داده شده‌اند.

در سینوسی اول  $A_1$  و  $A_2$  بالای آستانه قرار می‌گیرند و فشرده می‌شوند. سینوسی، بعد از گذشت زمان آزادسازی تأثیری نمی‌پذیرد. اگرچه تنها سیگنال‌های قوی‌تر تضعیف می‌شوند اما کاهش کلی بازه‌ی دینامیکی این‌گونه درک می‌شود که گویا سیگنال‌های ضعیف‌تر نیز تقویت شده‌اند. این ویژگی، ریشه‌ی این عبارت عامه‌پسند، اما گمراه‌کننده است که فشرده‌سازها سیگنال‌های قوی را تضعیف و سیگنال‌های ضعیف را تقویت می‌کنند.

پرش بین سطوح مانای  $A_1$  و  $A_2$  متناظر تغییر 6db است. چون هردو سطح فشرده می‌شوند سطوح مانای خروجی  $6\rho = 3\text{db}$  تفاوت خواهند داشت. به منظور حذف برخی از فراجش‌ها یک تاخیر مناسب در مسیر سیگنال قرار داده می‌شود که خروجی با  $y_n = G_n x_{n-d}$  محاسبه می‌شود.



شکل 6-3



شکل 7-3

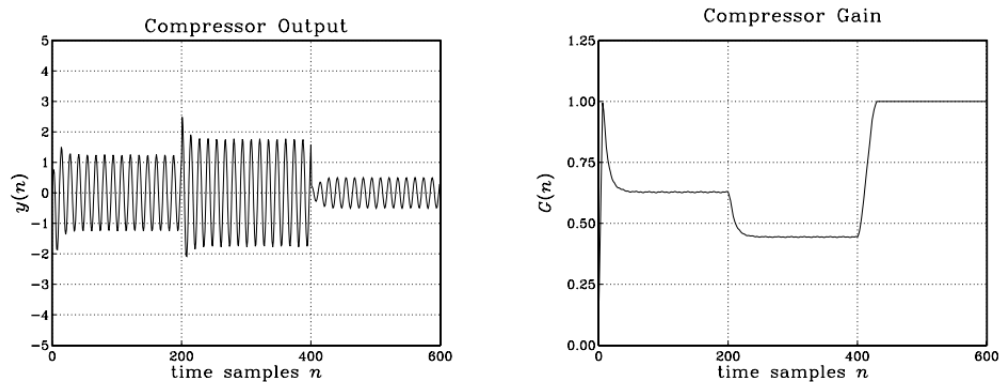
یک بهبودی دیگر هموار کردن بیش‌تر بهره‌ی غیرخطی  $g_n = f(c_n)$  با یک فیلتر پایین‌گذر می‌باشد، به گونه‌ای که بهره‌ی نهایی این‌گونه محاسبه شود:

$$G_n = \frac{1}{L} (g_n + g_{n-1} + \dots + g_{n-L+1}) \quad (23-3)$$

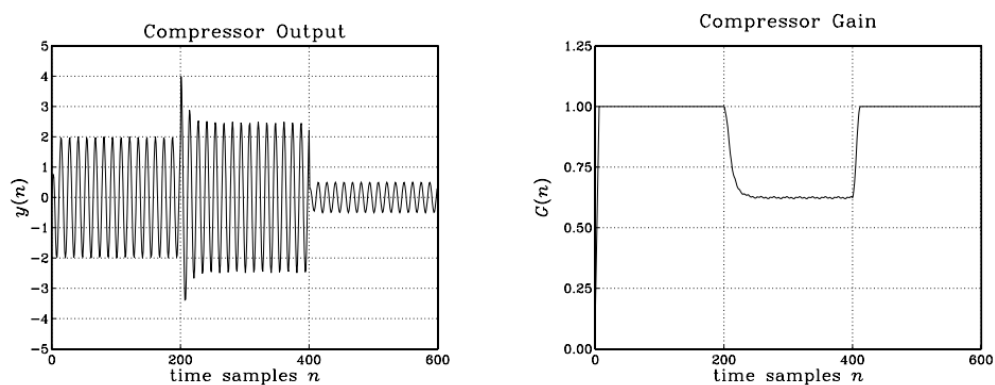
شکل 8-3 سیگنال خروجی و هم‌چنین بهره‌ی یک فشرده‌ساز را، که با یک هموارساز 7 نقطه‌ای هموار شده است، نشان می‌دهد. حالت‌های گذرای اولیه‌ی  $G_n$  به دلیل حالت گذرای ورودی‌روشن هموارساز ایجاد شده است.

