

به نام خدا



درس : سیگنال‌ها و سیستم‌ها

استاد: دکتر کربلایی

گزارش پروژه شماره ۴

امین زمانی

۹۴۱۰۰۷۸۷

سید محمد امین منصوري

۹۴۱۰۵۱۷۴

قسمت اول: طراحی فیلتر گستته

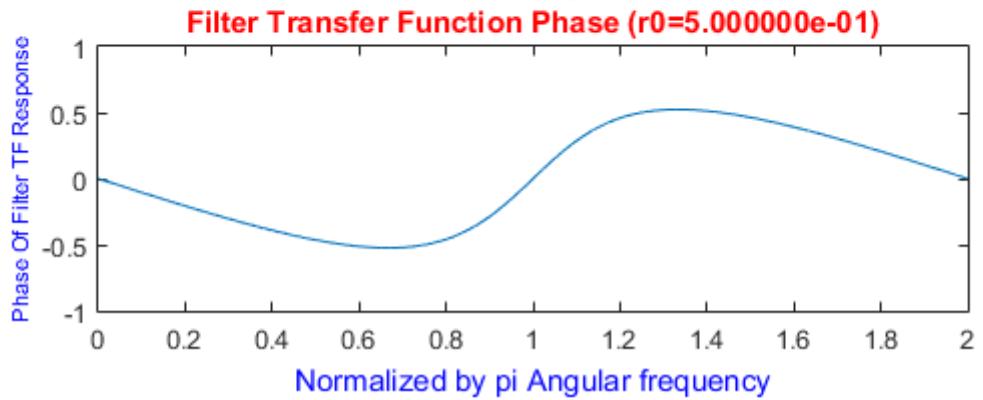
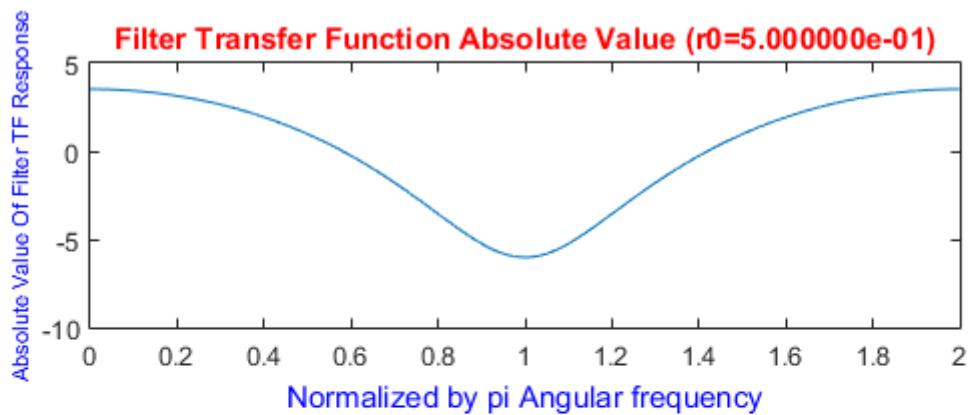
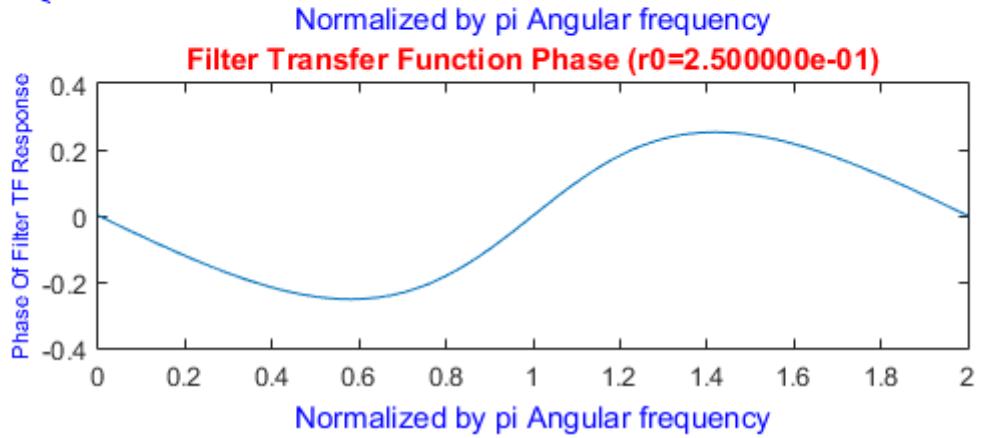
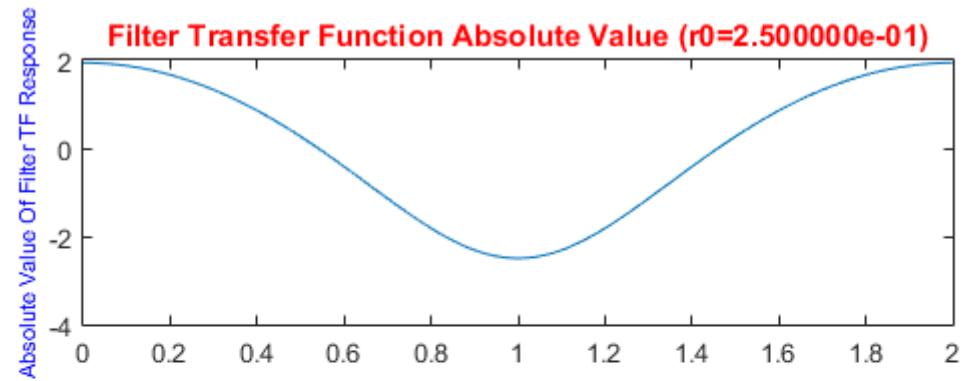
۱. توضیح کوتاه راجع به هر قسمت: دستور *freqZ* در یکی از ورودی خود تعداد نقطه‌هایی که در آن تابع انتقال محاسبه می‌شود را می‌گیرد. در دو ورودی دیگر ضرایب صورت و مخرج گرفته می‌شود. دستور *filter* سیگنال ورودی و ضرایب صورت و مخرج فیلتر را گرفته و در خروجی سیگنال فیلتر شده را تحويل می‌دهد. دستور *butter* نیز فیلتر آنالوگ و دیجیتال با ترورث می‌سازد. دستور *tf* نیز ضرایب صورت و مخرج را گرفته و تابع انتقال سیستم پیوسته در زمان معادل آن را در شیءی از جنس *sys* ذخیره می‌کند.

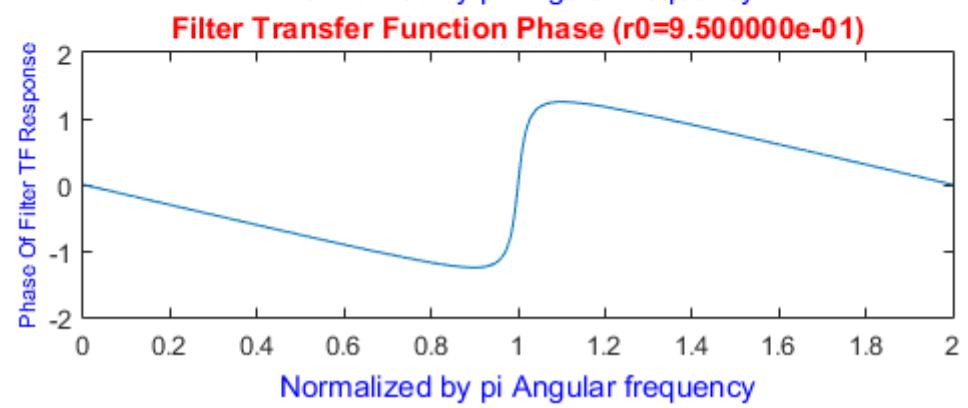
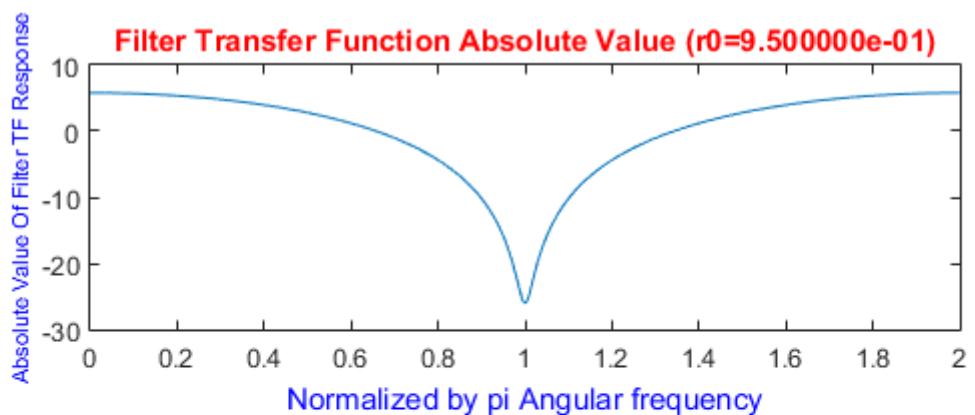
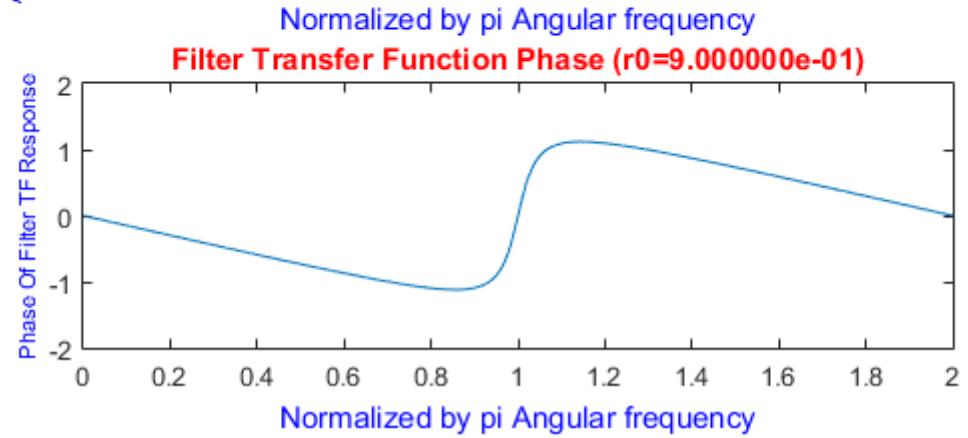
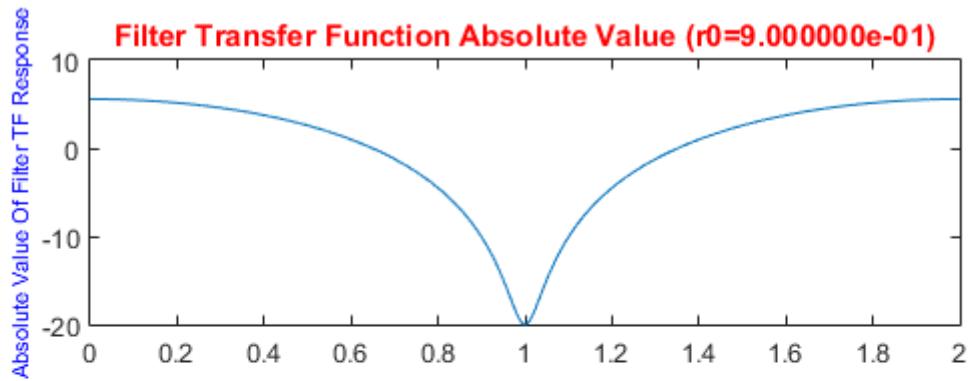
۲. همانطور که می‌دانیم برای ساخت فیلتر *FIR* نباید به جز در مبدا قطبی داشته باشیم. با توجه به این که سوال هیچ اشاره‌ای به ویژگی‌های مورد نیاز فیلتر نداشته، ما ساده‌ترین فیلتر پایین‌گذر را طراحی می‌کنیم. برای این منظور از یک صفر تنها در نزدیکی بیشینه فرکانس (π) استفاده می‌کنیم. همچنین آن را روی محور افقی انتخاب می‌کنیم. به این صورت تابع تبدیل به صورت زیر در می‌آید:

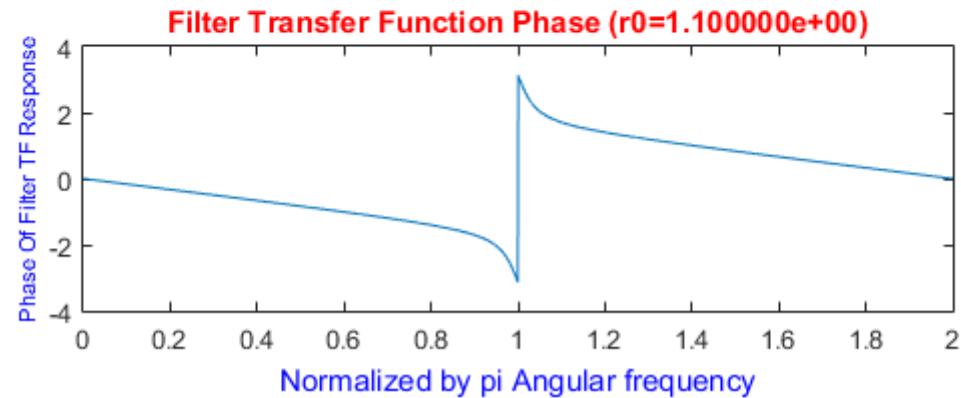
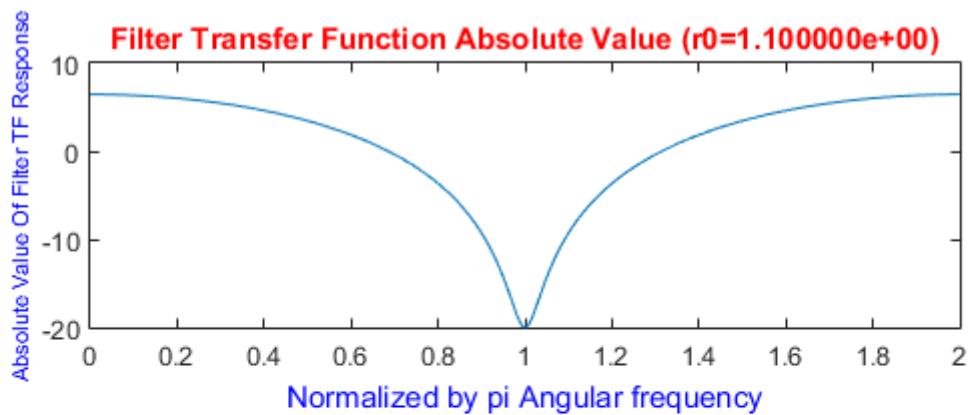
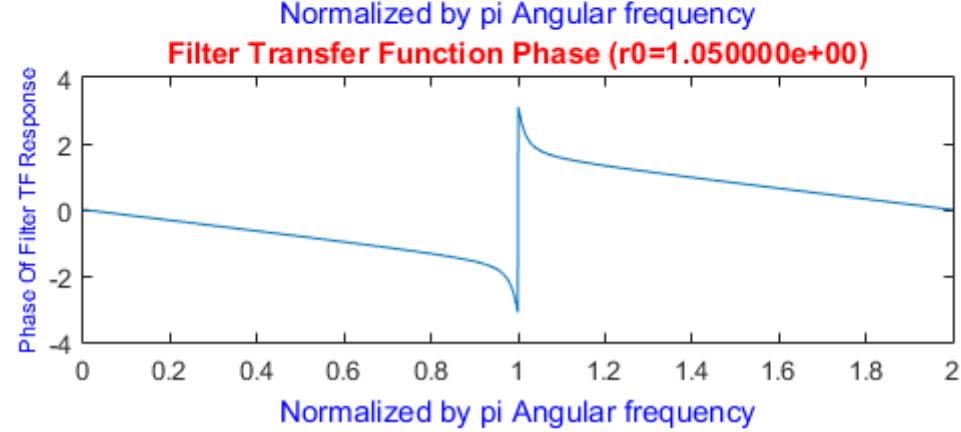
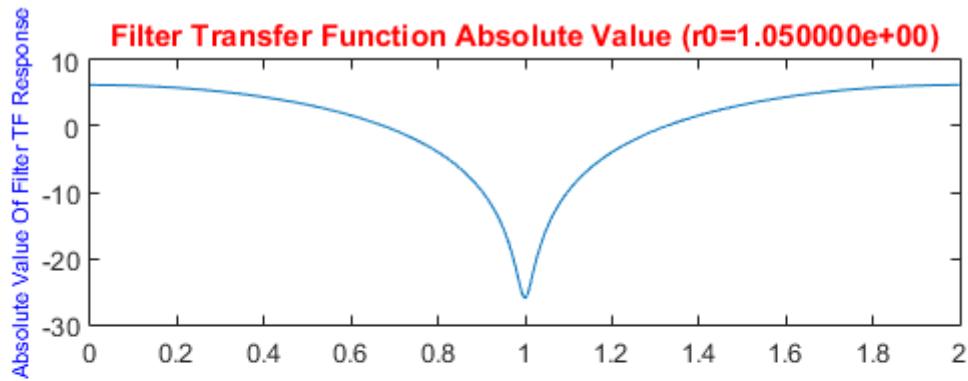
$$H(e^{j\omega}) = e^{j\omega} - r_0$$

(دقت می‌کنیم که مقادیر r_0 منفی هستند.)

۳. نتایج در تصاویر زیر مشاهده می‌شود.

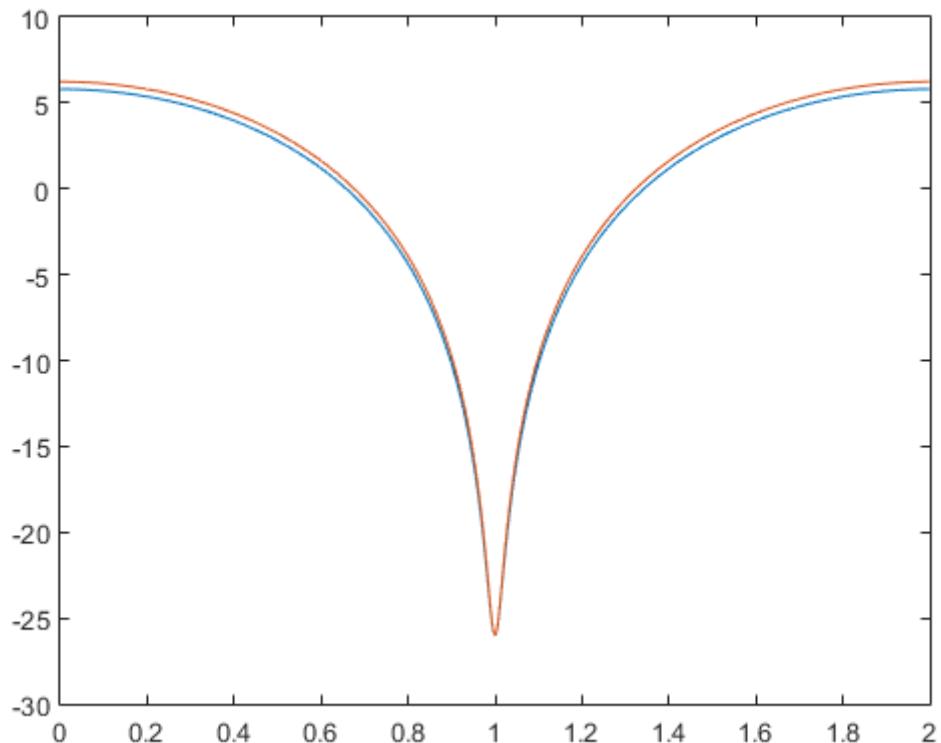


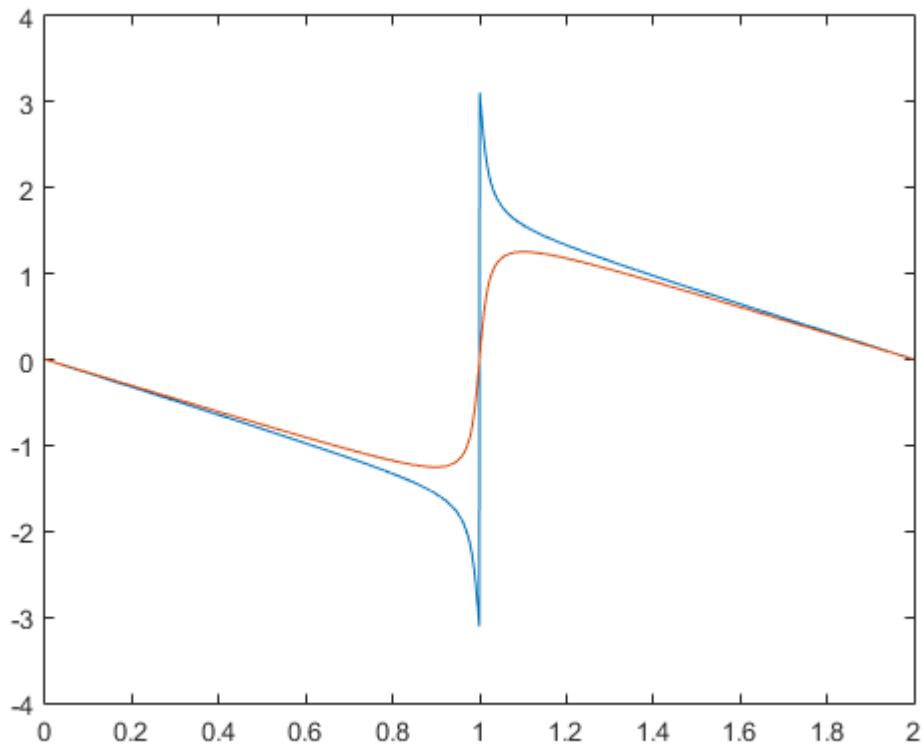




همان‌طور که مشاهده می‌شود در فیلتری که طراحی کردیم، با افزایش r_0 تا قبل از رسیدن به دایره واحد، پهنه‌ای باند گذر زیاد شده و شدت تضعیف فرکانس‌های بالا نیز زیادتر می‌شود.(به فیلتر *notch* نزدیک

می شود). پس از آن با خارج شدن از دایره واحد و دور شدن از آن شدت تضعیف فرکانس های بالا کم می شود. اثر دیگر پله ای شدن فاز پاسخ فرکانسی به ازای دور شدن از مبدا است. همچنین با توجه به این که در حالت $\tau_0 = 0.95$, 1.1 اختلاف با دایره واحد یکسان و بسیار کم است، پاسخ فرکانسی این دو حالت تفاوت چندانی ندارند و می توان اختلاف اندک آنها را در تصویر زیر مشاهده کرد. علت این تفاوت برای فرکانس های بالا پایین این است که برای این نقاط این اختلاف به صورت فاصله افقی ظاهر شده و برای فرکانس های بالا به صورت فاصله عمودی و در حالت اول این تفاوت بیشتر مشهود است. اما در اثر خارج شدن از دایره واحد اثر پله ای شدن فاز پاسخ فرکانسی به وضوح دیده می شود.



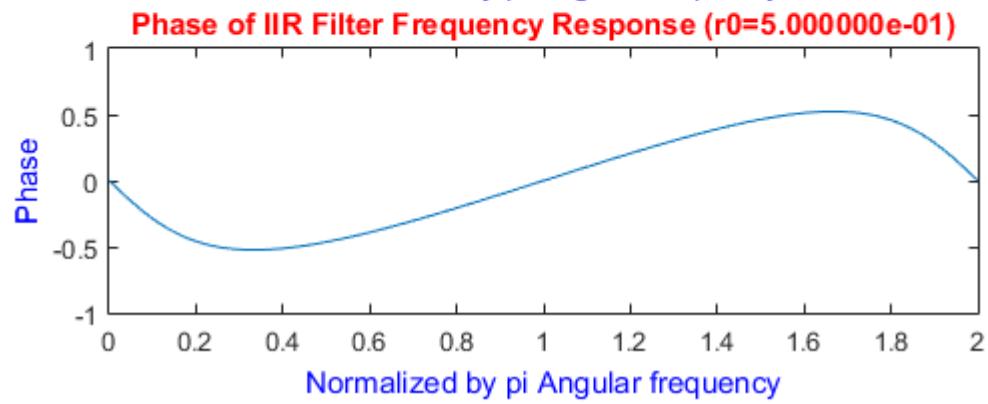
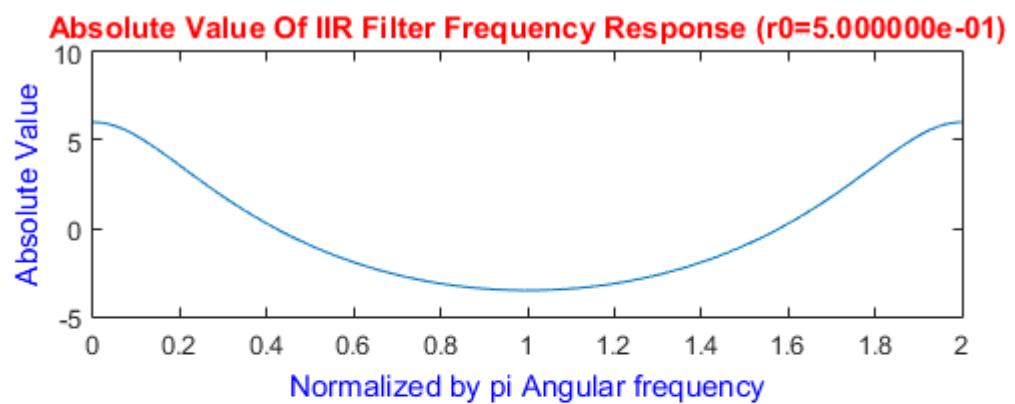
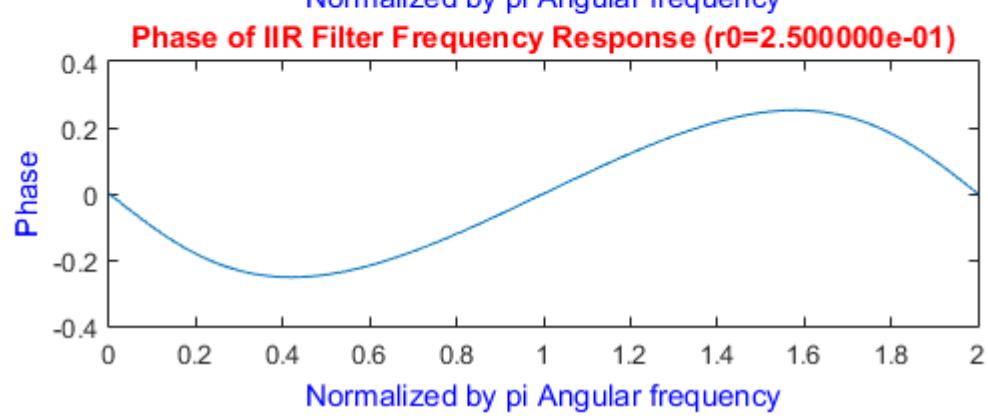
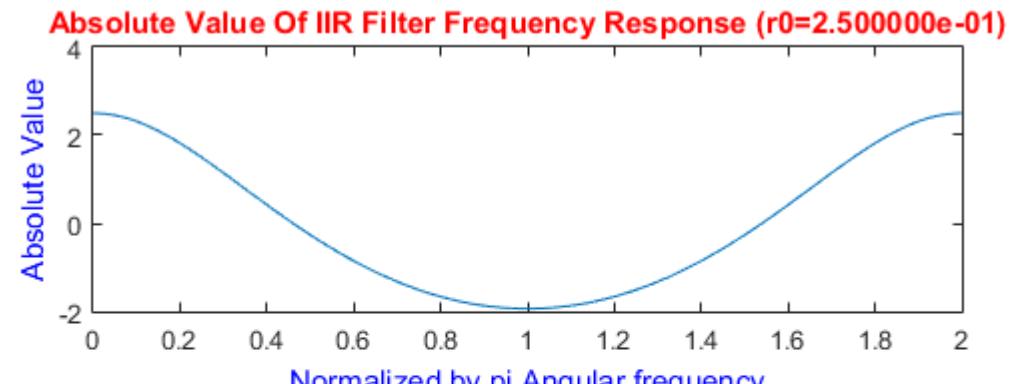


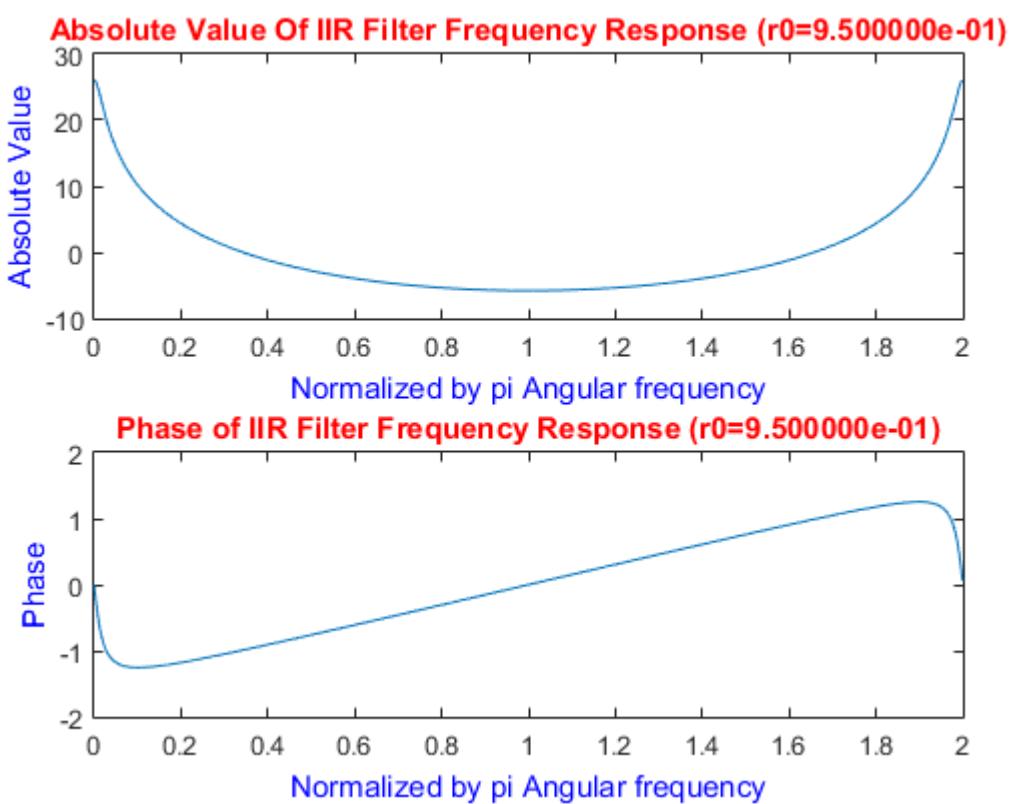
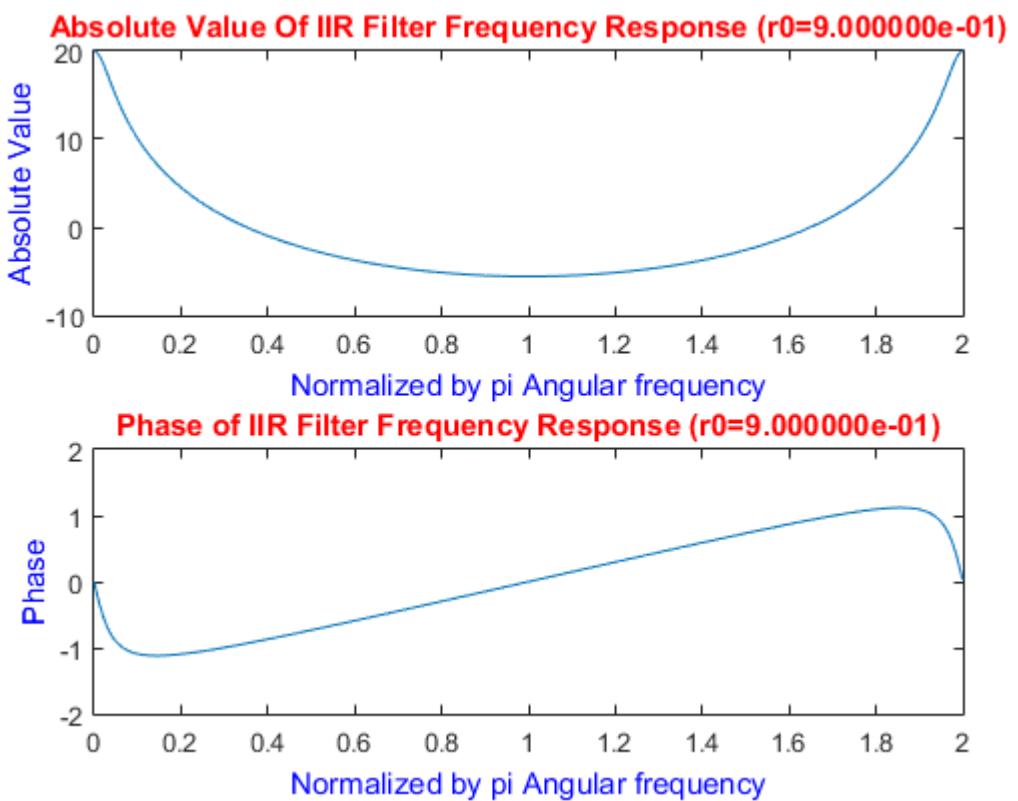
۴. برای ساخت فیلتر پایین‌گذر IIR می‌توان از قطب غیر از در مبدا نیز استفاده کرد. باز هم با توجه به ذکر نکردن سوال در مورد ویژگی‌های این فیلتر ما ساده‌ترین فیلتر از این دست را استفاده می‌کنیم. بدین صورت که یک قطب تنها در نزدیکی فرکانس‌های پایین قرار می‌دهیم. همچنان این قطب را روی محور افقی انتخاب می‌کنیم و تابع تبدیل به صورت زیر در می‌آید:

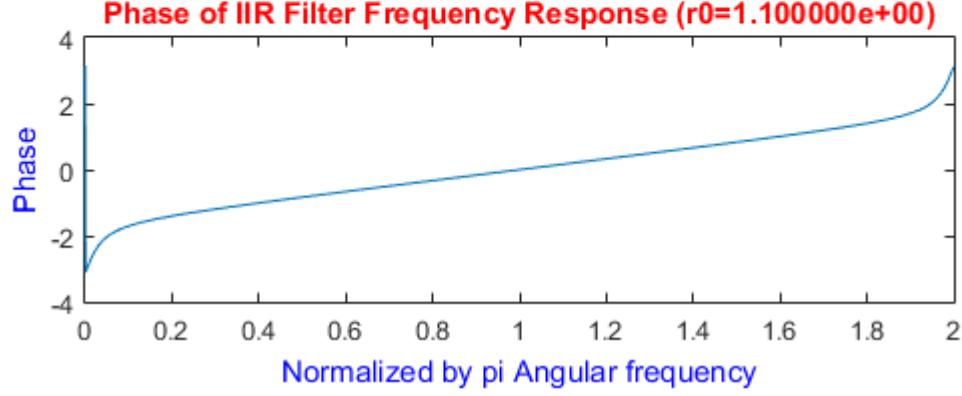
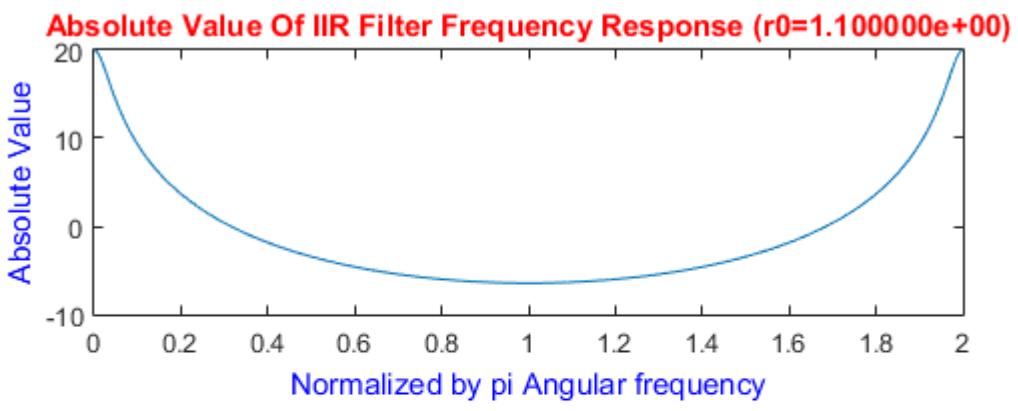
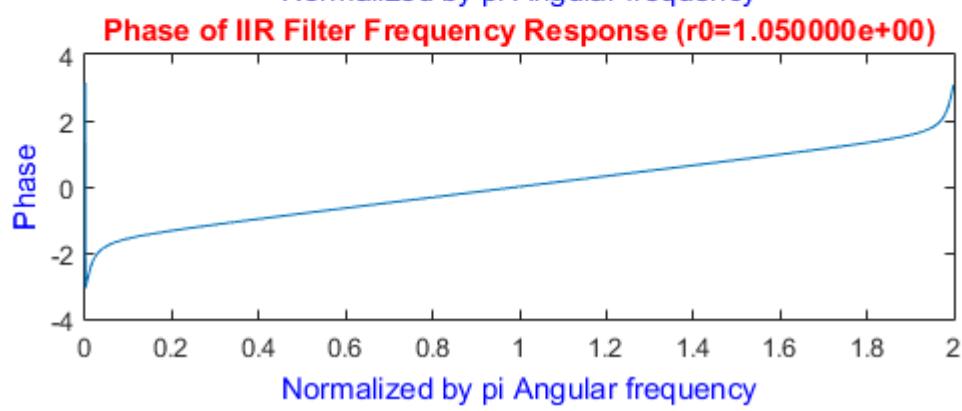
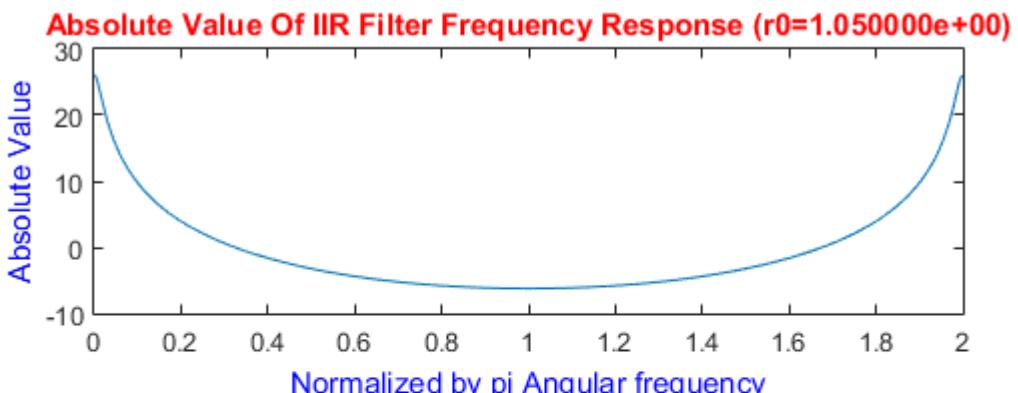
$$H(j\omega) = \frac{1}{e^{j\omega} - r_0}$$

(دقت می‌کنیم که مقادیر r_0 مثبت هستند).

۵. نتایج در تصاویر زیر مشاهده می‌شود.