شکل 9-3 سیگنال خروجی و بهره‌ی یک محدودکننده را نشان می‌دهد که نسبت فشرده‌سازی 10:1 دارد،  $\rho = 0.1$ ، و هم‌چنین از یک هموارساز 7 نقطه‌ای استفاده می‌کند. در این‌جا آستانه تا  $c_0 = 1.5$  افزایش یافته تا تنها  $A_2$  بالای آن قرار گیرد و فشرده شود.

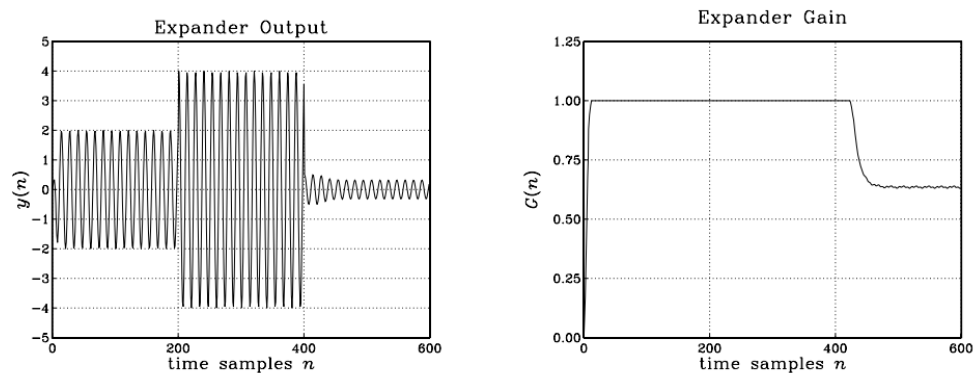
شکل 10-3 یک مثال منبسط‌کننده را با  $\lambda = 0.9$ ,  $c_0 = 0.5$ ,  $\rho = 2$  و تابع بهره‌ای که با معادله‌ی 19-3 محاسبه می‌شود، نشان می‌دهد که با یک هموارساز 7 نقطه‌ای هموارتر شده است. تنها  $A_3$  زیر آستانه قرار می‌گیرد و تضعیف می‌شود. این باعث می‌شود که بازه‌ی دینامیکی کلی افزایش یابد. اگرچه منبسط‌کننده فقط بر روی سیگنال‌های ضعیف‌تر اثر می‌گذارد اما بهره‌ی دینامیکی کلی این‌گونه درک می‌شود که گویا سیگنال‌های قوی‌تر بلندتر شده‌اند و ضعیف‌ترها آهسته‌تر.



شکل 8-3



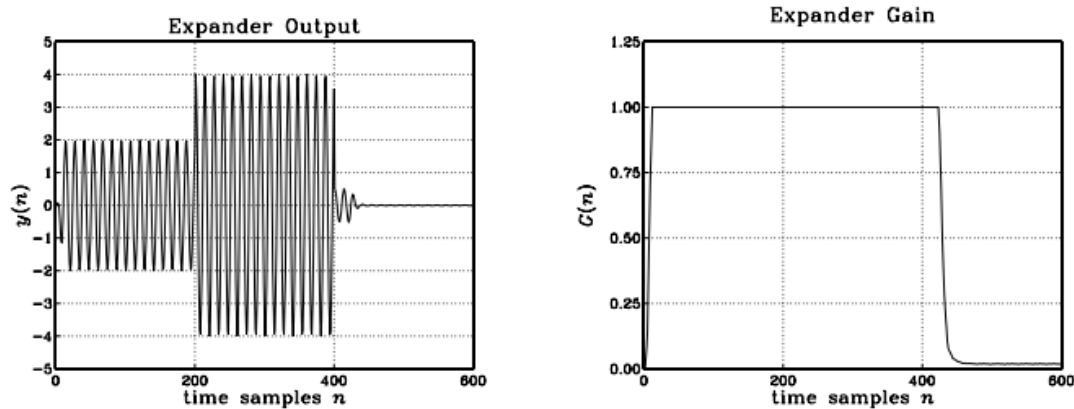
شکل 9-3



شکل 10-3

در نهایت شکل 3-11 یک مثال از گیت نویز را نشان می‌دهد که به عنوان یک منبسط کننده با نسبت انبساط 10:1،  $\rho = 10$  پیاده‌سازی شده است و همان آستانه‌ی شکل 3-10 را دارد و ضرورتاً سینوسی  $A_3$  را حذف می‌کند که می‌تواند مربوط به نویز ناخواسته باشد.

امکان بهبود الگوریتم 3-24 برای دستیابی به ثابت زمانی‌های مختلف حمله و آزادسازی وجود دارد. به هر حال ما در این آزمایش به همین الگوریتم‌ها اکتفا می‌کنیم.