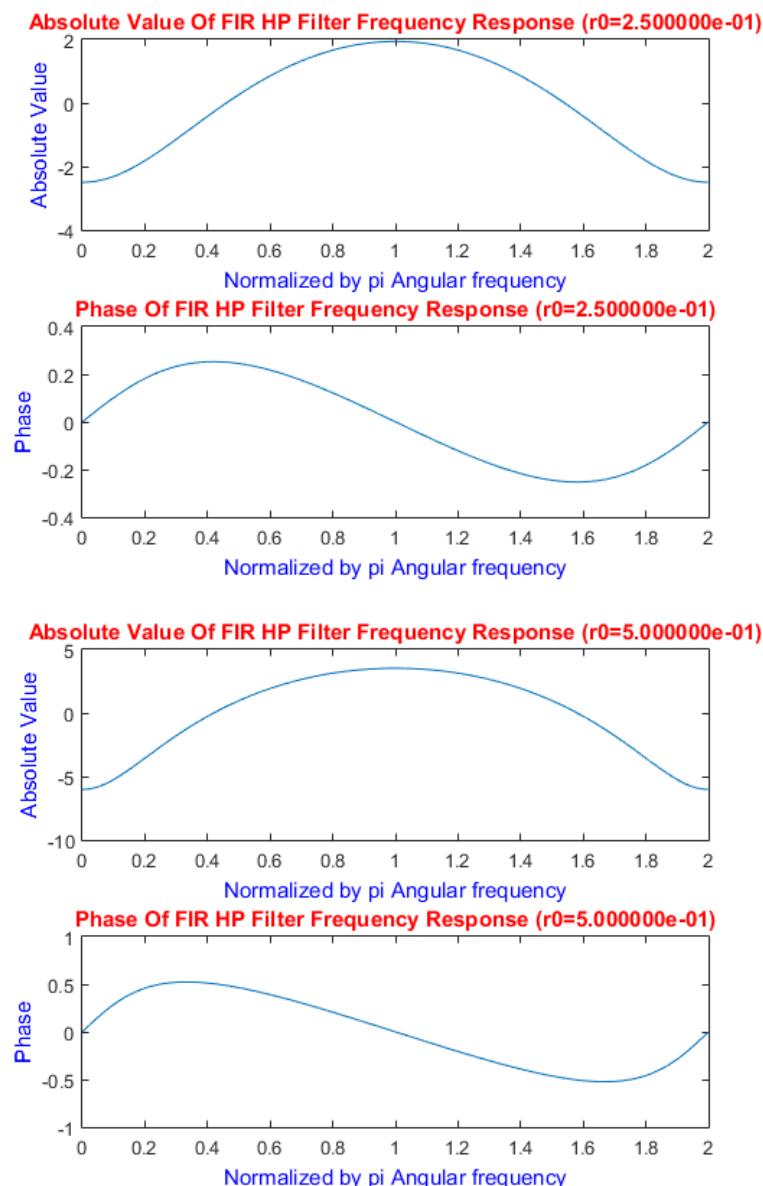


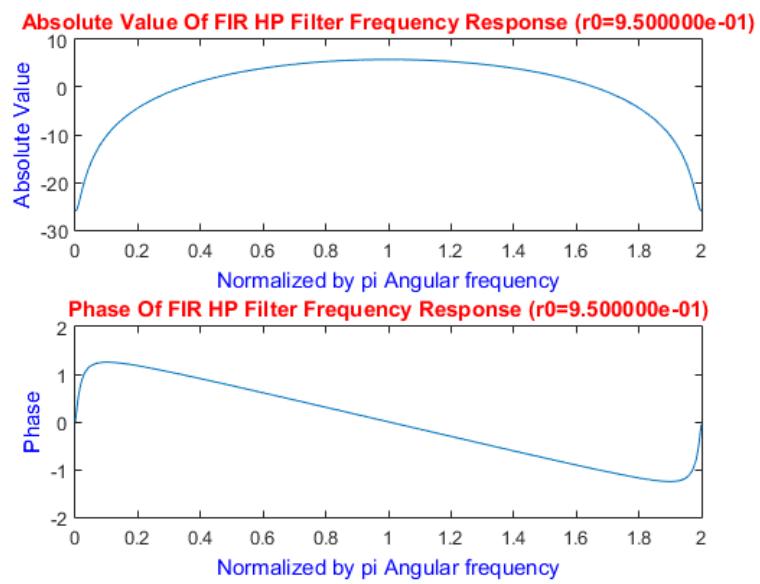
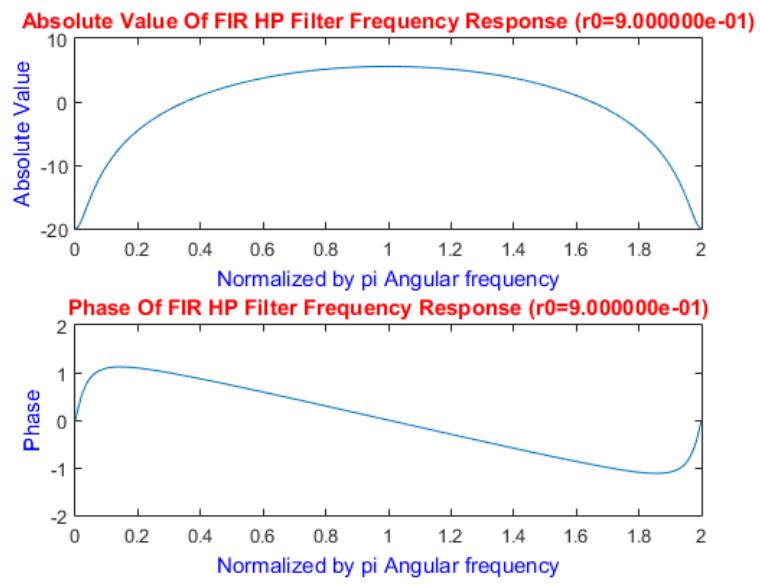


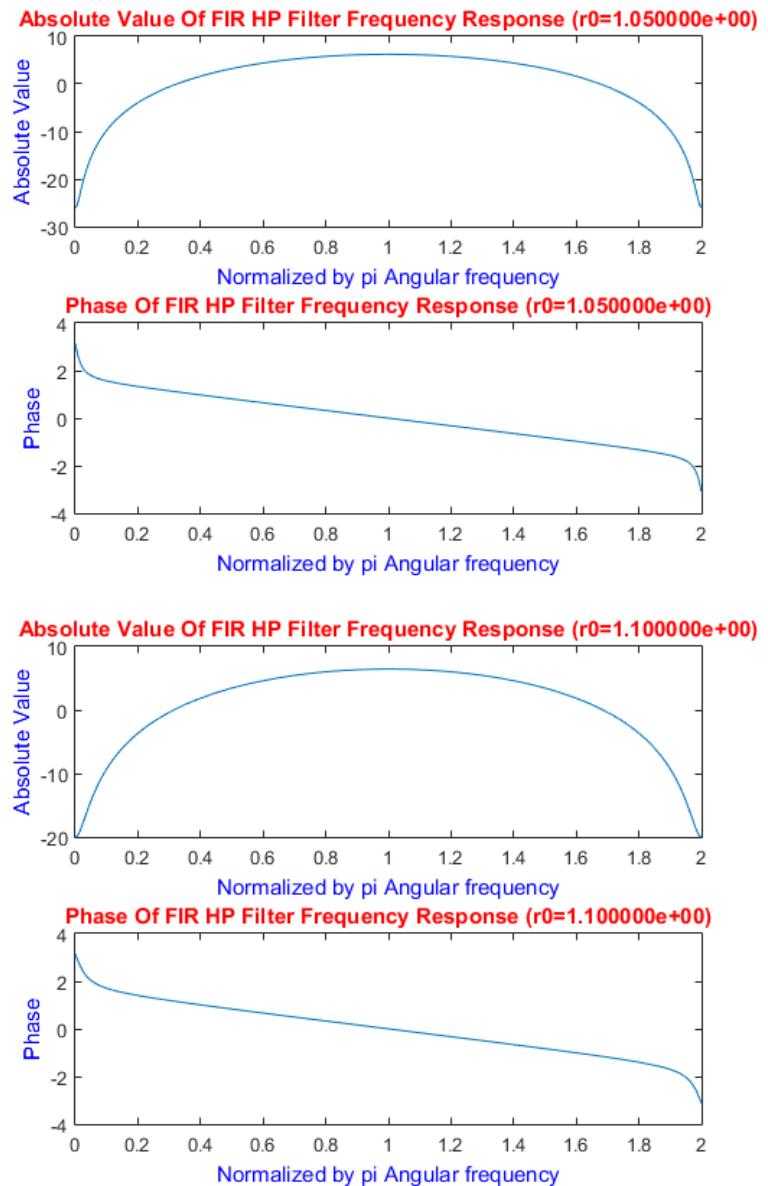
مشاهده می شود با دور شدن از مبدأ پهنهای باند گذر کم شده و فاز در ناحیه بیشتری خطی می ماند. در مجموع فیلترهای *IIR* در پهنهای باند کمتری سیگنال را عبور می دهند و فاز آن ها در بازه بزرگتری خطی است، اما

در نزدیکی فرکانس‌های پایین همان‌طور که مشاهده می‌شود به صورت پله‌ای تغییر می‌کنند. در واقع اگر با این فرکانس‌ها کار کنیم، در اطراف صفر تغییر ناگهانی فاز داریم (در تصاویر بالا ملاحظه می‌شود) و اگر با فرکانس‌های بالا کار کنیم در ناحیه بزرگی خطی می‌ماند. در طرف مقابل اگر از فیلتر FIR استفاده کنیم و در فرکانس‌های پایین کار کنیم، فاز در ناحیه بزرگی خطی است و تغییر ناگهانی در فرکانس‌های بالا رخ می‌دهد، نه در فرکانس‌های پایین. اما به علت پهنای باند زیاد خاصیت پایین‌گذری آن ممکن است در کاربردهای مطلوب نباشد.

۶. متناظر قسمت‌های ۲ و ۴: برای فیلتر بالاگذر در حالت FIR در نزدیکی فرکانس‌های پایین روی محور صفر تنها می‌گذاریم. در حالت IIR در نزدیکی فرکانس‌های بالا روی محور قطب تنها می‌گذاریم. متناظر ۳ و ۵:

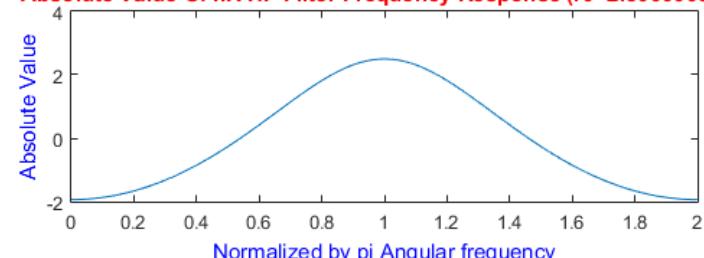




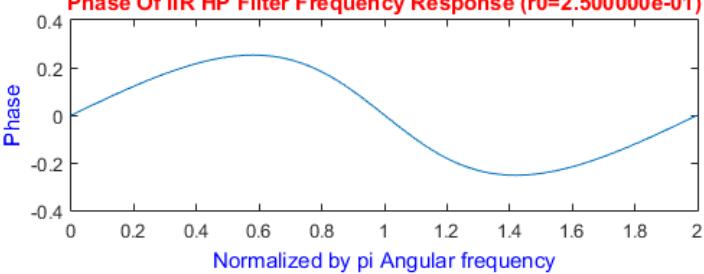


همان‌طور که در تصاویر بالا (FIR) مشاهده می‌شود، با دور شدن صفر از مبدا پهنه‌ای باند و ناحیه‌ای که در آن خروجی خطی است افزایش می‌یابد. در صورتیکه بخواهیم ناحیه خطی بیشتری داشته باشیم و همزمان در فرکانس‌های بالا کار کنیم، می‌توانیم از این فیلتر استفاده کنیم، اما باید دقیق داشته باشیم در ازای خطی بودن بیشتر، خاصیت فیلتری را از دست می‌دهیم و فرکانس‌های کمتری را حذف می‌کنیم.

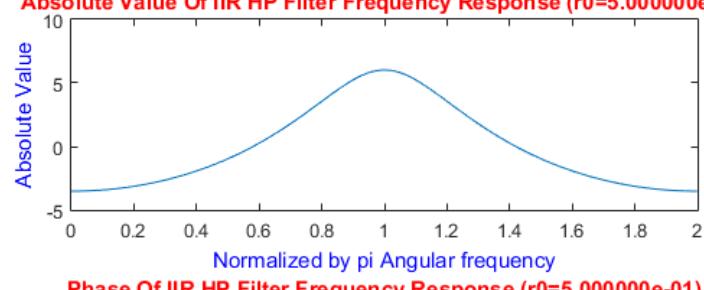
Absolute Value Of IIR HP Filter Frequency Response ($r=2.500000e-01$)



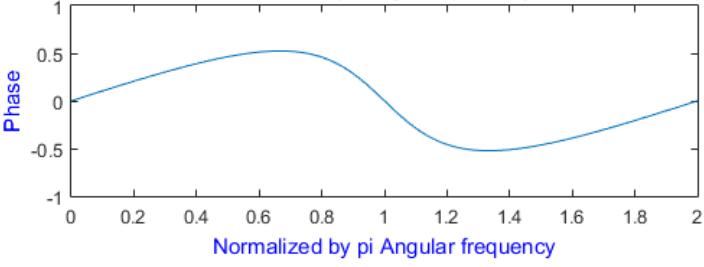
Phase Of IIR HP Filter Frequency Response ($r=2.500000e-01$)



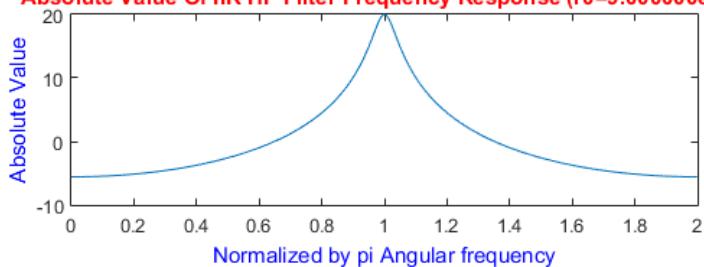
Absolute Value Of IIR HP Filter Frequency Response ($r=5.000000e-01$)



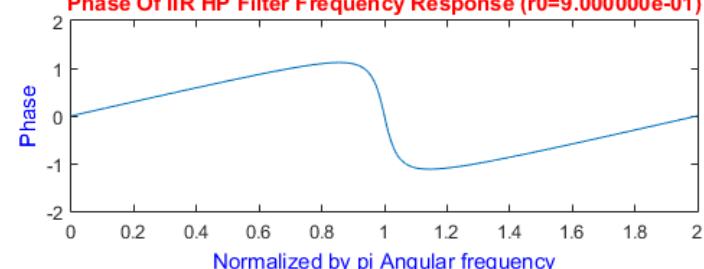
Phase Of IIR HP Filter Frequency Response ($r=5.000000e-01$)

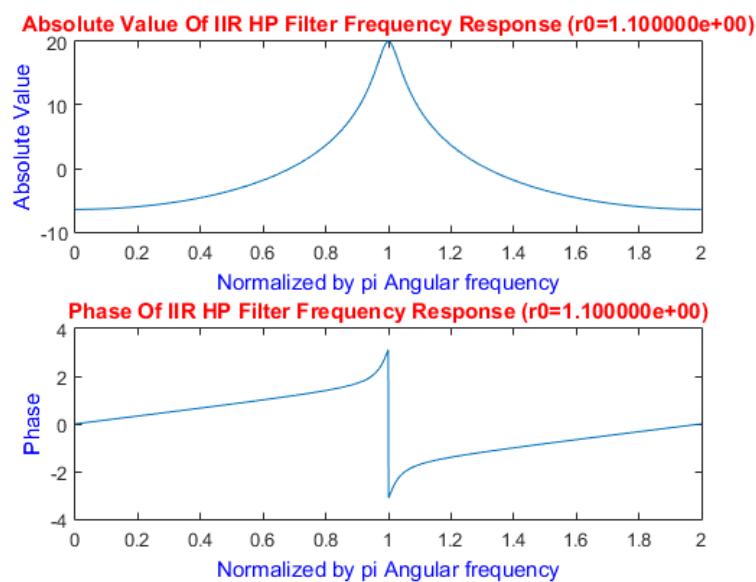
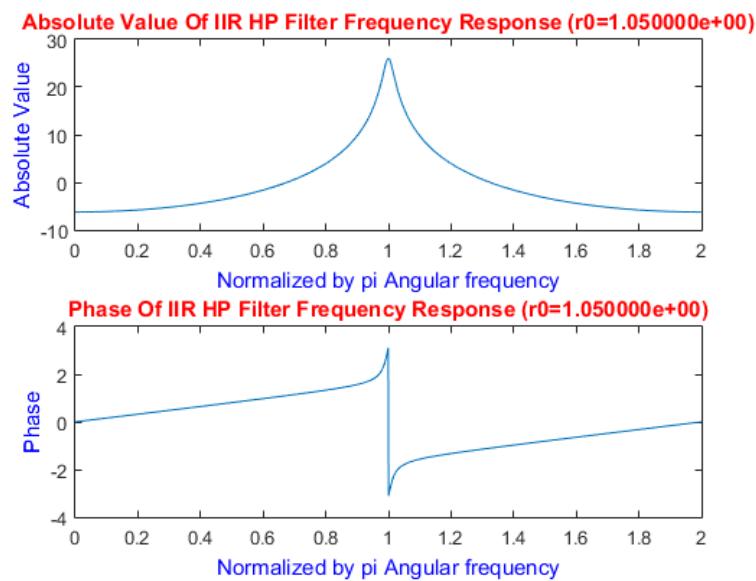
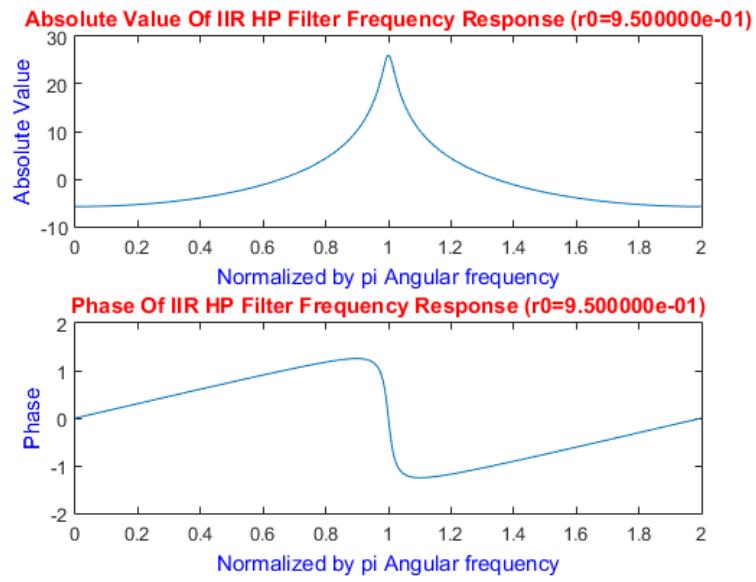


Absolute Value Of IIR HP Filter Frequency Response ($r=9.000000e-01$)



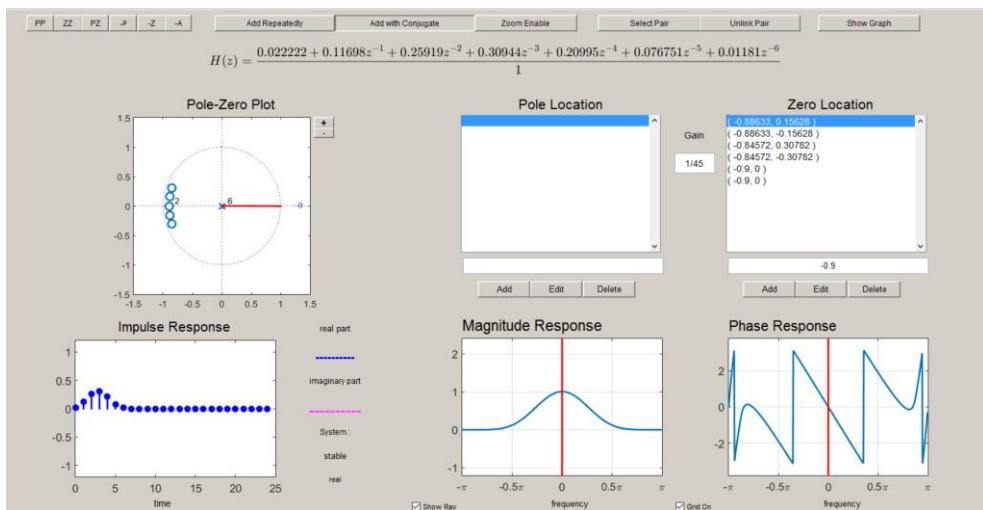
Phase Of IIR HP Filter Frequency Response ($r=9.000000e-01$)



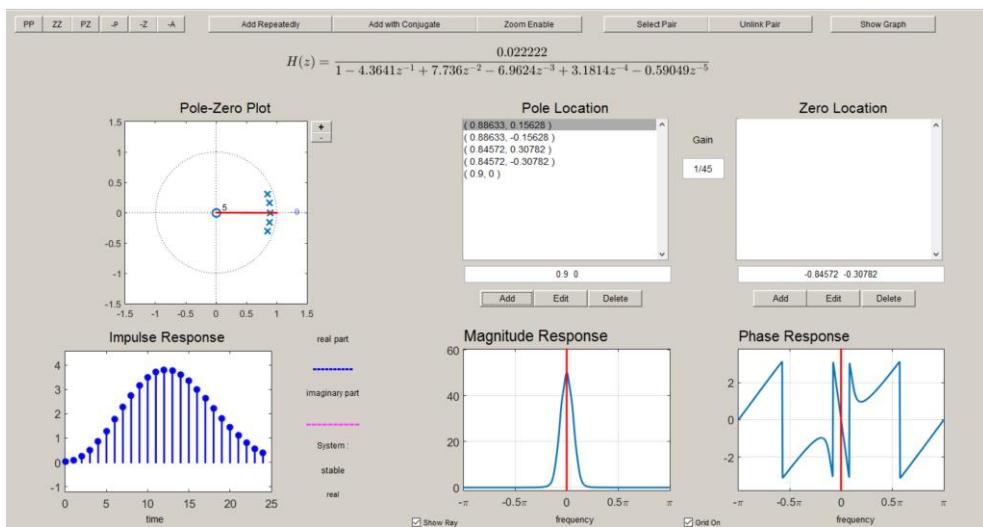


همان‌طور که در تصاویر بالا (IIR) مشاهده می‌شود با دور شدن قطب از مبدا پهنهای باند کم شده و تغییر فاز پله‌ای شدیدتر می‌شود. مطابق تصاویر اگر بخواهیم در فرکانس‌های بالا کار کنیم باید تغییر فاز ناگهانی پاسخ فرکانسی را تحمل کنیم. البته مزیت این فیلترها پهنهای باند کم آن‌ها و خاصیت فیلتری بیشتر آن‌هاست.