شکل 3-11

3-3-الف) یک سینوسی  $x(t) = A \cos(2\pi ft)$  را در نظر بگیرید. نشان دهید که مقدار موثر آن روی یک دوره‌ی تناوب و میانگین قدرمطلق آن به ترتیب به صورت زیر است:

$$\overline{x^2(t)} = \frac{1}{2} A^2 \quad \overline{|x(t)|} = \frac{2}{\pi} A \quad (24-3)$$

بنابراین مقادیر قله‌ی  $[A_1 \ A_2 \ A_3] = [2 \ 4 \ 0.5]$  هر سه جزء سینوسی به مقادیر میانگین قدرمطلق خود نظیر می‌شوند که با آستانه‌ی فشرده‌ساز مفروض مقایسه خواهند شد:

$$[A_1 A_2 A_3] = [2 \ 4 \ 0.5] \Rightarrow \frac{2}{\pi} [A_1 A_2 A_3] = [1.27 \ 2.55 \ 0.32] \quad (25-3)$$

3-3-ب) برنامه‌هایی با متلب بنویسید که تمام نمودارهای شکل‌های 3-6 تا 3-11 را تولید کنند.

3-3-ج) نمودارهای فشرده‌ساز شکل‌های 3-6 تا 3-7 را این بار با  $\rho = 1/4$  و سایر پارامترهای قبلی انجام دهید.

3-3-د) نمودارهای فشرده‌ساز شکل‌های 3-6 تا 3-7 را این بار با  $c_0 = 1.3$ ،  $\rho = 1/2$  و  $\lambda = 0.9$  توضیح دهید که چرا فقط قسمت میانی فشرده می‌شود.

# آزمایش چهارم: آشنایی با پردازش تصویر

## مقدمه

پردازش تصاویر امروزه بیشتر به موضوع پردازش تصویر دیجیتال گفته می‌شود که شاخه‌ای از دانش رایانه است که با پردازش سیگنال دیجیتال که نماینده تصاویر گرفته شده با دوربین دیجیتال یا پوشش شده توسط پوششگر هستند سر و کار دارد.

پردازش تصویر دارای دو شاخه عمده‌ی بهبود تصاویر و بینایی ماشین است. بهبود تصاویر دربرگیرنده‌ی روش‌هایی چون استفاده از فیلترهای محوکننده و افزایش کنتراست برای بهتر کردن کیفیت دیداری تصاویر و اطمینان از نمایش درست آن‌ها در محیط مقصد (مانند چاپگر یا نمایشگر رایانه) است، در حالی که بینایی ماشین به روش‌هایی می‌پردازد که به کمک آن‌ها می‌توان معنی و محتوای تصاویر را درک کرد تا از آن‌ها در کارهایی چون رباتیک استفاده شود.

در معنای خاص آن پردازش تصویر عبارت است از هر نوع پردازش سیگنال که ورودی یک تصویر است مثل عکس یا صحنه‌ای از یک فیلم. خروجی پردازشگر تصویر می‌تواند یک تصویر یا مجموعه‌ای از نشان‌های ویژه یا متغیرهای مربوط به تصویر باشد. اغلب تکنیک‌های پردازش تصویر شامل برخورد با تصویر به عنوان یک سیگنال دو بعدی و به کار بستن تکنیک‌های استاندارد پردازش سیگنال روی آن‌ها می‌شود. پردازش تصویر اغلب به پردازش دیجیتالی تصویر اشاره می‌کند ولی پردازش نوری و آنالوگ تصویر هم وجود دارند. در این آزمایش با مفاهیم ابتدایی تصویر دیجیتال و همچنین کدهای مقدماتی متلب در پردازش تصویر آشنا خواهید شد.

## مراحل آزمایش

### 1- مبانی تصاویر دیجیتال

4-1-الف) تصویر با نام 'lena.bmp' را با استفاده از دستور `imread` خوانده، درون یک متغیر بریزید و سپس با استفاده از دستور `imshow` آن را نمایش دهید.

4-1-ب) دو نوع رایج تصاویر در MATLAB، `uint8` (اعداد صحیح 8 بیتی بین 0 تا 255) و `double` (اعداد حقیقی بین 0 تا 1) هستند. به صورت پیش فرض تصاویر با فرمت `uint8` خوانده می‌شوند اما به دلیل نبود مشکل



گرد شدن در هنگام ضرب و تقسیم، اجرای محاسبات بر روی تصاویر double ساده تر است. با استفاده از دستور im2double تصویر حاصل شده را به نوع double تبدیل کنید.

4-1-ج) با مطالعه راهنمای متلب برای دستور imhist، هیستوگرام شدت نور تصویر مذکور را رسم کنید.