۷. برای بهتر کردن ویژگی‌های فیلترها، با توجه به روش‌های آموخته شده، در اطراف صفری که روی محور در فاصله r_0 از مبدا قرار دارد، روی دایره‌ای به همان شعاع ولی با زاویه اندکی از محور عددی (۴) صفر دیگر به صورت متقارن نسبت به محور افقی قرار می‌دهیم. در اطراف قطب برای حالت‌های IIR نیز همین روش را دنبال می‌کنیم. به عنوان مثال برای حالت $r_0 = 0.9$ و فیلتر پایین‌گذر از pezdemo نتایج زیر را می‌بینیم.



برای فیلتر بالاگذر و $r_0 = 0.9$



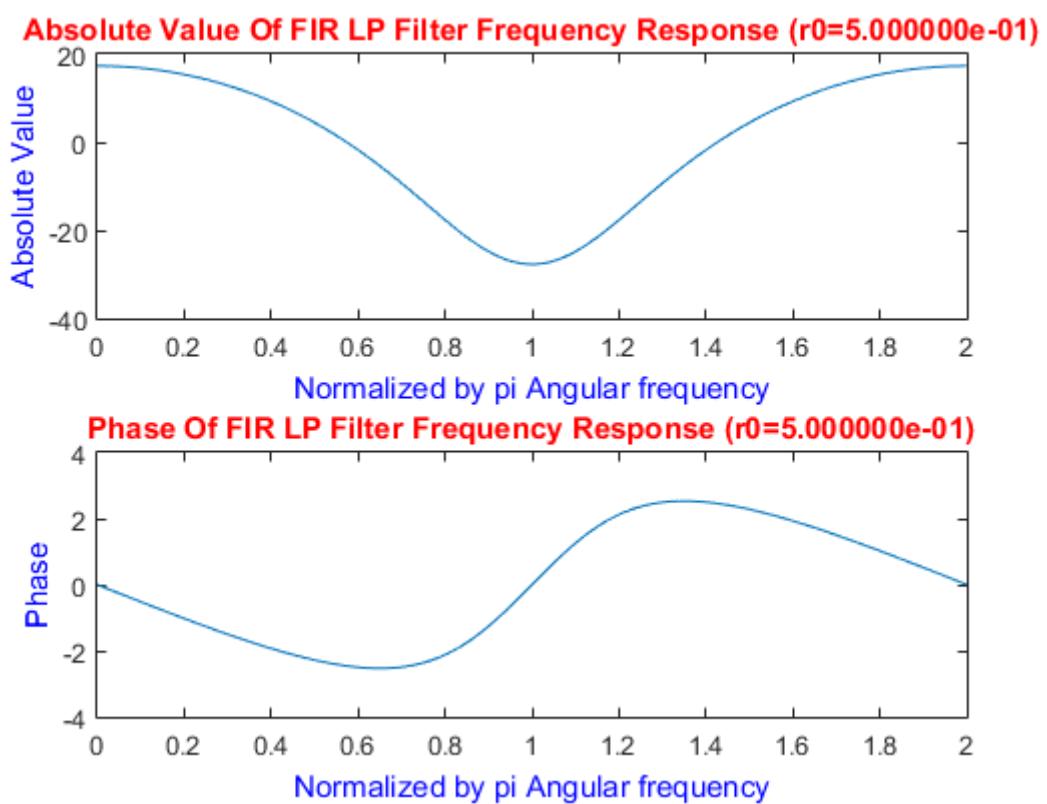
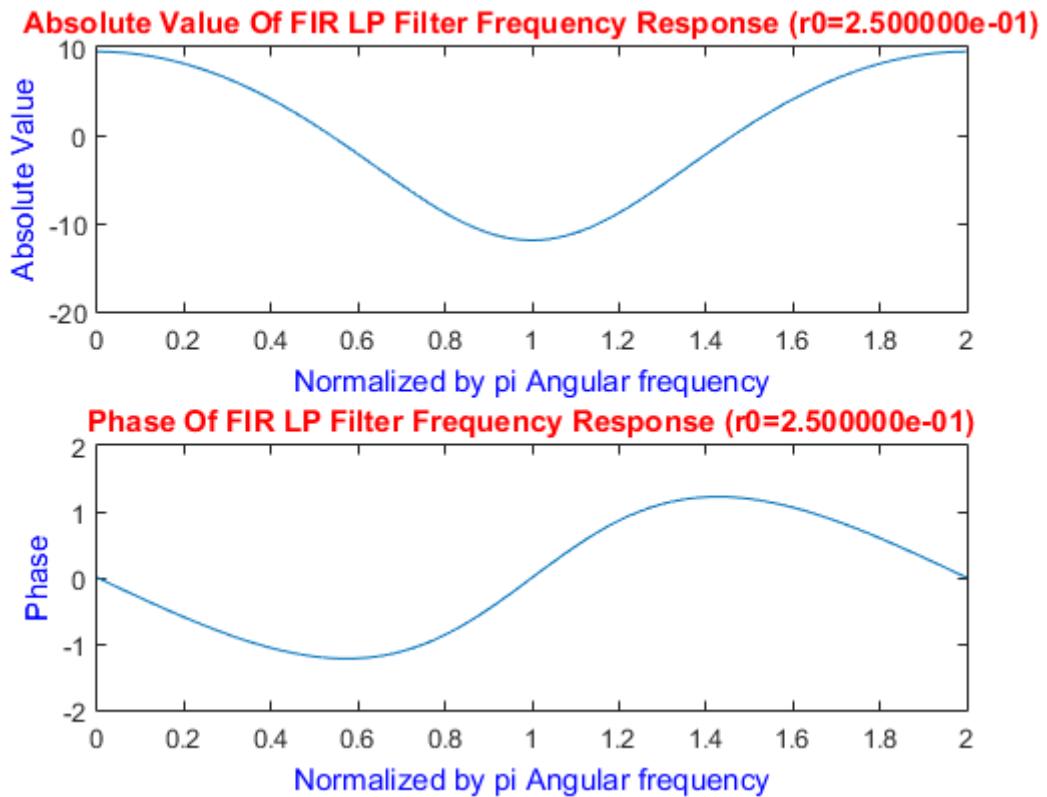
در ادامه پاسخ فرکانسی را برای فیلترهای IIR و FIR پایین‌گذر و بالاگذر ملاحظه می‌کنیم.

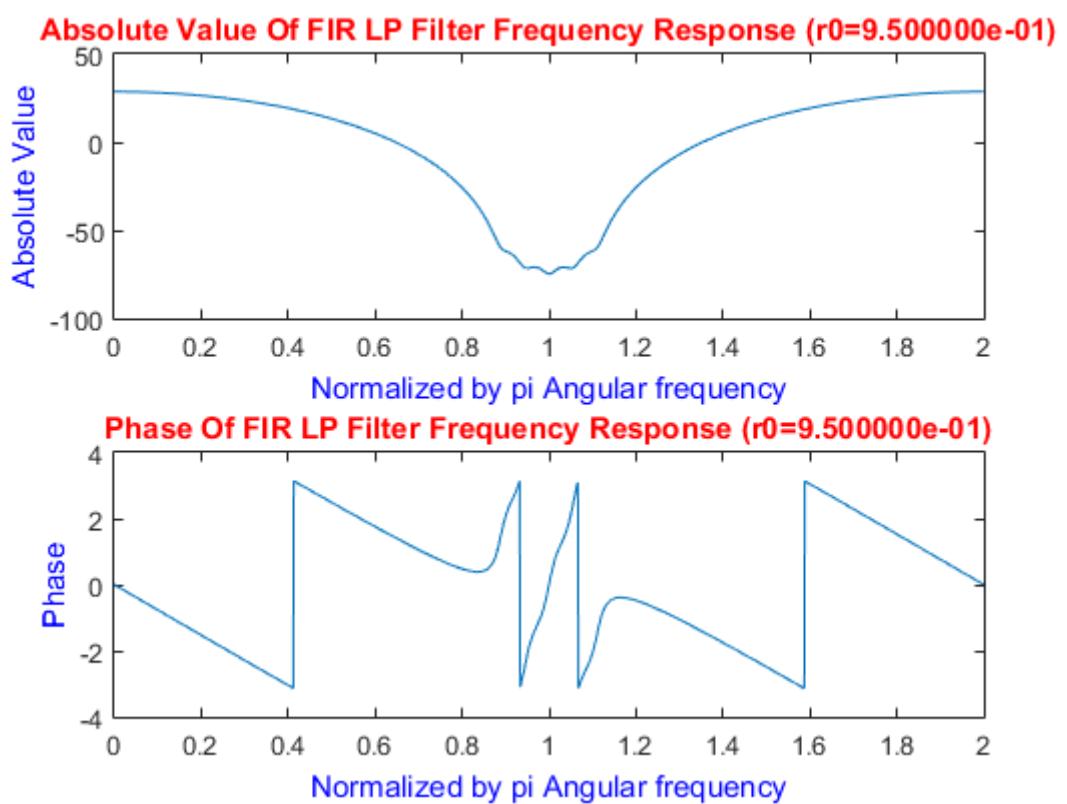
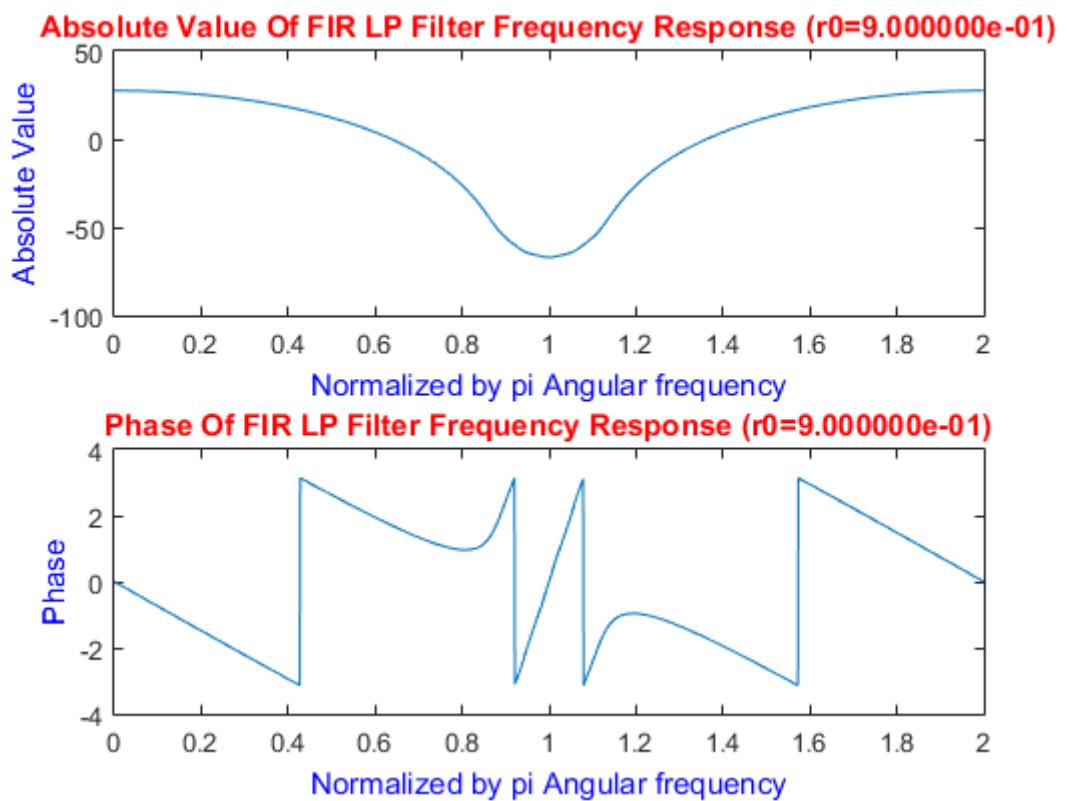
توضیح کد: ابتدا زوایای ۱۰ و ۲۰ درجه مشخص می‌شوند. در ادامه برای هر ۵ صفر مربوط به فیلتر پایین‌گذر حالت FIR و برای تمام مقادیر r_0 ، ماتریس‌هایی با ۲ سطر و ۶ ستون تشکیل شده‌اند که هر ستون به یک

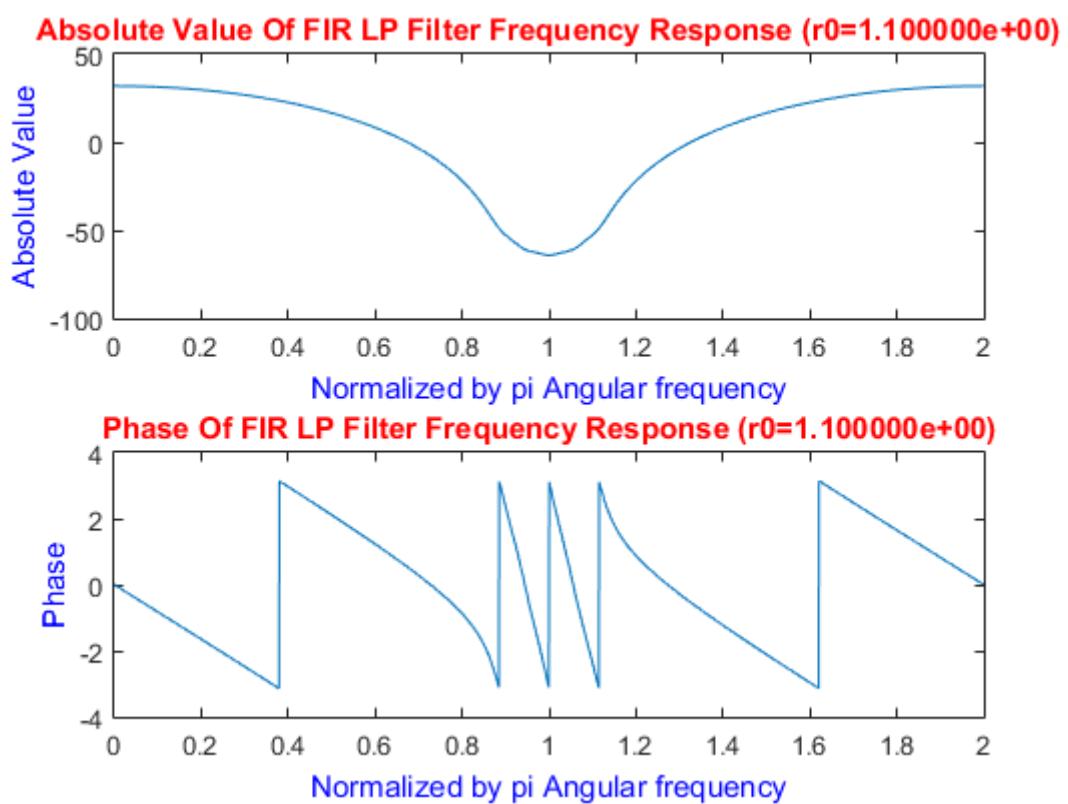
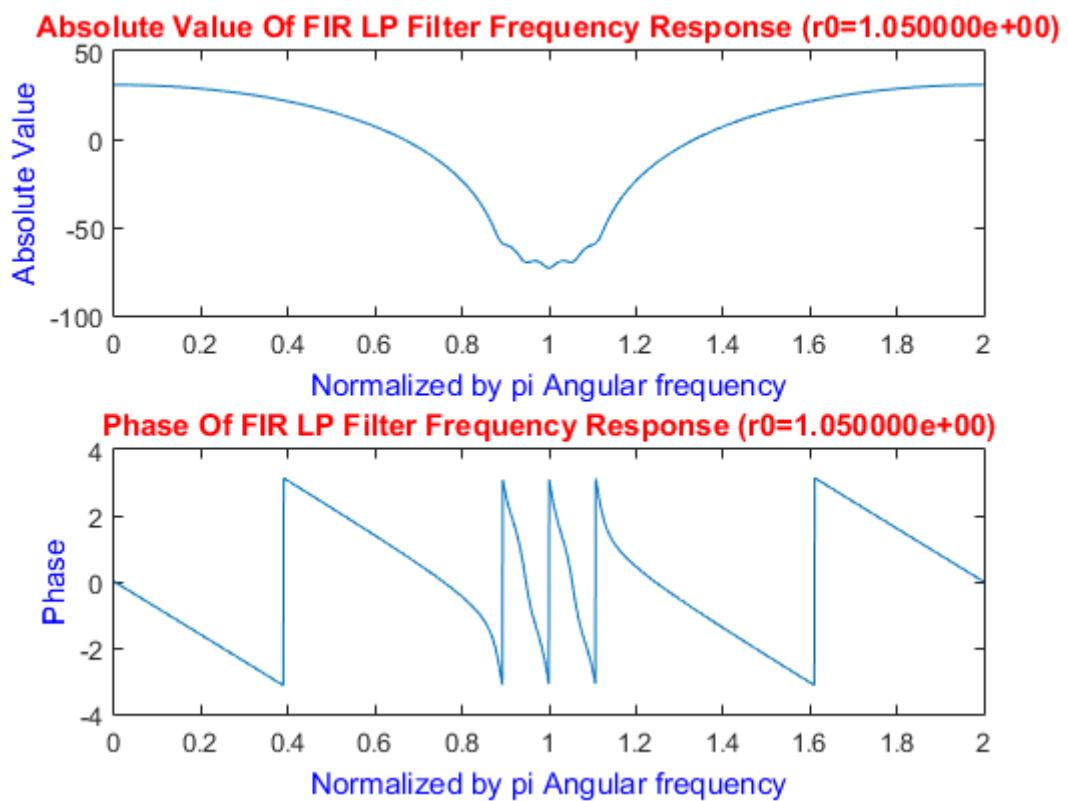
مقدار r_0 اختصاص دارد. در هر ستون، ضرایب صورت کسر تابع تبدیل قرار دارند. در نهایت باید این ۵ صفر باید در هم ضرب شوند تا صورت تابع تبدیل نهایی بدست آید. در ادامه ماتریسی 6×6 تعریف می‌شود که در هر سطر آن مقادیر ضرایب صورت تابع تبدیل به ازای یک r_0 مشخص قرار دارد. ۵ صفر در هم ضرب می‌شوند و یک عدد ثابت داریم). پس از آن نیز مخرج مشخص می‌شود که می‌دانیم برابر یک است. در حلقه for برای هر ۶ مقدار r_0 ضرایب در هم ضرب می‌شوند. (از تمرین سری ۱ به خاطر داریم کانولوشن ضرایب دو چندجمله‌ای ضرایب حاصل ضرب دو چندجمله‌ای را بدست می‌دهد). پس از آن پاسخ فرکانسی‌ها محاسبه می‌شوند. بعد از آن برای حالت FIR و پایین‌گذر اندازه و فاز پاسخ فرکانسی رسم می‌شود. پس از آن نیز برای حالت IIR و بالاگذر رسم می‌شود. پس از رسم همه نمودارهای فوق، همین روند برای بالاگذرها دنبال می‌شود. نتایج در تصاویر زیر مشهودند:

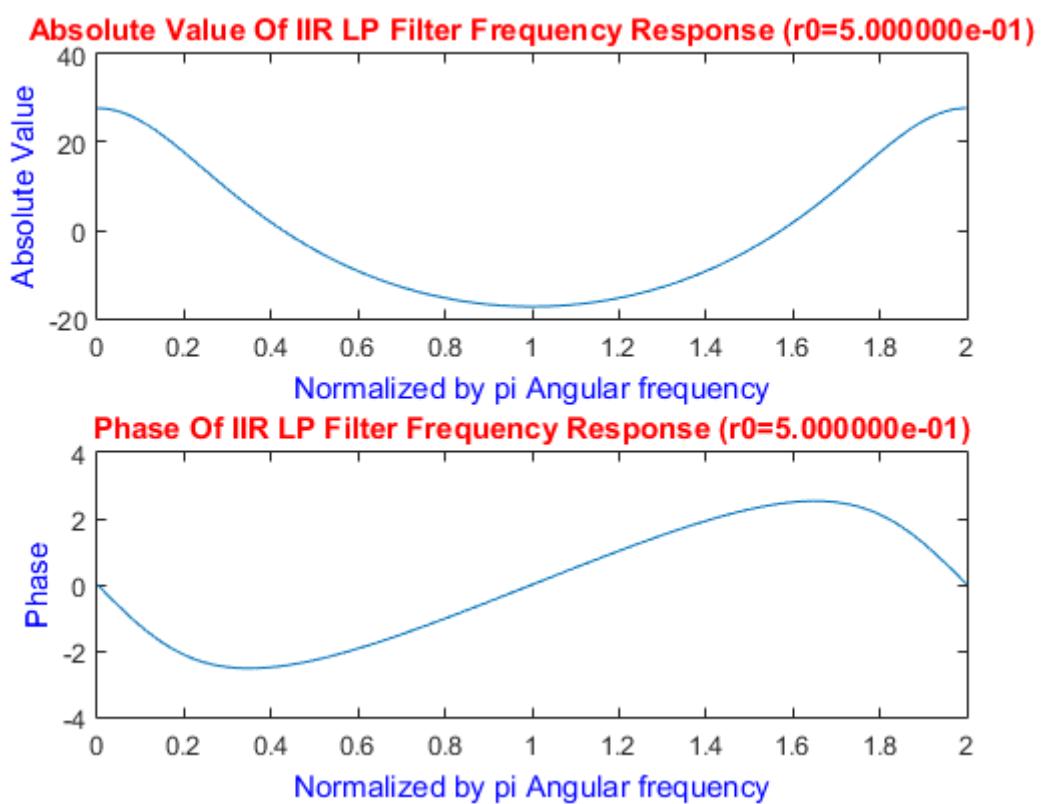
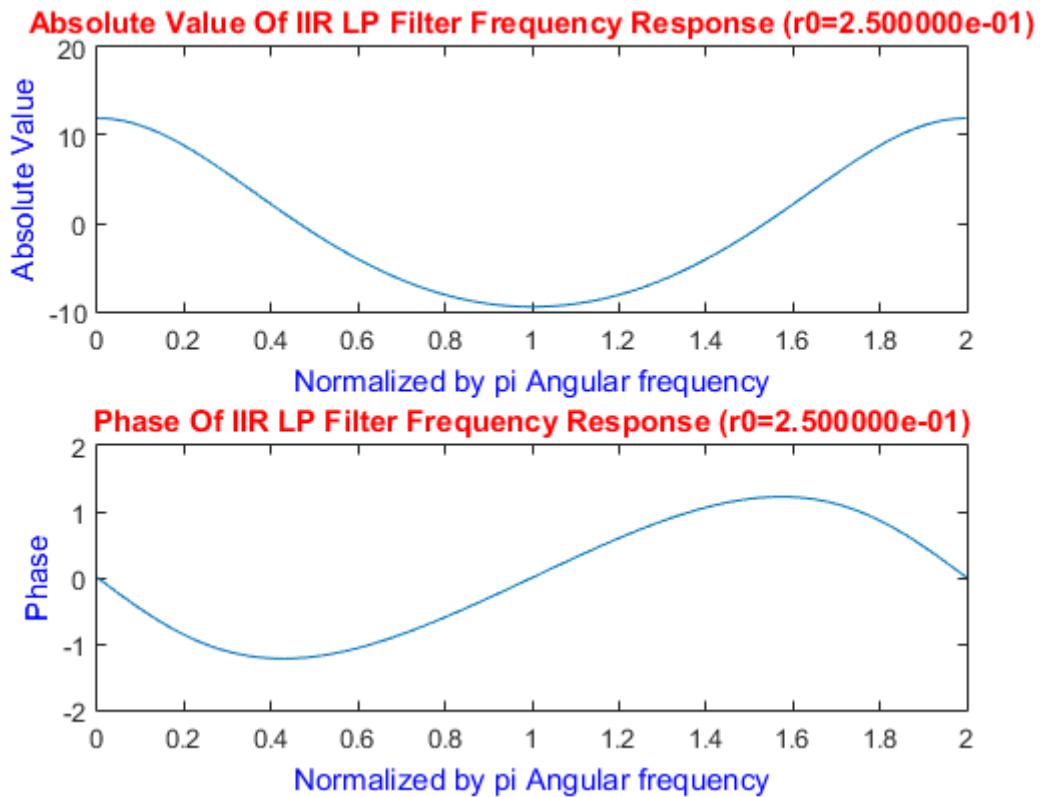
با مقایسه این فیلترهای جدید با همتاها یشان می‌توان به خاصیت فیلتری بهتر آن‌ها پی‌برد. (هم از لحاظ شدت تضعیف دیگر فرکانس‌ها و هم از لحاظ پهنای باند). اما فاز آن‌ها مشخصه مطلوبی ندارند.

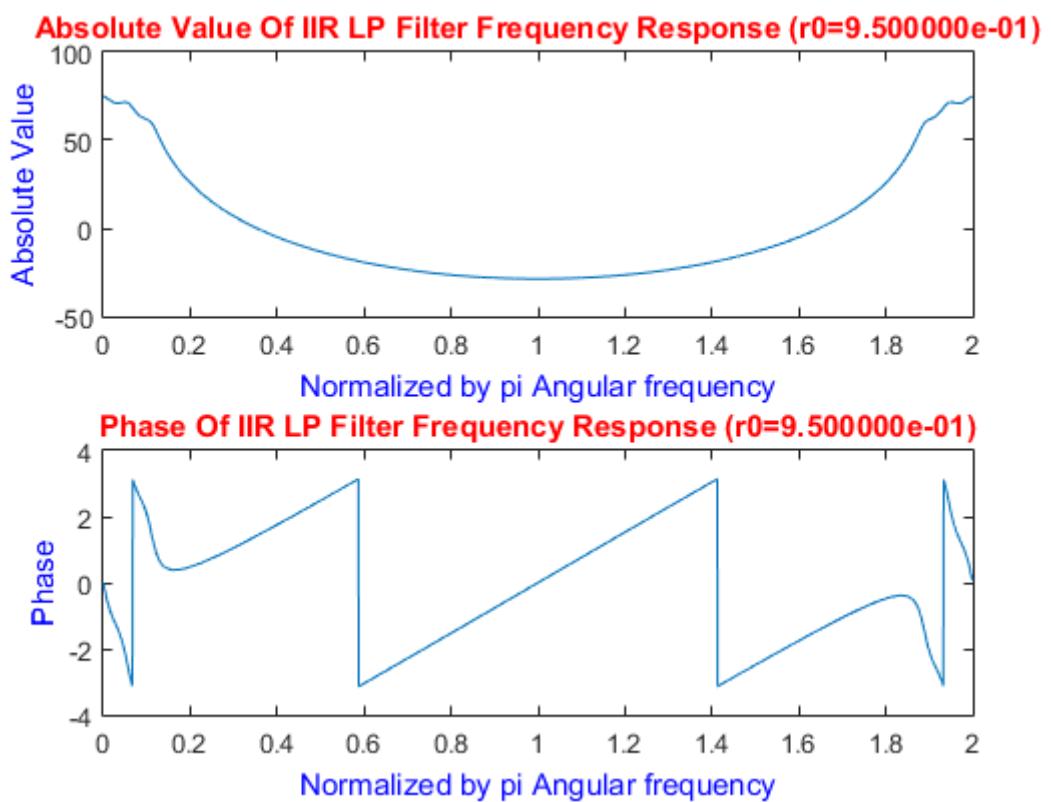
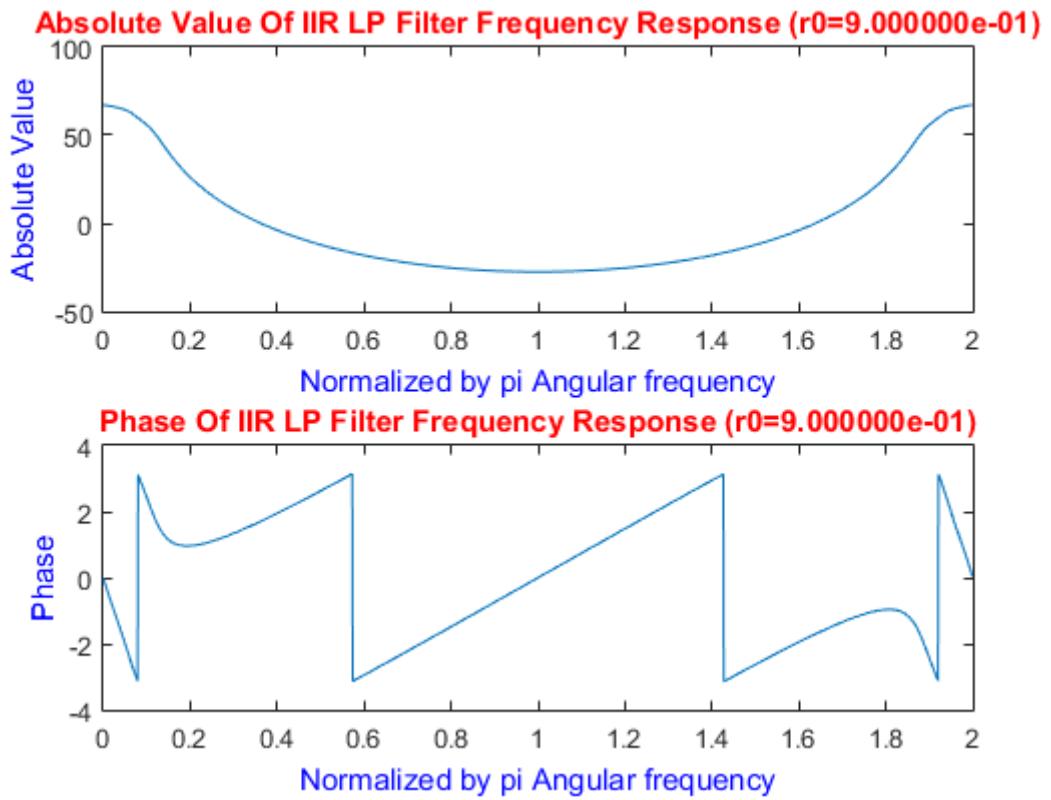
پایین‌گذر:

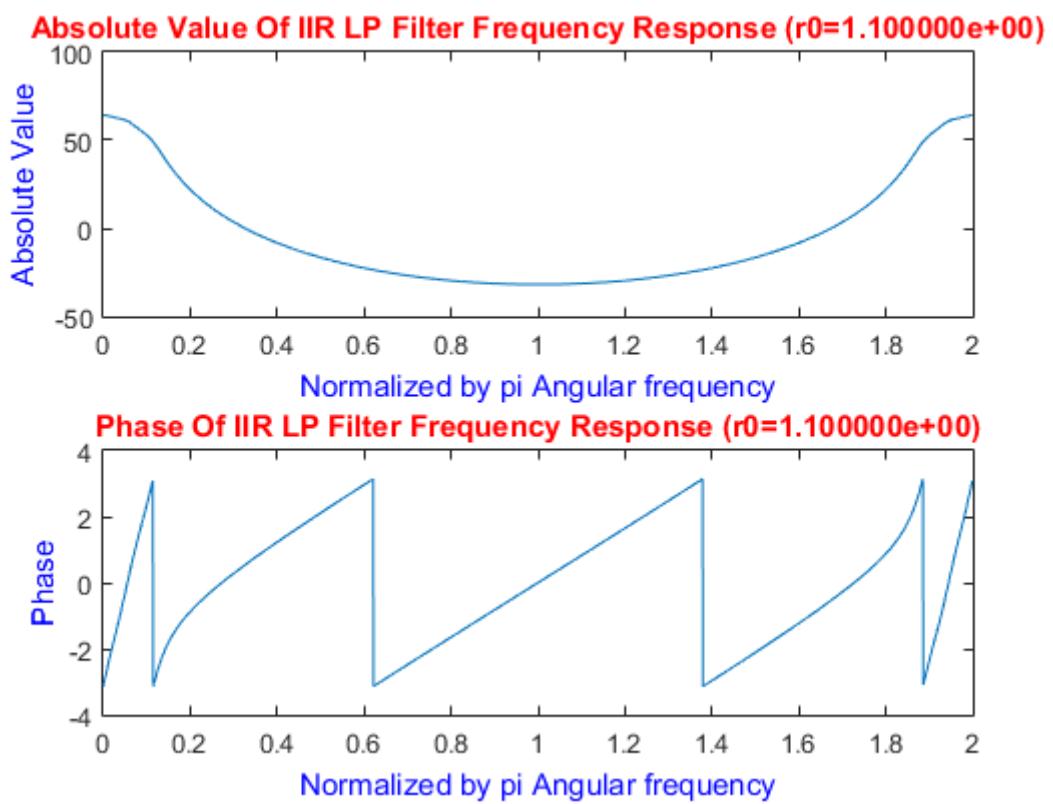
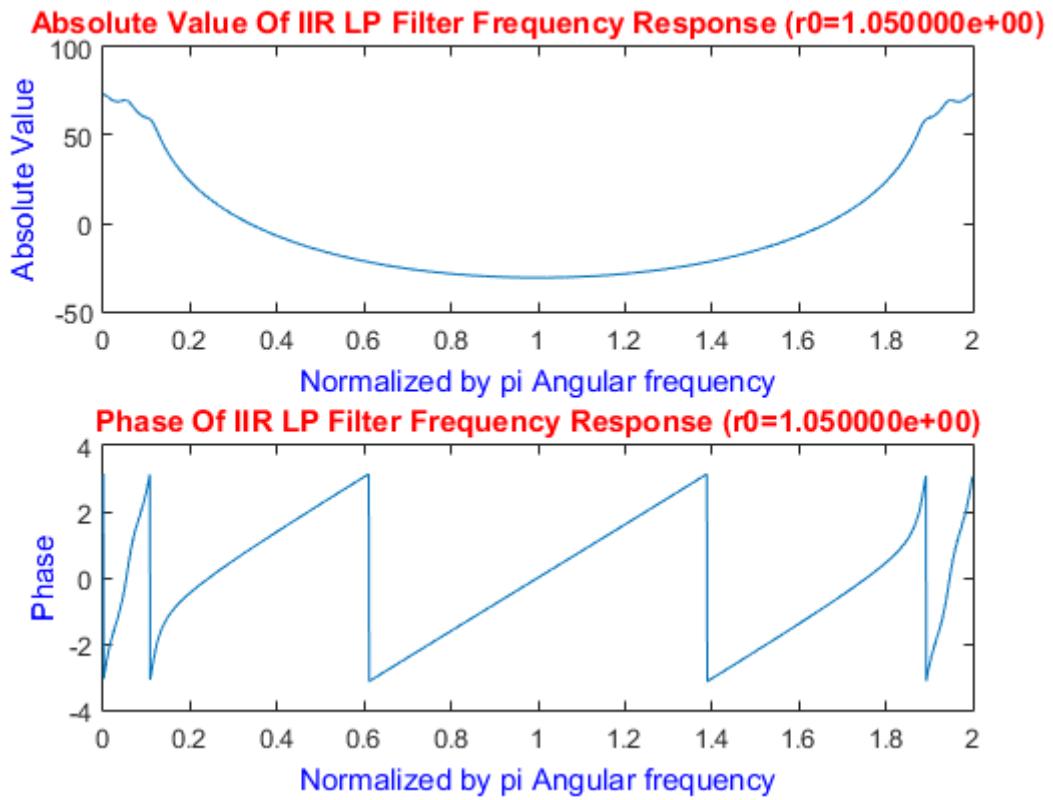




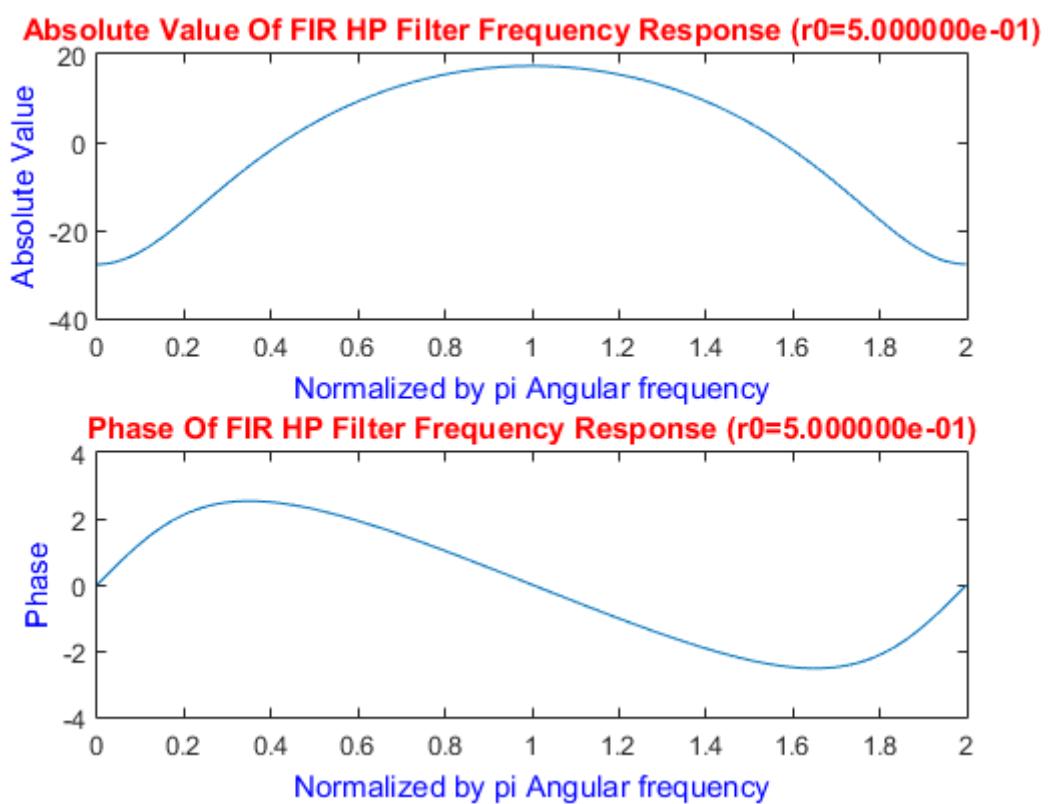
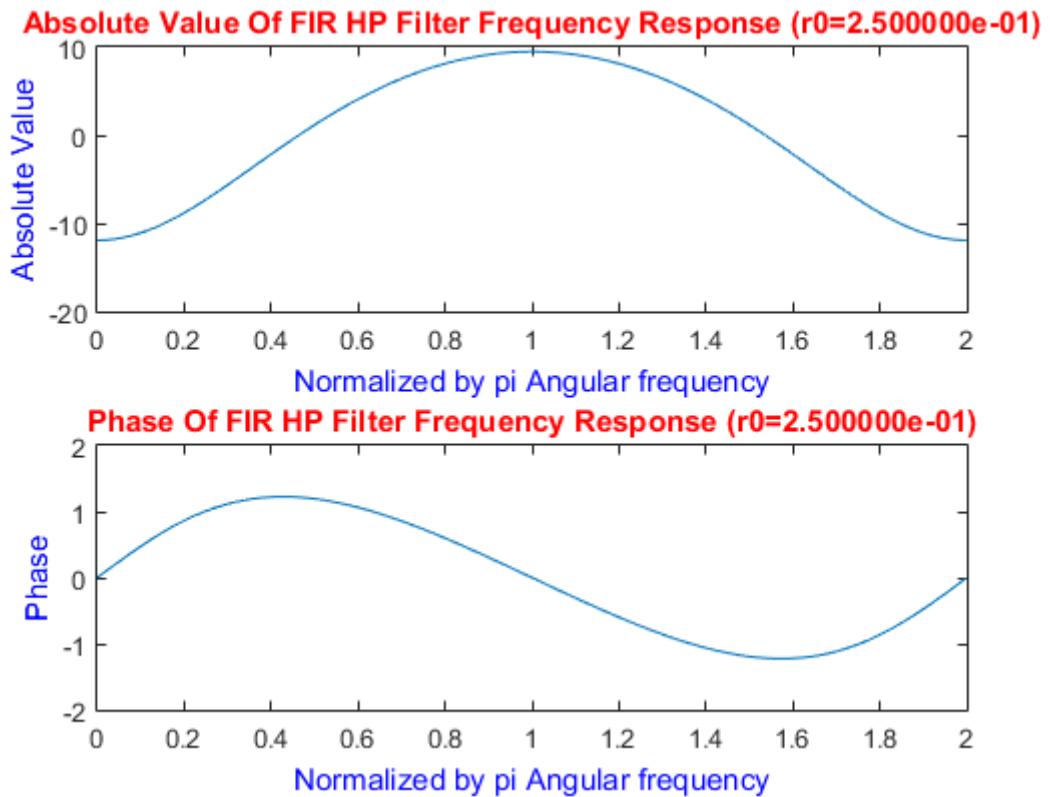