4-1-د) با مطالعه راهنمای متلب برای دستور histeq، روش یکسانسازی هیستوگرام در تصویر را شرح دهید. به چه دلیل و در کجا از این روش استفاده می شود؟ این عمل را بر روی تصویر lena اعمال کنید و تصاویر ورودی و خروجی را مقایسه کنید.

4-1-ه) در بخش قبلی هیستوگرام تصویر حاصل را رسم کنید. چرا هیستوگرام بدست آمده به صورت یکنواخت در نیامده است؟ دلیل آن را شرح دهید.

### نویززدایی تصویر (image denoising)

در این بخش با دو فیلتر دیجیتال مهم یعنی میانگین (mean) و میانه (median) آشنا می شوید. این فیلترها برای کاهش انواع خاصی از نویز در تصویر استفاده می شوند. ایده فیلتر میانگین عوض کردن مقدار هر پیکسل تصویر با میانگین مقادیر همسایگی شامل خود آن است. تاثیر این کار حذف مقادیر پیکسل هایی است که در اطراف وجود ندارند. فیلترینگ میانگین معمولاً به عنوان فیلتر کانولوشن در نظر گرفته می شود. مانند دیگر کانولوشن ها این فیلتر بر مبنای یک هسته (kernel) می باشد که شکل و سائز همسایگی را هنگام محاسبه میانگین انتخاب می شود را مشخص می کند. همان طور که در شکل زیر نشان داده شده است معمولاً از یک هسته مربعی  $3 \times 3$  استفاده می شود، اگرچه در هموار سازی شدید هسته های بزرگ تر (مثلاً  $5 \times 5$ ) نیز استفاده می شوند.

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

4-2-الف) تصویر با نام 'Image02.jpg' را بارگذاری کنید.

4-2-ب) یک نویز گوسی با میانگین صفر و انحراف استاندارد ( $\sigma$ ) برابر 0.2 را به آن اضافه کنید. می توانید از دستور imnoise متلب استفاده کنید. نتیجه را نمایش دهید.

4-2-ج) فیلتر میانگین  $3 \times 3$  را اعمال کرده و خروجی را نمایش دهید. در مورد نتیجه و پارانترهای نویز گوسی حاضر در تصویر بحث کنید.

4-2-د) فیلتر میانگین  $5 \times 5$  را روی آن اعمال کنید و تصویر نتیجه را نمایش دهید. تفاوت بین این قسمت و قسمت قبل چیست؟ آیا یک بده‌بستان دارید؟ اگر پاسخ مثبت است، آن چیست؟

4-2-ه) تصویر اصلی را در نظر بگیرید. این بار یک نویز نمک و فلفل (salt and pepper) با  $p=10\%$  با استفاده از دستور imnoise به آن اضافه کنید و تصویر خراب را نمایش دهید.

4-2-و) یک فیلتر میانگین  $3 \times 3$  به این تصویر خراب اعمال کرده و نتیجه را نمایش دهید. آیا این فیلتر برای کاهش نویز نمک و فلفل مفید است؟ چرا؟

4-2-ز) با استفاده از fdatool یک فیلتر پایین‌گذر FIR طراحی کنید که فرکانس عبور آن حدود فرکانس نرمالیزه 0.5 باشد. سپس با استفاده از دستور ftrans2 فیلتر یک بعدی طراحی شده را به یک فیلتر دو بعدی تبدیل کنید. سپس با استفاده از دو دستور freqz و freqz2 پاسخ فوریه فیلترهای یک بعدی و دو بعدی طراحی شده را با یکدیگر مقایسه کنید.

4-2-ح) فیلتر طراحی شده در بخش قبلی را به دو تصویر حاوی نویز اعمال کنید. نتایج حاصل از این فیلتر را با فیلتر میانگین مقایسه کنید. علت این تفاوت چیست؟

همان‌طور که در بخش قبل دیدید فیلترینگ میانگین در کاهش نویز نمک و فلفل فایده‌ای ندارد، بنابراین ما از یک فیلتر دیگر استفاده می‌کنیم، فیلتر میانه. این فیلتر مانند فیلتر میانگین هر پیکسل تصویر را به نوبت در نظر می‌گیرد و به همسایگی اطراف آن نگاه می‌کند تا تصمیم بگیرد که آیا آن نماینده اطرافش هست یا نه. به جای تعویض مقدار پیکسل با میانگین مقادیر پیکسل‌های همسایگی، آن را با مقدار میانه عوض می‌کند. میانه با ابتدا مرتب کردن عددی همه‌ی مقادیر پیکسل‌های همسایگی اطراف و سپس تعویض پیکسل مورد نظر با مقدار پیکسل میانی محاسبه می‌شود (در صورتی که همسایگی مورد نظر تعداد زوجی پیکسل داشته باشد، از میانگین دو عدد میانی استفاده می‌شود). شکل زیر یک محاسبه‌ی نمونه را نشان می‌دهد:

123	125	126	130	140
122	124	126	127	135
118	120	150	125	134
119	115	119	123	133
111	116	110	120	130

مقادیر همسایگی:

115, 119, 120, 123, 124,

125, 126, 127, 150

مقدار میانه: 124

4-2-ط) تابعی برای پیاده‌سازی میانه بنویسید. تابع شما باید تصویر اصلی و سایز پنجره که طول و عرض آن فرد می‌باشد را به عنوان ورودی بگیرد و تصویر اصلاح‌شده را به عنوان خروجی بدهد و همچنین در صورتی که طول و یا عرض وارد شده عدد فردی نبود پیغام خطا ظاهر شود.

4-2-ک) فیلتر خود را با سایز 3 به تصویر نویزی قسمت 4-2-ه اعمال کنید. نتیجه را نمایش دهید. چگونه این فیلتر می‌تواند به شما کمک کند تا بر نویز نمک و فلفل غلبه کنید؟

4-2-ل) اثر blurring فیلترهای میانه و میانگین را مقایسه کنید. آیا فیلتر میانه بدی‌هایی نیز دارد؟

## تبدیل موجک دوبعدی

در این بخش شما استفاده از تبدیل موجک دوبعدی را تمرین خواهید کرد.

4-3-الف) 'Image03.png' را بارگذاری کرده و با استفاده از 'db1' به عنوان نام موجک، تبدیل موجک را به این تصویر اعمال کنید و سپس تصاویر تقریبی و جزئیات را نمایش دهید. می‌توانید از تابع متلب dwt2 که تبدیل موجک گسسته دوبعدی می‌باشد استفاده کنید. ورودی‌ها و خروجی‌های این تابع چه می‌باشند؟ هر زیر تصویر حاوی چه اطلاعاتی است؟

4-3-ب) اکنون در نظر بگیرید که ما می‌خواهیم خطوط افقی را highlight کنیم. شما چه پیشنهادی دارید؟ آن را اعمال کنید.

## تارشدگی (Motion Blurring)

یکی از مشکلات عکاسی تار شدن تصویر است. این زمانی اتفاق می‌افتد که جسم در زمان باز بودن شاتر دوربین دیجیتال حرکت کند. ما می‌توانیم با علم به الگوی جسم آن را اصلاح کنیم.

4-4-الف) 'Image04.png' را بارگذاری کرده و نمایش دهید. با استفاده از توابع داخلی متلب یک تاری (motion blur) به اندازه 15 و با زاویه  $20^\circ$  به تصویر اعمال کنید. تصویر تار شده را نمایش دهید.

4-4-ب) تصویر تار شده را با فیلتر wiener اصلاح کنید. شما می‌توانید از تابع deconvwnr استفاده کنید. برای این تابع به Help متلب مراجعه کنید. شما باید پارامتر NSR (Noise to Signal Power Ratio) را تغییر دهید تا به یک نتیجه منطقی برسید. ابتدا NSR را صفر بگیرید و حاصل را نمایش دهید، سپس با تکرار، مقادیر مناسبی برای NSR انتخاب کنید.

4-4-ج) اکنون یک نویز گوسی با میانگین صفر و واریانس 10 به تصویر تار قسمت 4-4-الف اضافه کنید. تصویر تار نویزی را نمایش دهید.

4-4-د) قسمت 4-4-ب را برای تصویر قسمت 4-4-ج تکرار نمایید.

## فیلتر Antialiasing

4-5-الف) فایل با نام 'glass.tif' را بارگذاری کرده و نمایش دهید.

4-5-ب) DFT دوبعدی تصویر را با تابع fft2 حساب کنید. فاز و لگاریتم دامنه DFT دوبعدی را در تصاویر مجزا نمایش دهید. از دستور fftshift استفاده کرده و توضیح دهید که این دستور چه کاری انجام می‌دهد.  
توجه: فوریه دوبعدی به صورت زیر تعریف می‌شود:

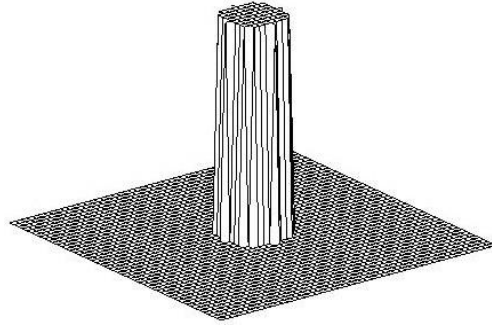
$$F(u, v) = \sum_m \sum_n f(m, n) e^{j2\pi(mv + nu)}$$

4-5-ج) تابع زیر را که فیلتر پایین‌گذر دوبعدی را پیاده‌سازی می‌کند در نظر بگیرید، تعریف کنید.