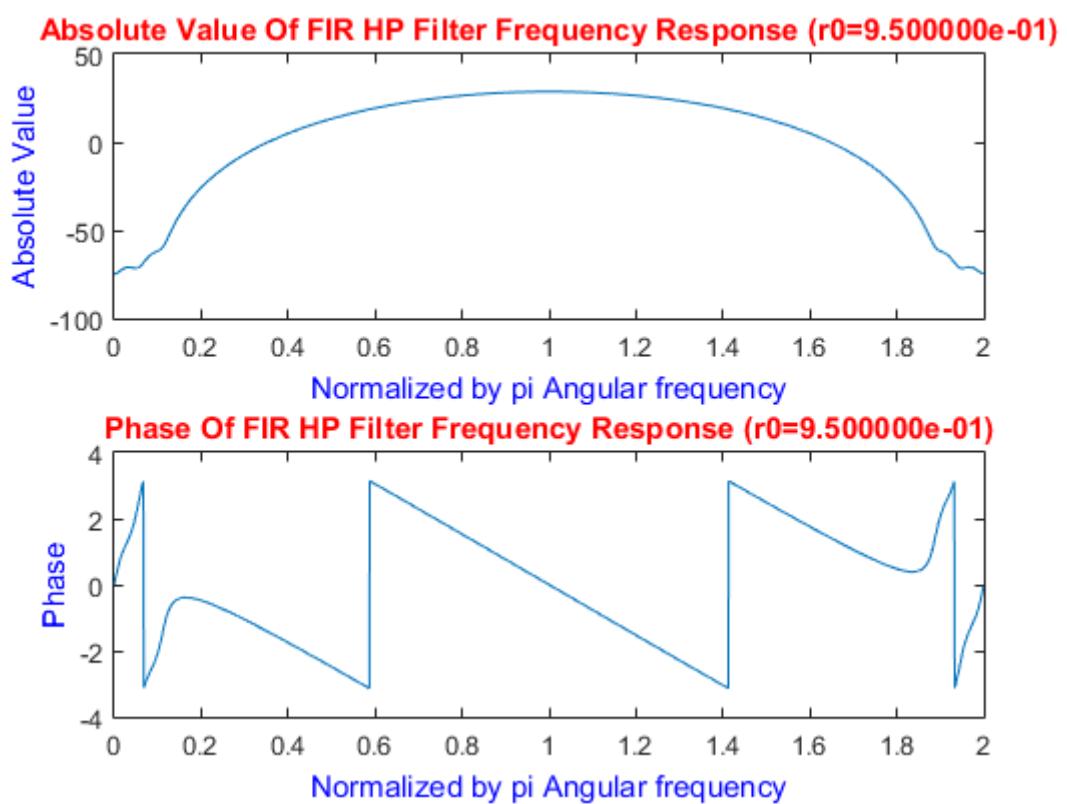
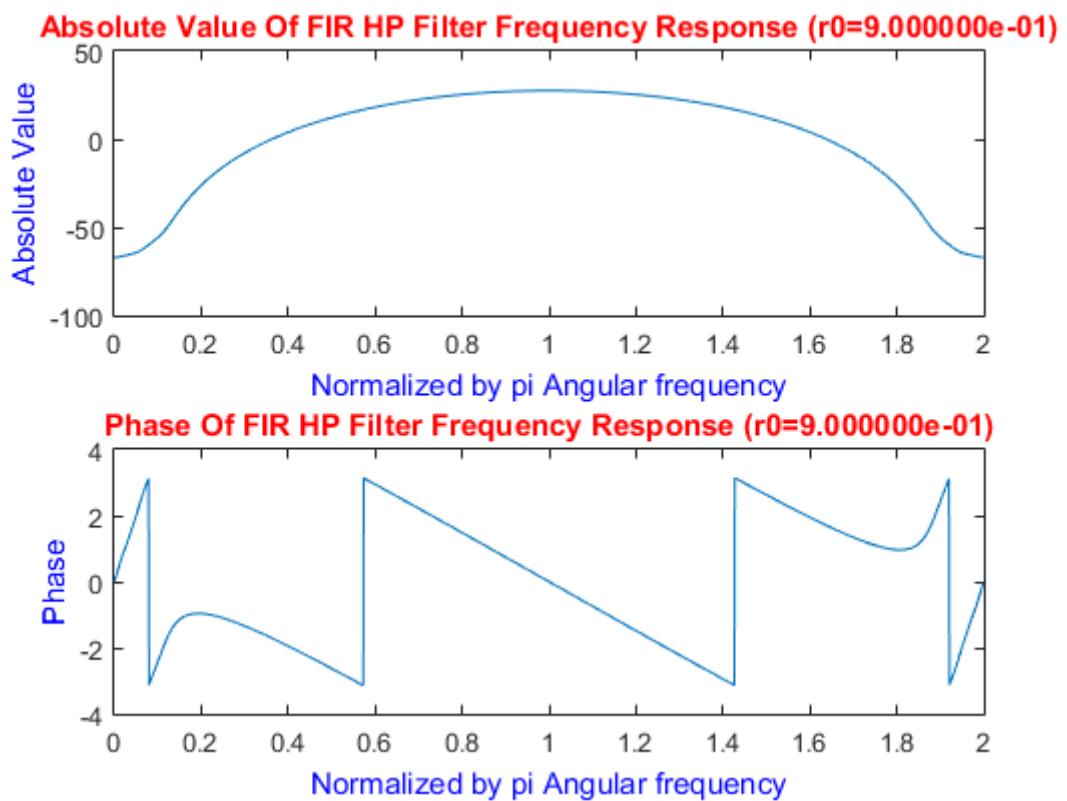


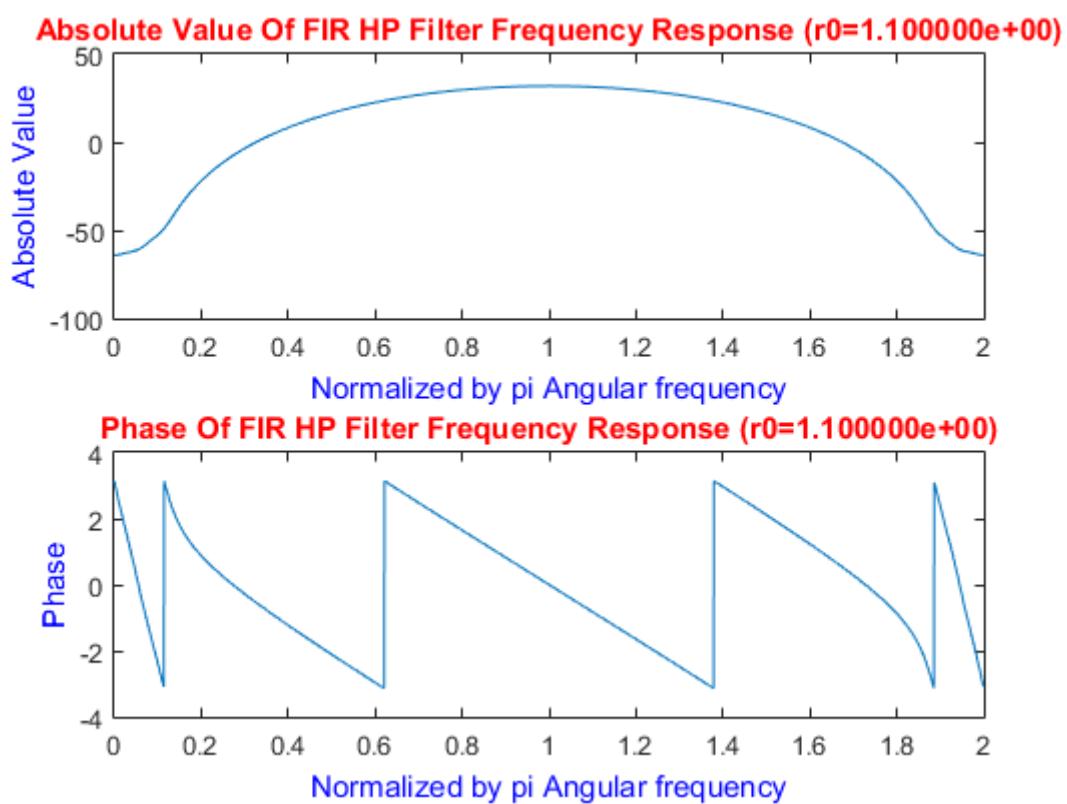
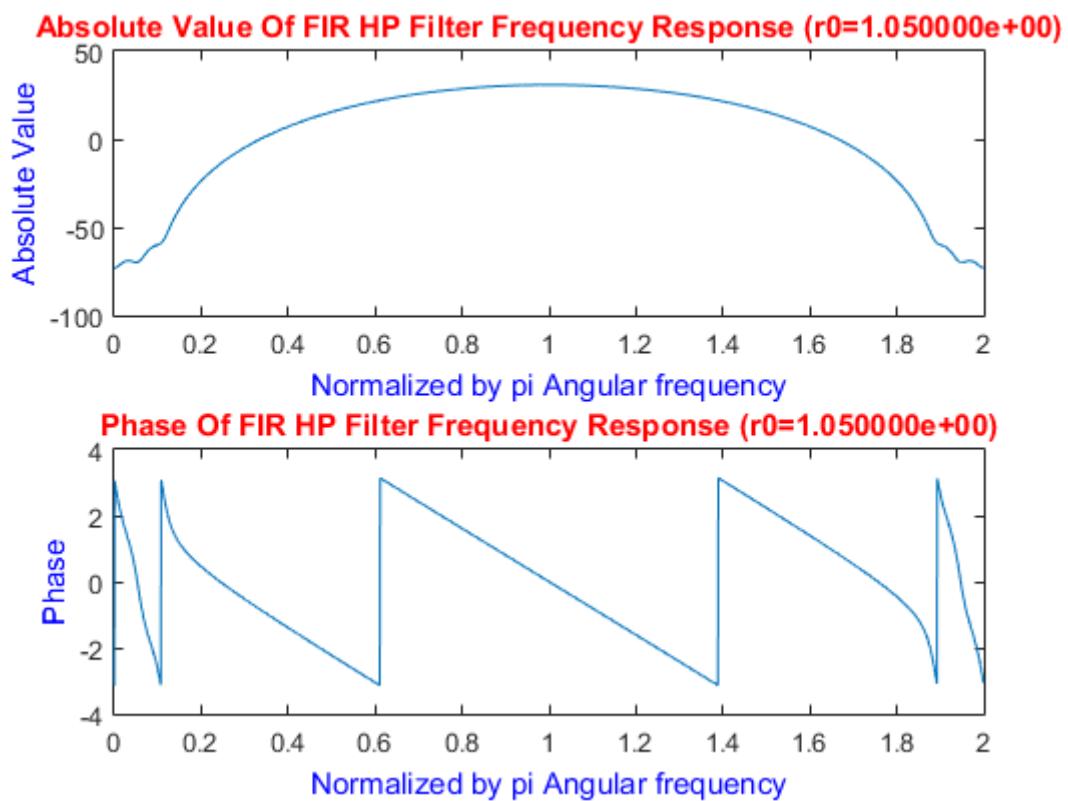


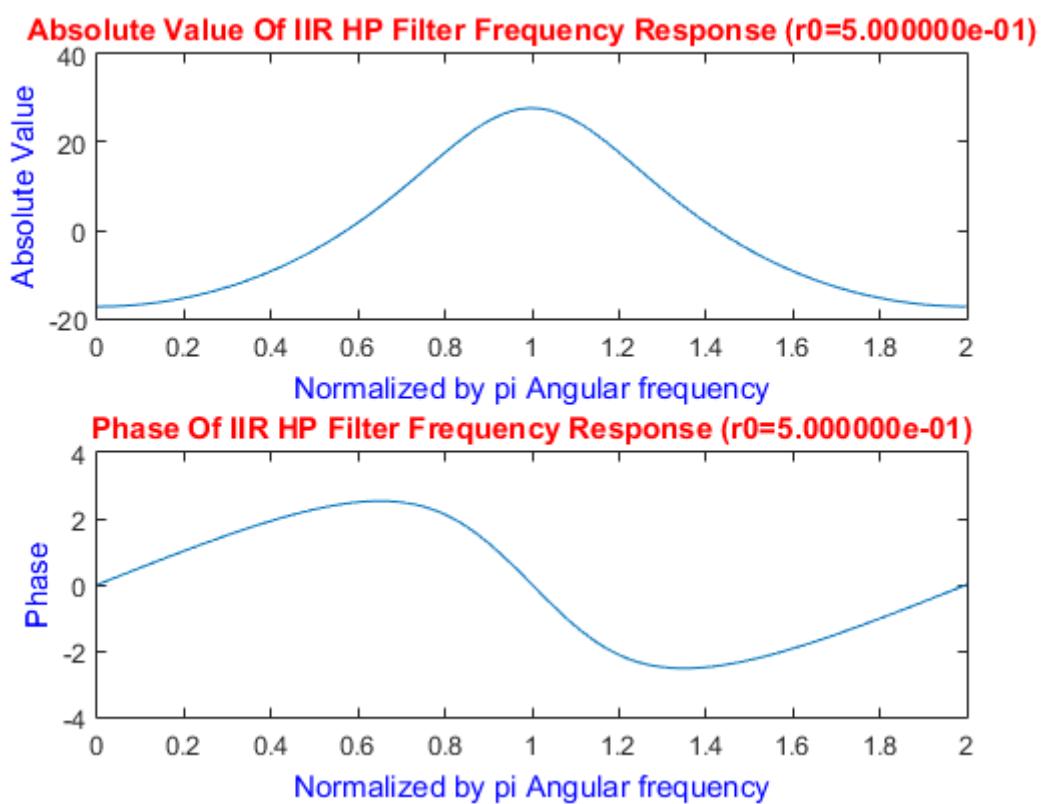
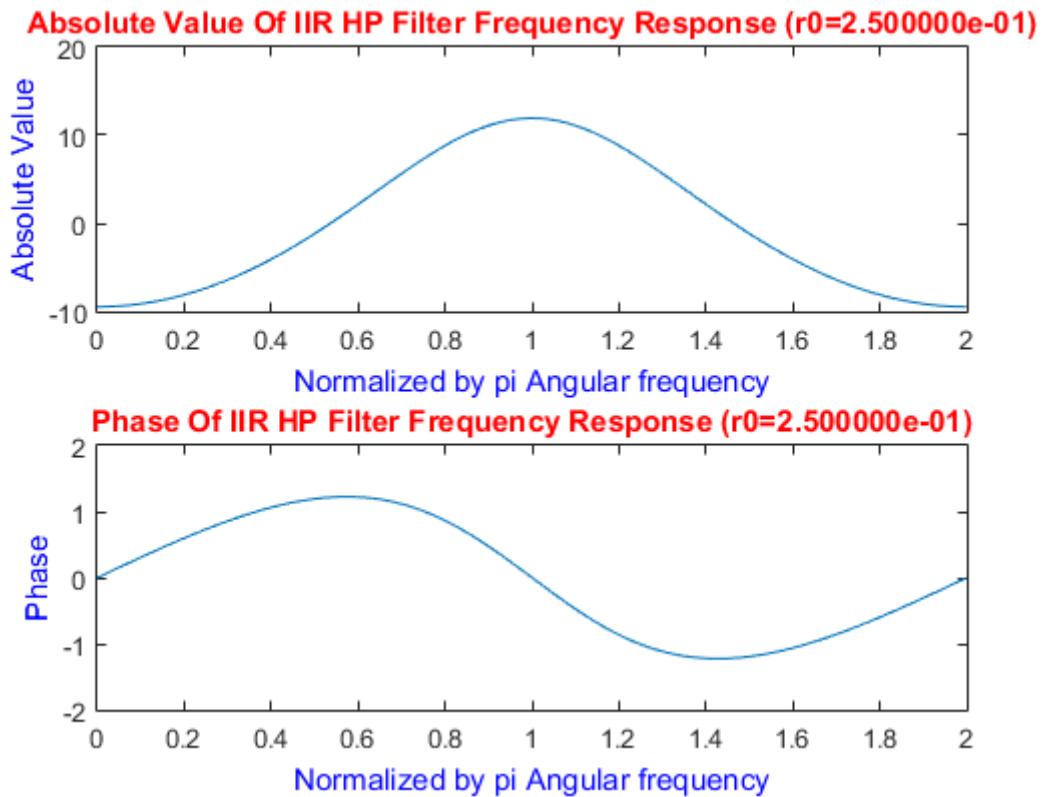


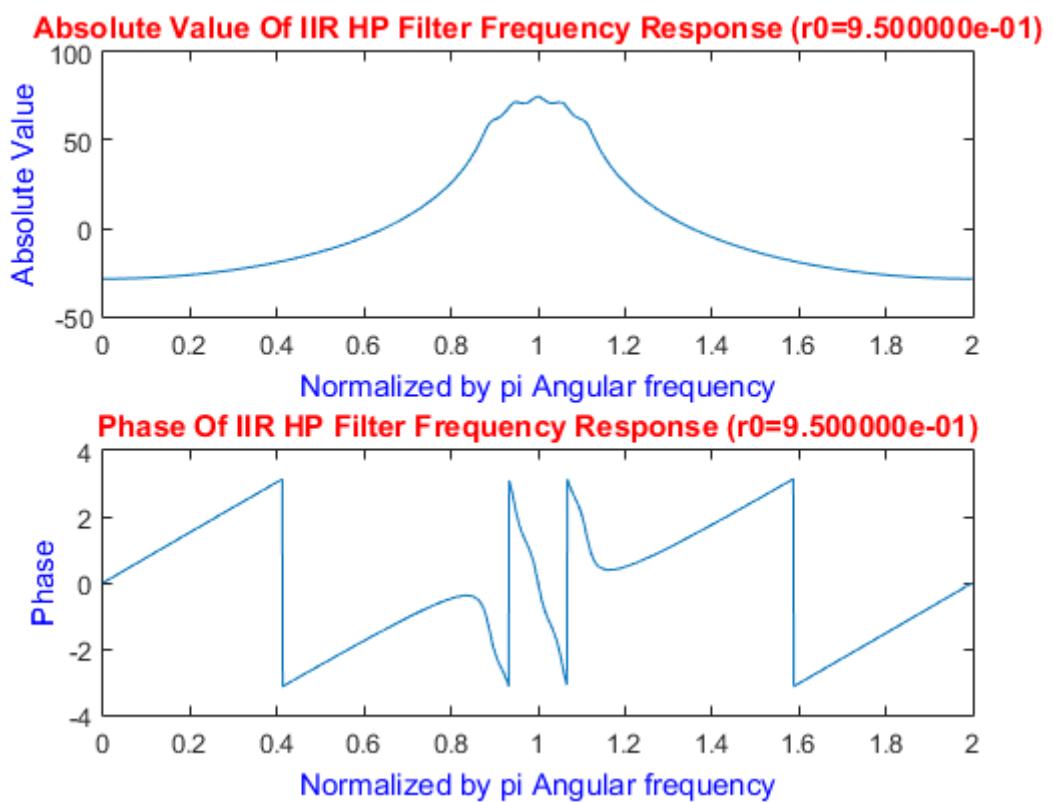
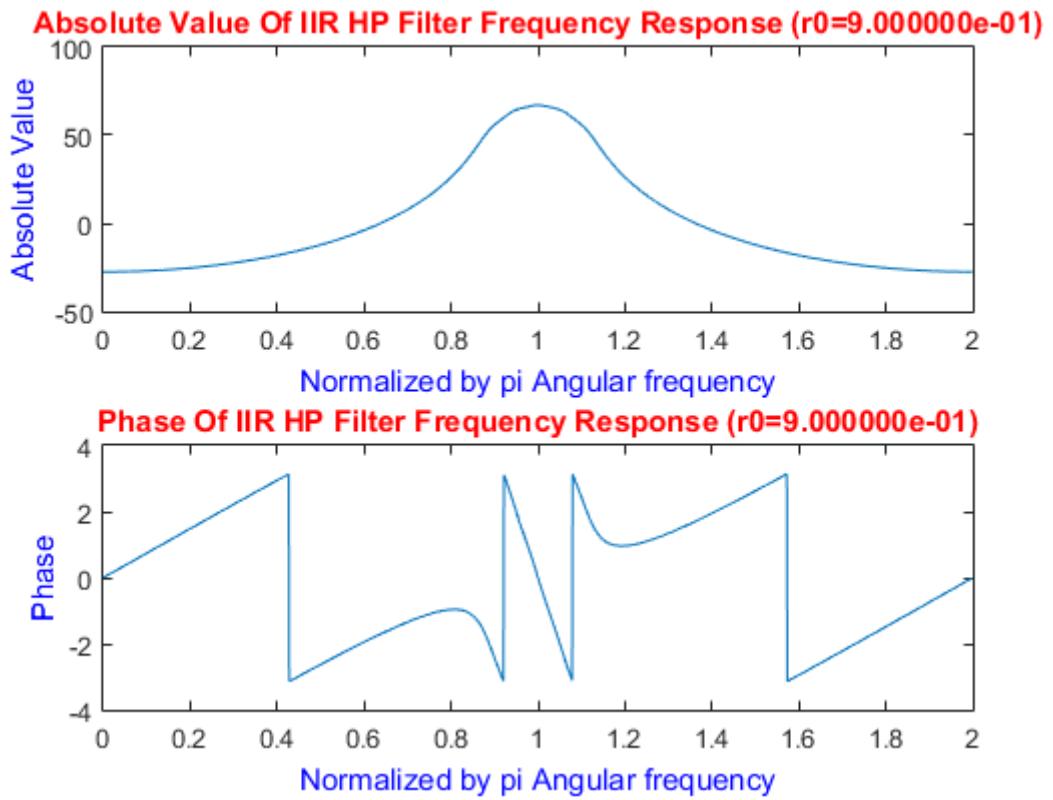
بالاگذر:

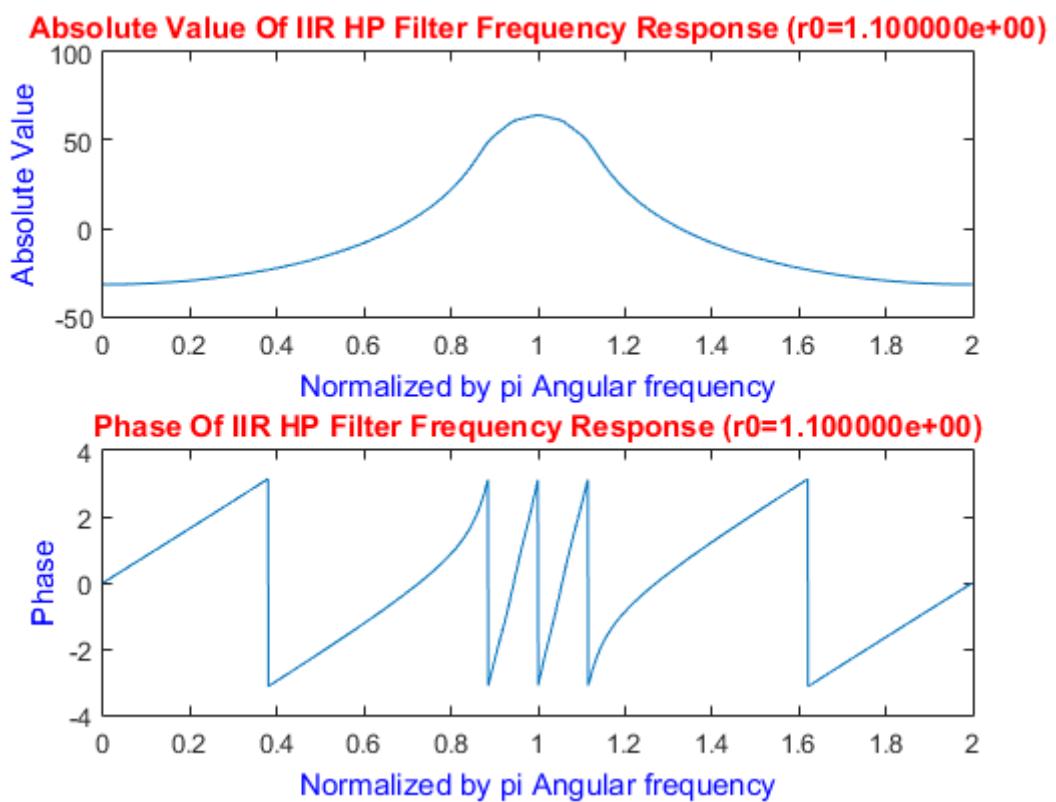
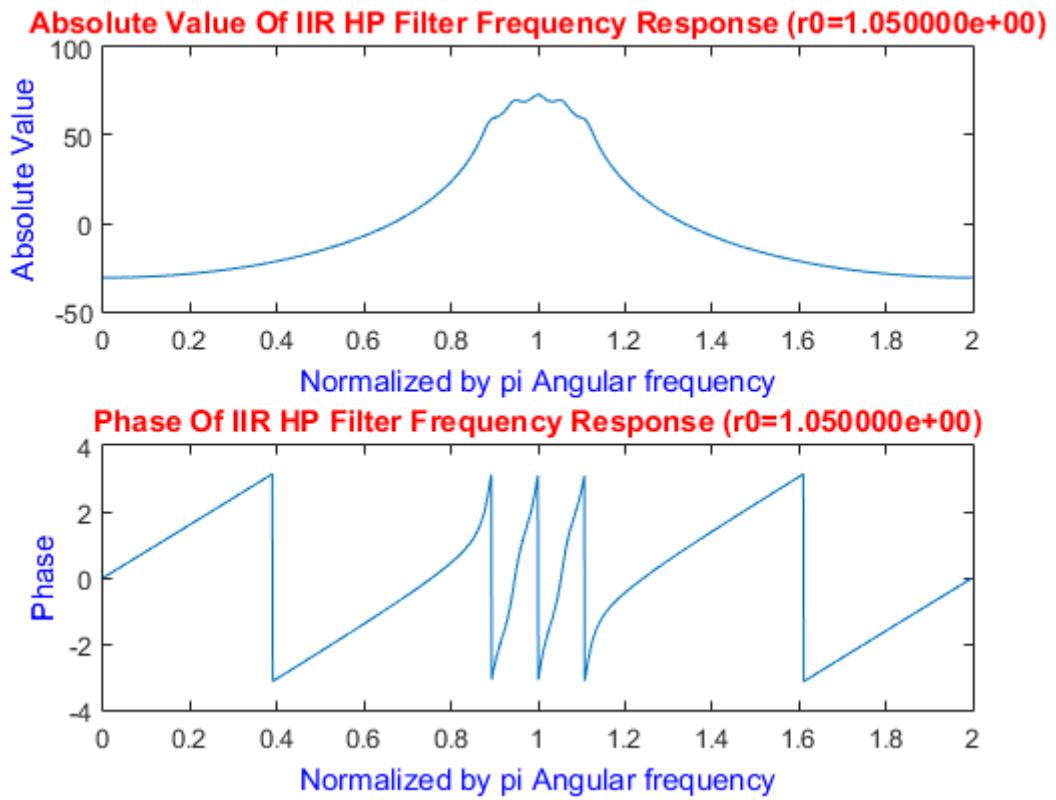






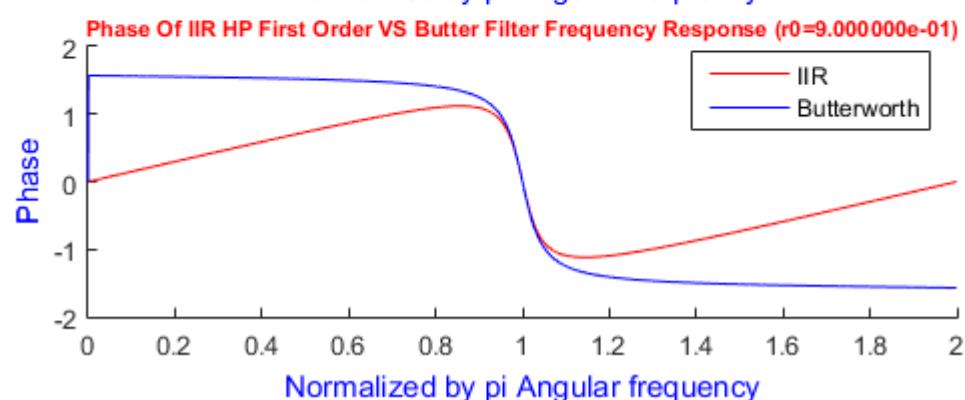
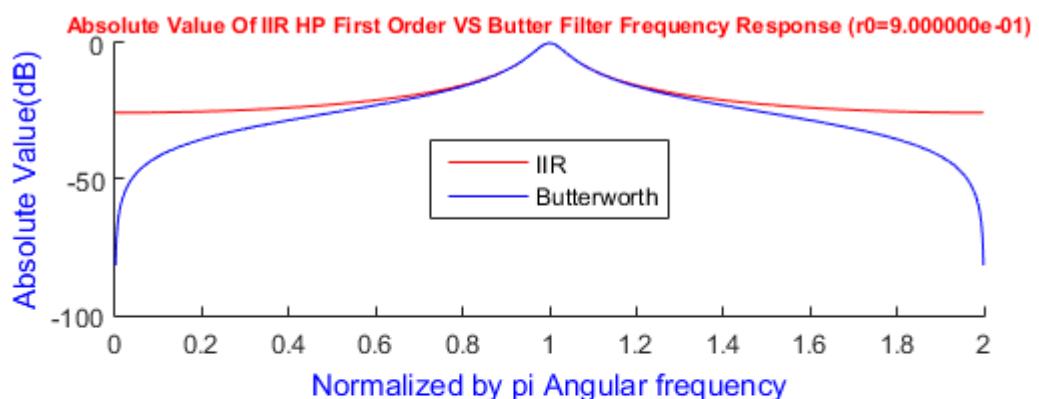
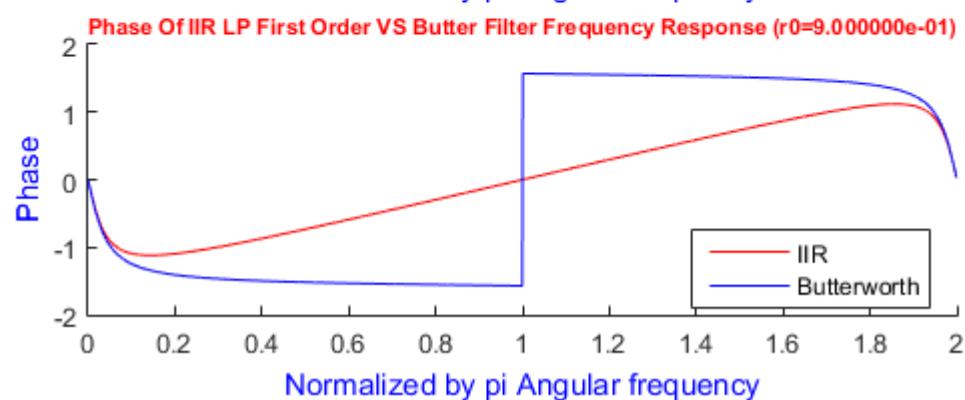
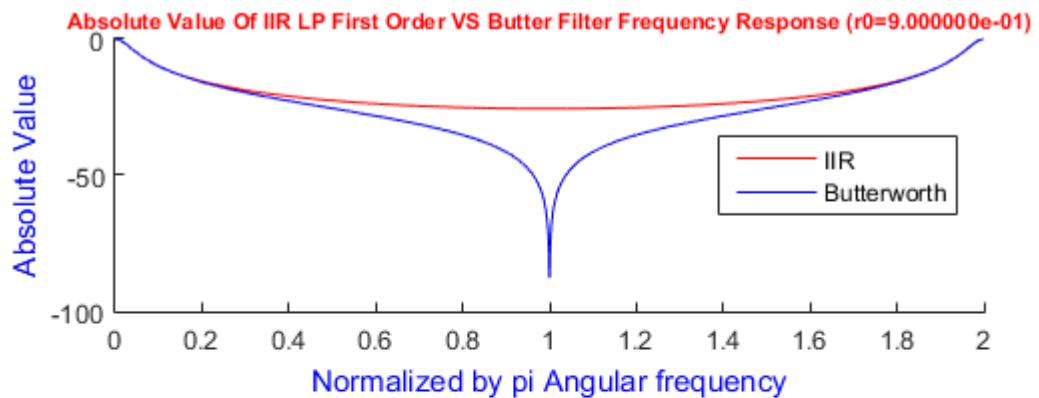


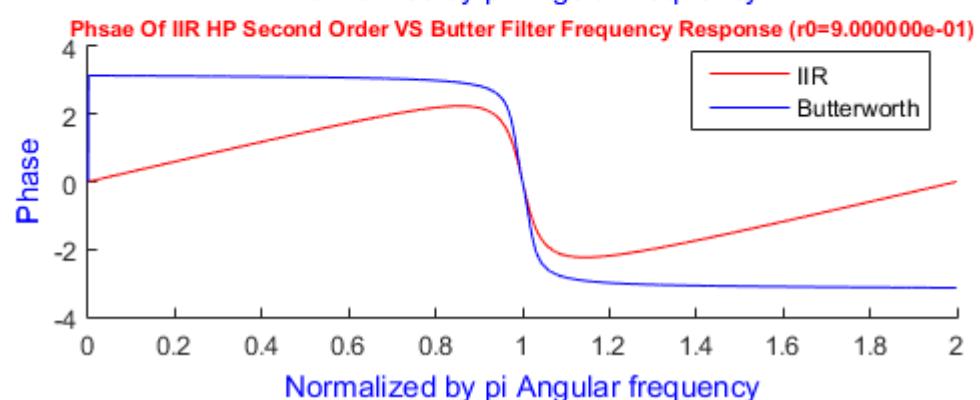
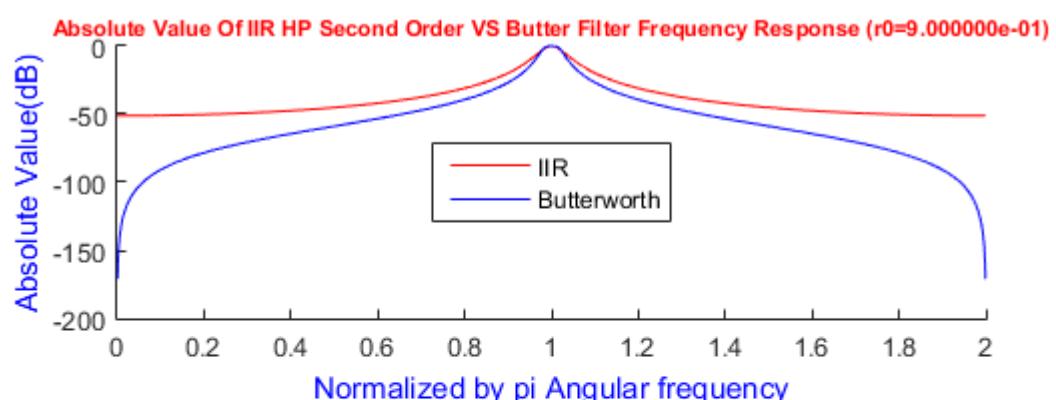
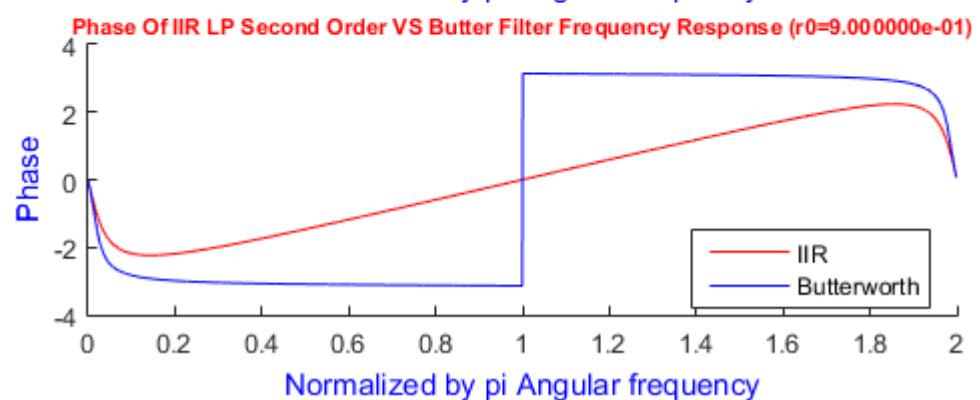
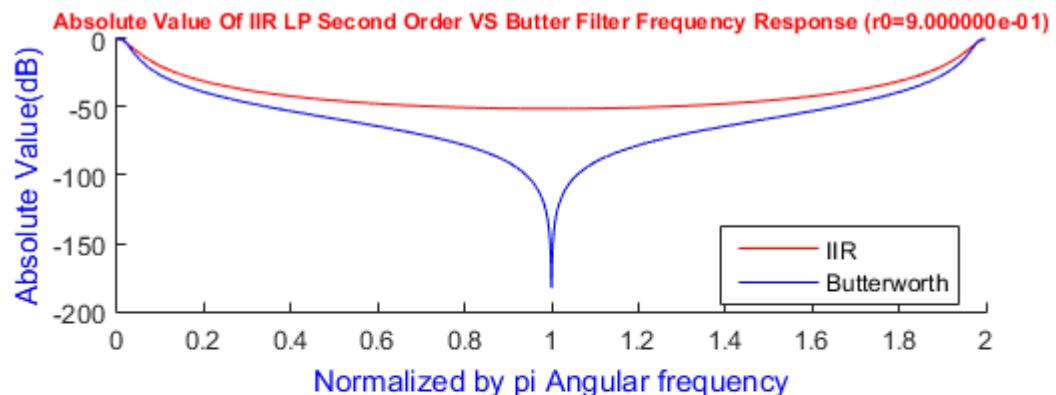