Output\_image = FFT\_LP\_2D(input\_image, cutoff\_frequency)

فرکانس قطع باید در بازه  $[0, \pi]$  باشد. Ifft2 را اعمال کرده تا معکوس DFT دوبعدی را حساب کنید.

فیلتر باید به فرم دایروی (Circular) باشد. به عبارت دیگر شعاع دایره باید فرکانس قطع باشد.



به شما توصیه می‌شود که از توابع `fftshift`، `ifftshift` و `meshgrid` استفاده کنید. همچنین فراموش نکنید که از گزینه 'symmetric' در `ifft` استفاده کنید. همچنین به بازه‌ای که DFT دوبعدی در آن حساب می‌شود توجه داشته باشید.

4-5-د) فیلتر پایین‌گذر با فرکانس قطع  $0.1\pi$  را روی تصویر قسمت 4-5-الف اعمال کرده و نتیجه را نمایش دهید. بر اساس آنچه تا کنون در مورد فیلترهای پایین‌گذر و بالاگذر آموخته‌اید، در مورد عملکرد این فیلتر بحث کنید.

4-5-ه) از تابع `downsample` به درستی استفاده کنید و ابعاد این تصویر را به  $1/4$  تصویر اصلی کاهش دهید. خروجی را نمایش دهید و تاثیر `aliasing` را در تصویر `downsampled` مشاهده کنید. `aliasing` گوشه‌های تصویر یا قسمت‌های راه راه را خراب می‌کند. بر اساس دانش خودتان از پردازش سیگنال یک‌بعدی، از فیلتری که تعریف کرده‌اید استفاده کنید و تصویر را به اندازه کافی تغییر دهید تا اثرات `aliasing` را در تصویر 'downsampled' حذف کنید. خروجی را نمایش دهید.

بخش دوم:

آزمایش‌های مربوط به پیاده‌سازی نرم‌افزاری

# **پردازش سیگنال مبتنی بر سخت افزار FPGA**

در جلسات مرتبط با FPGA هدف آشنایی با ساختار FPGA و نحوه پیاده سازی الگوریتم های مخابراتی و پردازش سیگنالی بر روی سخت افزار FPGA می باشد. با توجه به گسترده بودن این مبحث و تنوع ابزارهای متفاوت برای پیاده سازی این الگوریتم ها در سخت افزار FPGA، از سیستم ژنراتور به عنوان ابزاری برای FPGA Programming استفاده خواهیم کرد. سیستم ژنراتور بهترین ابزار برای مهندسين مخابرات و مهندسين الگوریتم است که کمک می کند در کوتاه ترین زمان الگوریتم پردازشی مد نظر را با بکارگیری مقدار مناسبی از منابع موجود در FPGA هدف، پیاده سازی کرد. ازین رو برای آزمایشگاه پردازش سیگنال این ابزار انتخاب شده است.

System Generator برای DSP یک ابزار طراحی پیشرو در معماری صنعتی برای تعریف، آزمایش و اجرای الگوریتم های DSP با کارایی بالا بر روی دستگاه های Xilinx است. که به عنوان یک جعبه ابزار اضافی برای MathWorks Simulink® طراحی شده است. سیستم ژنراتور برای DSP از IP های موجود از قبل بهینه شده برای FPGA استفاده می کند. سیستم ژنراتور برای مدل های مبتنی بر DSP با یک محیط تشکیل شده از تعداد زیادی ابزار شبیه سازی و تست توسط Simulink® ارائه شده است. با بکارگیری System Generator می توان الگوریتم های DSP با کیفیت مناسب جهت تولید را در زمان کمتری در مقایسه با زمان های توسعه RTL سنتی ایجاد کرد.

مهمترین ویژگی های که سیستم ژنراتور به طراحان الگوریتم می دهد می توان به موارد زیر اشاره کرد:

- بیش از 100 بلوک RTL بهینه سازی شده به همراه تعداد زیادی مدل شبیه سازی شده مبتنی بر C برای شبیه سازی های 2 تا 3 برابر سریع تر نسبت به RTL
- در کنار هم قرار داشتن Xilinx IP، سیمولینک RTL و ابزارهای متلب برای سیستم های DSP
- شبیه سازی های fixed-point و floating-point دقیق در بیت و کلاک
- شبیه سازی همزمان به منظور سرعت بخشی شبیه سازی و ارزیابی الگوریتم بر روی سخت افزار
- تولید اتوماتیک کد از سیمولینک به IP یا Low-level HDL
- تولید اتوماتیک تست HDL، شامل بردارهای تست و شبیه سازی

از طرفی اگر بخواهیم مهمترین نکات سخت افزاری استفاده از System generator را بیان کنیم می توان موارد زیر را بیان داشت:

- ایجاد و رفع باگ سیستم های DSP با کارایی-بالا با بکارگیری Xilinx-optimized RTL IPs با سیمولینک برای پردازش سیگنال (نظیر FIR filter و FFTs)، جبران خطا (مانند Viterbi decoder، Reed-Solomon encoder/decoder)، محاسبات ریاضیاتی، کنترل حافظه ها (مانند



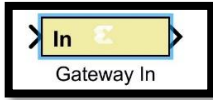
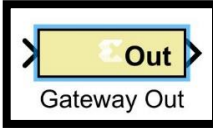
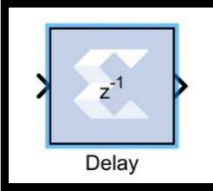
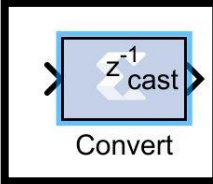
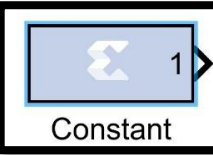
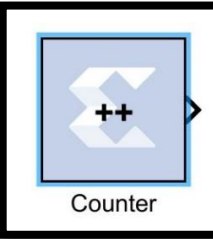
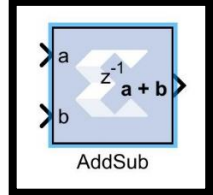
FIFO, RAM و ROM) و عملگرهای منطقی. دسترسی و کنترل تمامی DSP48ها با سیمولینگ برای طراحی های با نرخ بالا

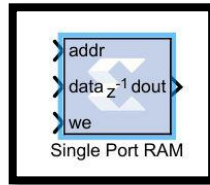
- پیاده سازی بهینه و دقیق در دو حالت fixed & floating – point سیستم ژنراتور از تمامی داده های single, double و انواع مدل های floating-point شخصی سازی شده پشتیبانی می کند.
  - تولید کد VHDL یا Verilog یا Packaged IP از سیمولینگ و Xilinx IP cores برای Xilinx Blockset (کتابخانه اصلی Xilinx در محیط سیستم ژنراتور)
  - شبیه سازی سخت افزاری هم زمان سیستم ژنراتور به ما این قابلیت را می دهد که برای ارزیابی سیستم طراحی شده برای سخت افزار آن را به صورت بلادرنگ با کمک Ethernet (10/100/Gigabit) و JTAG آزمایش نماییم.
  - تحلیل زمانی و منابع مصرفی ارزیابی زمانی و منابع بکارگرفته شده برای طراحی خود را می توان با بکارگیری ابزارهای مربوط ( Post-synthesis و Post-Implementation ) موجود در سیستم ژنراتور با سرعت بسیار بالا انجام داد. همچنین این ابزارها در یافتن خطاهای زمانی بسیار سریع و کارآمد هستند.
- دستگاه های مورد پشتیبانی توسط سیستم ژنراتور:

Kintex®-7, Virtex®-7, Zynq®-7000, Artix®-7, Kintex UltraScale™, Kintex UltraScale+, Virtex UltraScale, Virtex UltraScale+, Zynq UltraScale+ RFSoc

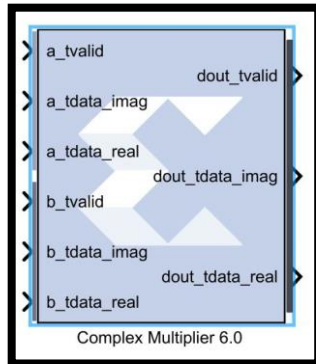
## آزمایش اول: آشنایی با بلوک های مشهور در سیستم ژنراتور و راه اندازی چند بلوک ساده در این محیط.

بلوک های مشهور:

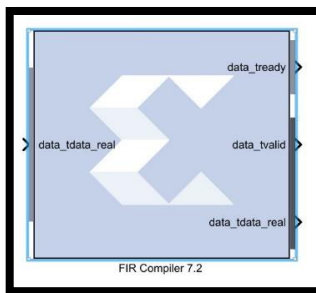
	<p>این بلوک مقادیر integer، double و Fixed point محیط سیمولینک را به مقادیر Fixed-point سیستم جنراتور تبدیل می کند.</p>
	<p>عملکرد برعکس بلوک Gateway-in دارد.</p>
	<p>بلوک تاخیر</p>
	<p>بلوک تبدیل کننده مقادیر Fixed-point</p>
	<p>بلوک مقدار ثابت</p>
	<p>بلوک شمارشگر</p>
	<p>بلوک جمع-تفریق کننده</p>



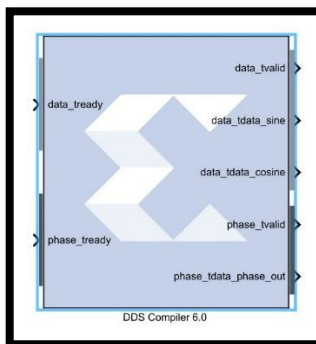
بلوک حافظه رم



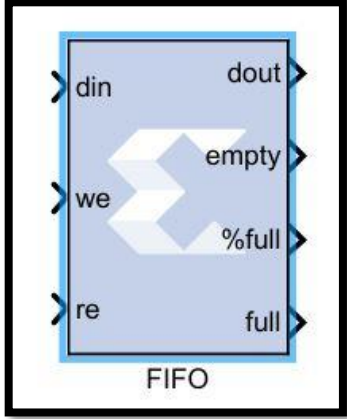
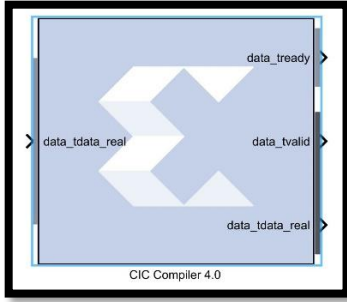

بلوک ضرب کننده مختلط



بلوک فیلتر FIR



بلوک تولید کننده سیگنال سینوسی

 <p>The diagram shows a FIFO (First In First Out) block. It has three input ports on the left: 'din', 'we' (write enable), and 're' (read enable). It has three output ports on the right: 'dout', 'empty', and '%full'. The label 'FIFO' is centered below the block.</p>	<p>بلوک حافظه فایفو</p>
 <p>The diagram shows a CIC (Cascaded Integrator-Cascade) filter block. It has two input ports on the left: 'data_tdata_real' and 'data_tdata_real'. It has two output ports on the right: 'data_tready' and 'data_tvalid'. The label 'CIC Compiler 4.0' is centered below the block.</p>	<p>بلوک فیلتر CIC</p>
 <p>The diagram shows the System Generator logo, which consists of a stylized 'X' shape made of red and white geometric blocks. Below the logo, the text 'System Generator' is written.</p>	<p>بلوک توکن</p>

در این بخش پس از توضیحات کلی در مورد بلوک ها، توضیحاتی در مورد راه اندازی بلوک های ساده جمع کننده و ضرب کننده ارائه می شود و نحوه ارتباط محیط سیستم جنراتور با محیط MATLAB توضیح داده می شود.

## آزمایش دوم: راه اندازی بلوک فیلتر CIC و FIR

در این بخش دو بلوک CIC و FIR راه اندازی می شود. پس از توضیحات کلی در مورد این بلوک ها و منطق استفاده کردن از آنها تک تک تب های موجود در رابط کاربری این بلوک ها توضیح داده می شود. پس از این توضیحات یکی از پروژه های ابتدای دستور کار با استفاده از این بلوک ها پیاده می شود.

هنگاه فرایند پیاده سازی از دانشجویان خواسته می شود تا پاسخ های گرفته شده از پیاده سازی MATLAB و پیاده سازی سیستم ژنراتور را با هم مقایسه کنند. بدیهی است که طراحی باید به گونه ای باشد که نتایج حاصله با دقت خیلی خوبی در حوزه زمان شبیه به هم باشند.

یکی از مسائل خیلی مهم در پیاده سازی عملی مبحث Fixed-Point می باشد که در این جلسه به صورت کامل برای دانشجویان تشریح داده می شود. تا در ادامه ی طراحی ها به این مساله دقت شود.

## آزمایش سوم: راه اندازی بلوک حافظه

هدف از انجام این آزمایش آشنایی با بلوک های حافظه می باشد. در این آزمایش دو بلوک حافظه FIFO و RAM راه اندازی می شود. پس از توضیحات کلی در مورد این بلوک ها و منطق استفاده کردن از آنها تک تک تب های موجود در رابط کاربری این بلوک ها توضیح داده می شود.

از دانشجوین خواسته می شود که یک سری داده بر روی بلوک حافظه RAM نوشته شود و به ترتیب دلخواه خوانده شود. همچنین از دانشجوین خواسته می شود که یک تاخیر به طول 50 با استفاده از بلوک FIFO طراحی شود.

در این جلسه در کنار طراحی های انجام شده توضیحاتی در مورد محیط ویوادو داده می شود. و فرایند تولید Bit stream به طور مختصر توضیح داده می شود.