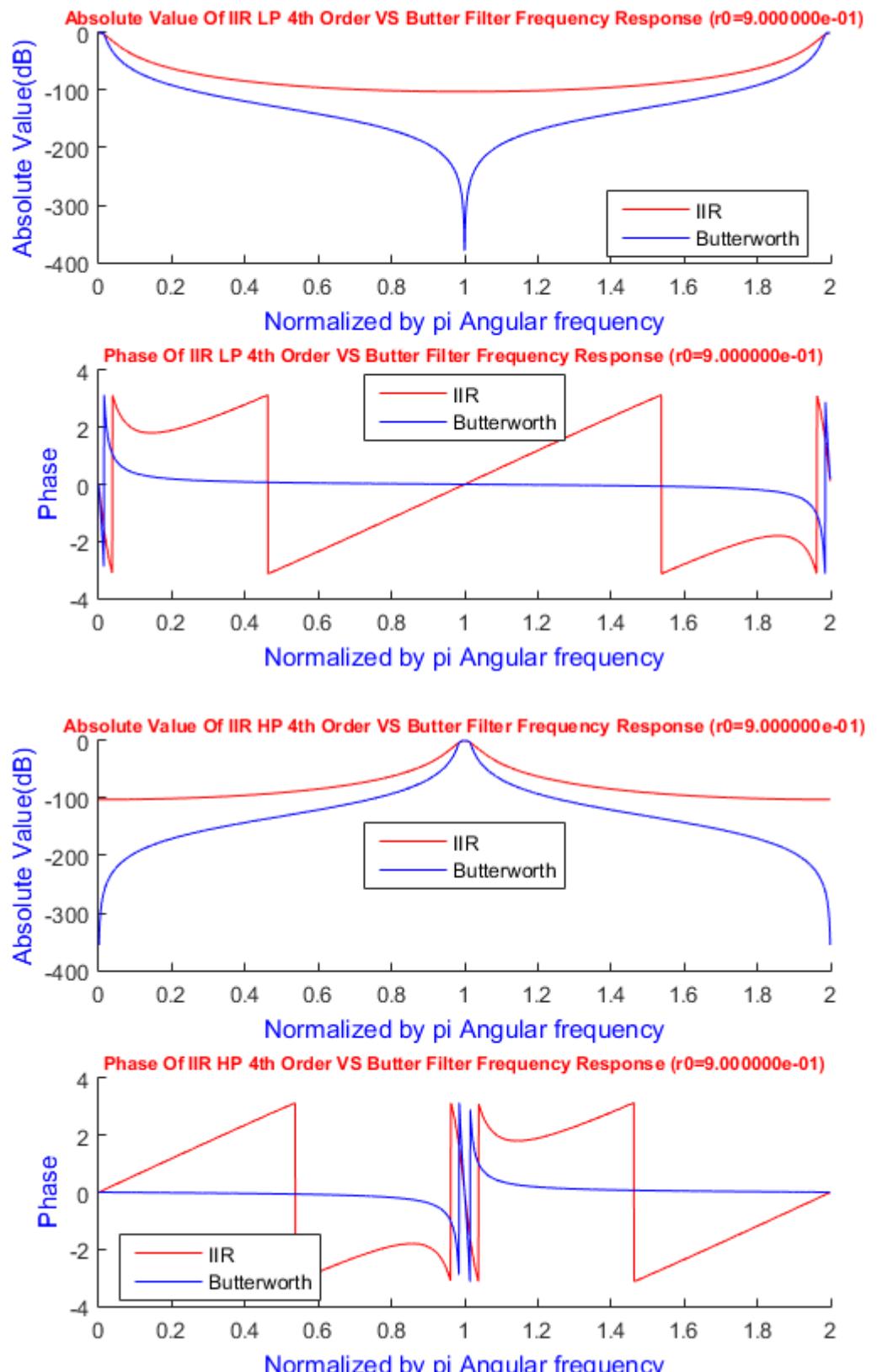


۸. فیلتر مناسب IIR : ویژگی مناسب بودن را پهنانی باند و خطی بودن فاز در نظر گرفتیم. به ازای $r_0 = 0.9$ ، هم پهنانی باند مقدار معقولی دارد هم فاز از لحاظ خطی بودن بهتر است. در سایر موارد یا تغییرات فاز زیاد است یا خاصیت فیلتری بیش از اندازه زیاد یا کم است. برای بالا بردن مرتبه فیلتر IIR با کانولوشن ضرایب صورت فیلتر مرتبه ۱ و همینطور کانولوشن ضرایب مخرج فیلتر مرتبه ۱ به تعداد دفعات مورد نیاز(n) مرتبه فیلتر را بالا می‌بریم. فیلتر متناظر پایین‌گذر و بالاگذر را با دستور butter می‌سازیم. این دستور در ورودی خود دو مقدار می‌گیرد. ورودی اول آن مرتبه فیلتر و ورودی دوم مقدار نرمالایز شده‌ی فرکانس قطع است. (مقادیر روی محور افقی در تصاویر قسمت‌های قبل همان فرکانس‌های زاویه‌ای نرمالایز شده هستند). بنابراین برای مقدار این ورودی دوم، از روی شکل‌های مربوط به فیلترهای ساخته شده با پشت سر هم گذاشتن فیلترهای قبلی، مقدار فرکانس قطع نرمالایز شده را از روی نقطه‌ای که $3db$ افت کرده است، می‌خوانیم. در هر مرحله این فرکانس‌های قطع نرمال شده را به ورودی butter داده و ضرایب تابع تبدیل را در خروجی آن می‌گیریم.

توضیح کد: در ابتدای قسمت ۱،۸، بدار مرتبه‌ها(۱ و ۲ و ۴) ساخته می‌شود. سپس صورت فیلتر IIR پایین‌گذر و بالاگذر مناسب برابر ۱ قرار داده می‌شود. (فقط قطب دارند و بنابراین صورت یک است). در ادامه ابتدا برای butter فیلتر پایین‌گذر و مرتبه ۱ ضرایب صورت از قسمت‌های قبل قرار داده می‌شود. سپس از دستور ضرایب صورت تابع تبدیل فیلتر با ترورث مرتبه یک و پایین‌گذر بدست می‌آید. بعد از آن از قسمت‌های قبل ضرایب صورت فیلتر مرتبه ۱ و بالاگذر جدید قرار داده می‌شود و در نهایت ضرایب مخرج مخرج تابع تبدیل فیلتر با ترورث مرتبه ۱ و بالاگذر از دستور butter بدست می‌آید. در ادامه کد با داشتن ضرایب صورت و مخرج برای هریک از حالات پایین‌گذر و بالاگذر و برای هردو فیلتر IIR و با ترورث، بهوسیله دستور freqz مقادیر پاسخ فرکانسی و فرکانس‌هایی که این مقادیر در آن‌ها محاسبه شده‌اند را بدست می‌آوریم. نهایتاً نوبت به مرحله رسم می‌رسد. با استفاده از subplot، در قسمت بالایی نمودار اندازه پاسخ فرکانسی برای هردو فیلتر IIR و با ترورث و در قسمت پایینی آن نمودار فاز پاسخ فرکانسی برای هردو فیلتر رسم می‌شود. دقت می‌کنیم که مقادیر اندازه پاسخ فرکانسی برای هردو فیلتر به منظور نرمال کردن، به بیشترین مقدار اندازه پاسخ فرکانسی تقسیم شده‌اند. ابتدا اندازه و فاز پایین‌گذر و پس از آن اندازه و فاز بالاگذر رسم می‌شوند. همان‌طور که در مشاهده می‌شود منحنی‌های قمزرنگ به فیلتر IIR و منحنی‌های آبرنگ به فیلتر با ترورث (legend) مربوطاند. در ادامه کد نیز دقیقاً همین مراحل برای مرتبه‌های ۲ و ۴ تکرار می‌شوند. نتایج در تصاویر زیر قابل رؤیت است.



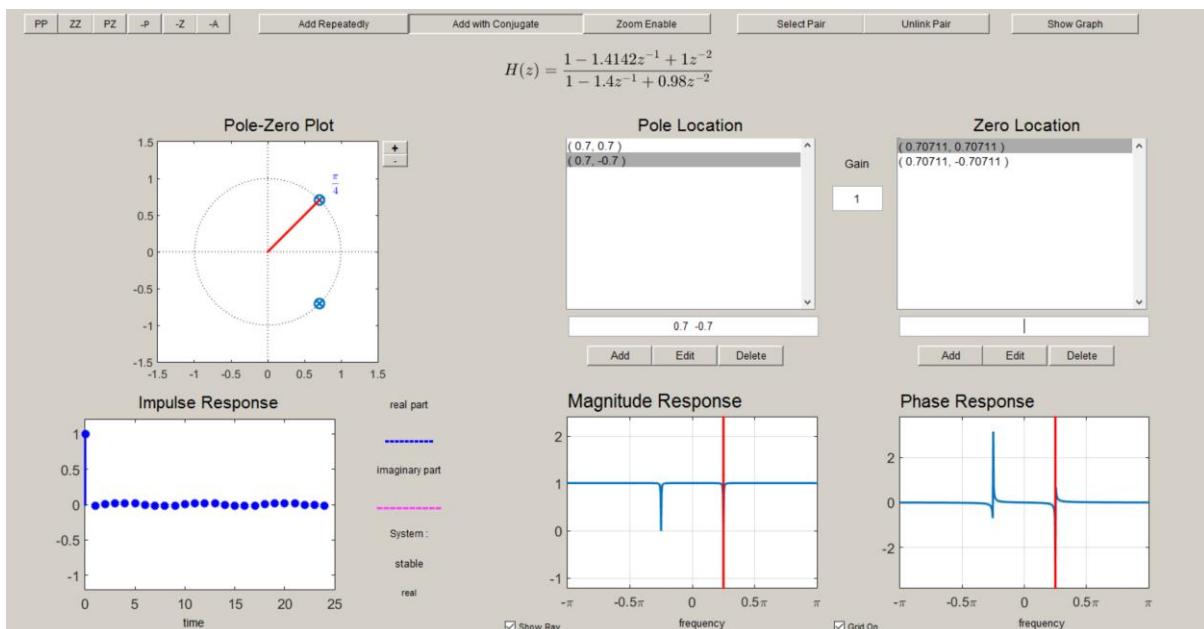




همان‌طور که در تصاویر فوق مشاهده می‌شود، در هر مرتبه‌ای و هریک از دو نوع بالاگذر با پایین‌گذار، فیلتر با ترورث نسبت به متناظر IIR خود خاصیت فیلتری بسیار بهتری دارد. همچنین در نواحی گذار سریع‌تر

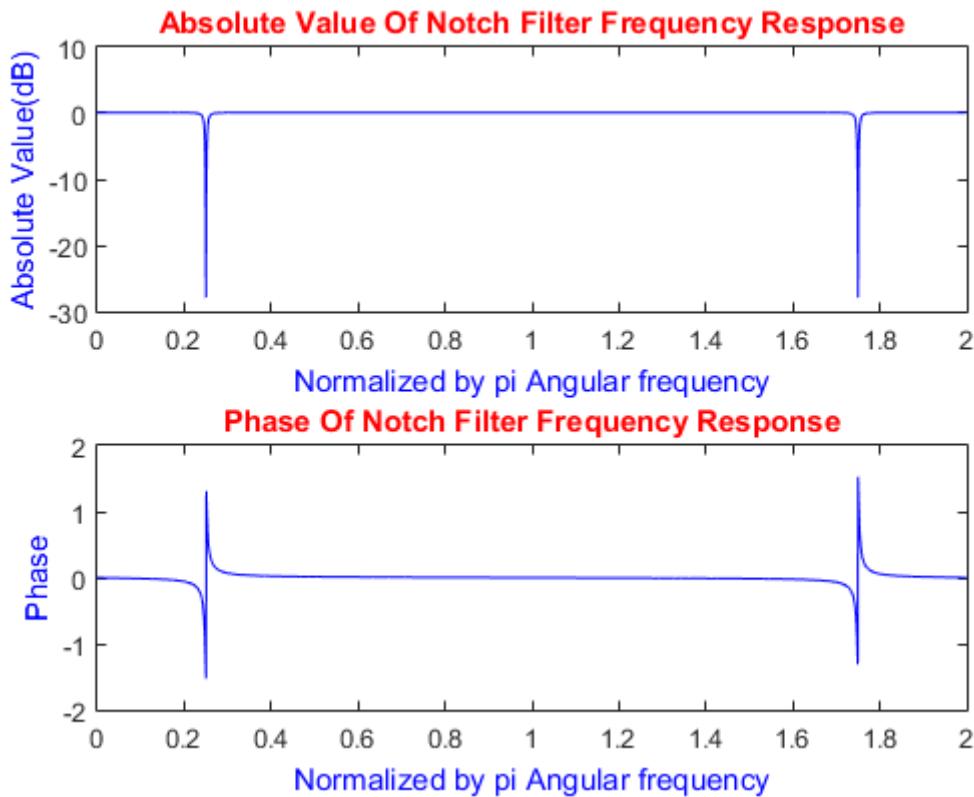
تغییر کرده و در قسمت‌های دیگر مقادیر یکنواخت‌تر و ثابت‌تری نسبت به متناظر IIR خود دارد. ضمناً مشخص است که در هر دو نوع فیلتر، با افزایش مرتبه خاصیت فیلتری تقویت می‌شود که کاملاً مورد انتظار است.

۹. همان‌طور که می‌دانیم ویژگی فیلتر notch این است که یک تک فرکانس را حذف نموده و روی بقیه فرکانس‌ها اثر بسیر ناچیزی بگذارد و حتی‌الامکان آن‌ها را تغییر ندهد. برای حذف یک فرکانس خاص، باید در آن فرکانس روی دایره واحد یک صفر قرار داد. با توجه notch خواسته‌شده ($\omega = 45^\circ$) برای حقیقی ماندن سیستم باید علاوه بر صفری که در $\frac{\pi}{4}$ قرار می‌دهیم، صفری نیز در مختصات مزدوج صفر اخیر یعنی بر روی دایره واحد و $\frac{\pi}{4}$ قرار دهیم. برای اینکه بر روی فرکانس‌های دیگر اثری نبینیم، لازم است یک قطب به فاصله بسیار کم از هر صفر قرار دهیم. (هر دو قطب باید مزدوج باشند تا سیستم و فیلتر حقیقی بماند). به این ترتیب فیلتر IIR خواهد شد. همچنان دقت می‌کنیم قطب‌ها با فاصله اندک درون دایره واحد قرار داشته باشند تا سیستم پایدار بماند. نتیجه از pezdemo در شکل زیر آورده شده‌است. در MATLAB نیز همین فیلتر notch را پیاده‌سازی کرده‌ایم.

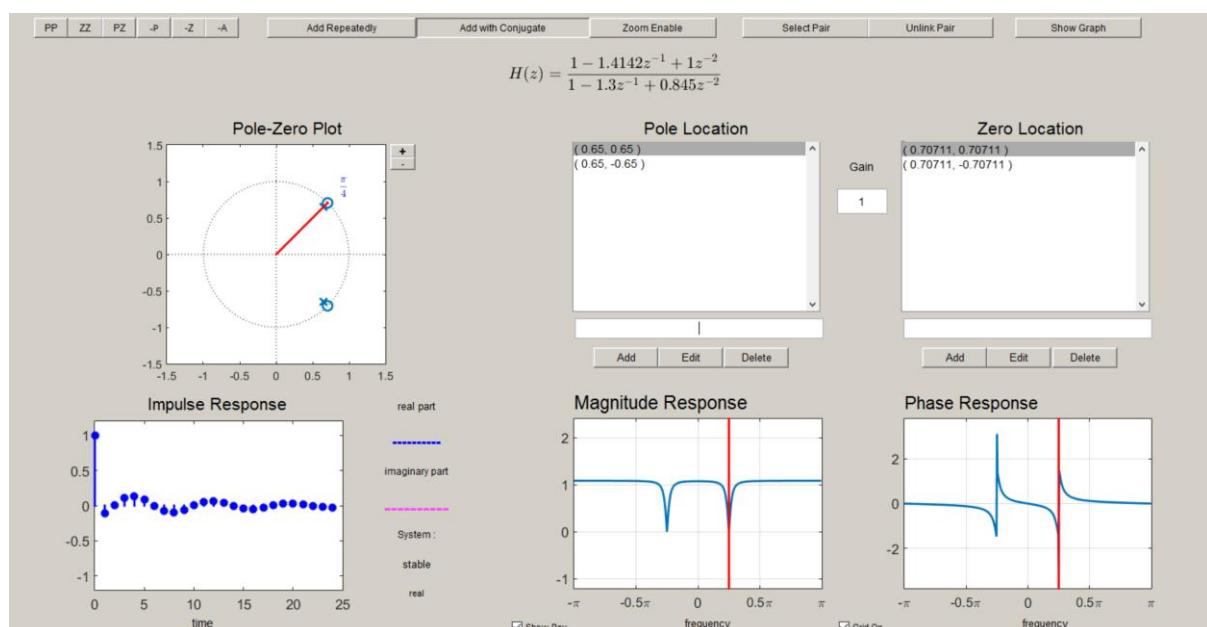


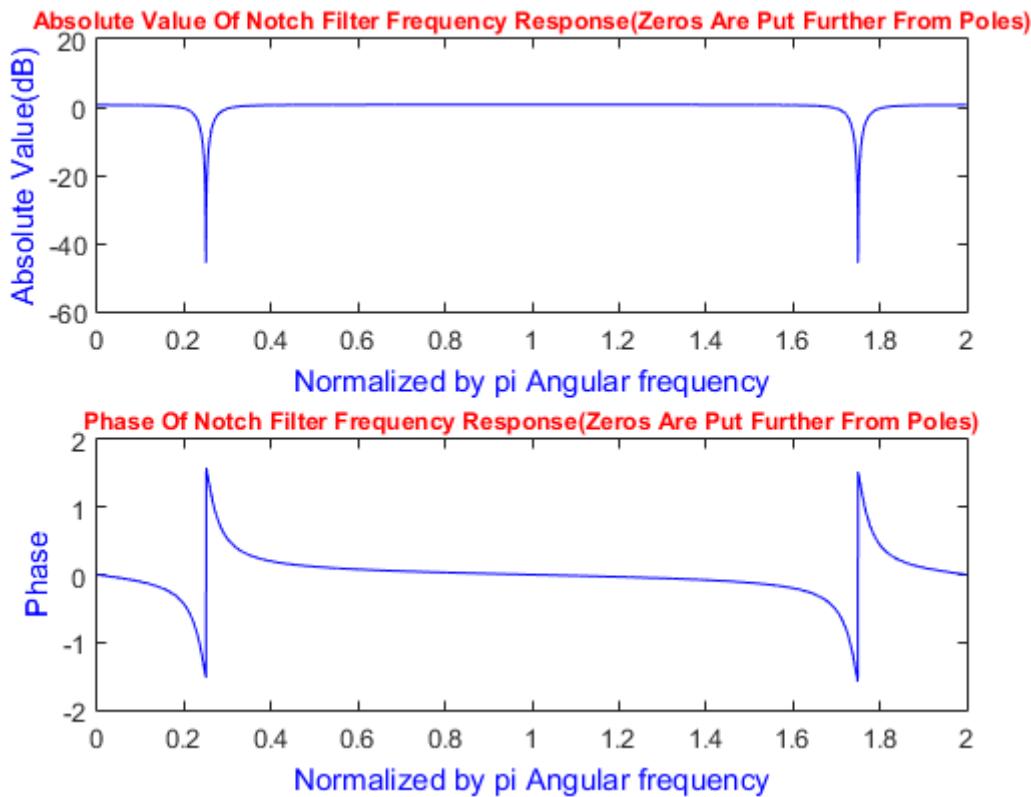
تبدیل Z در شکل فوق مشخص شده‌است.

۱۰. توضیح کد: ابتدا در کامنت‌ها می‌توان مکان صفر و قطب‌ها را ملاحظه کرد. سپس ضرایب صورت فیلتر notch و پس از آن ضرایب مخرج از رویتابع تبدیل داده می‌شوند. سپس با دستور freqz پاسخ فرکانسی فیلتر و فرکانس‌هایی که در آن پاسخ فرکانسی محاسبه می‌شوند با گرفتن ضرایب صورت و مخرج فیلتر notch بدست می‌آیند. پس از آن نیز اندازه و فاز پاسخ فرکانسی فیلتر رسم می‌شوند. (شکل زیر)



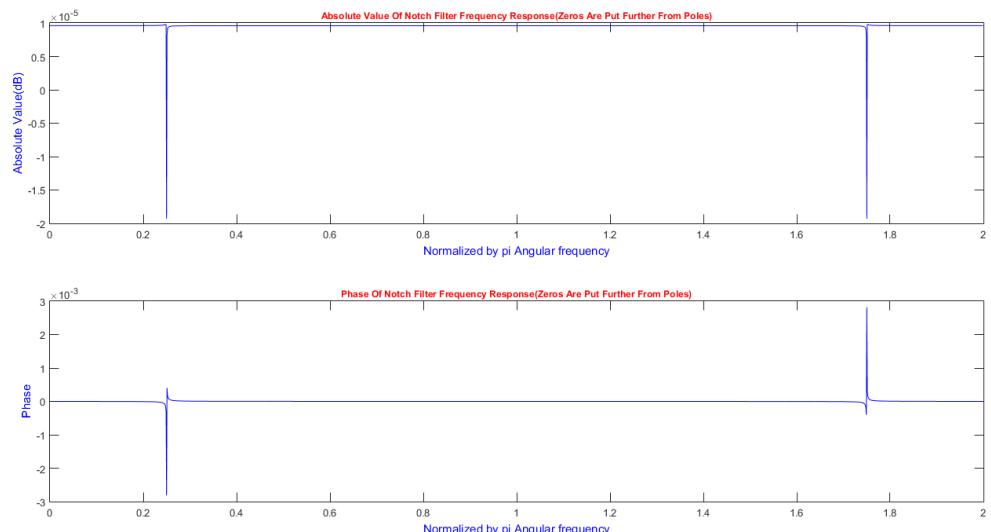
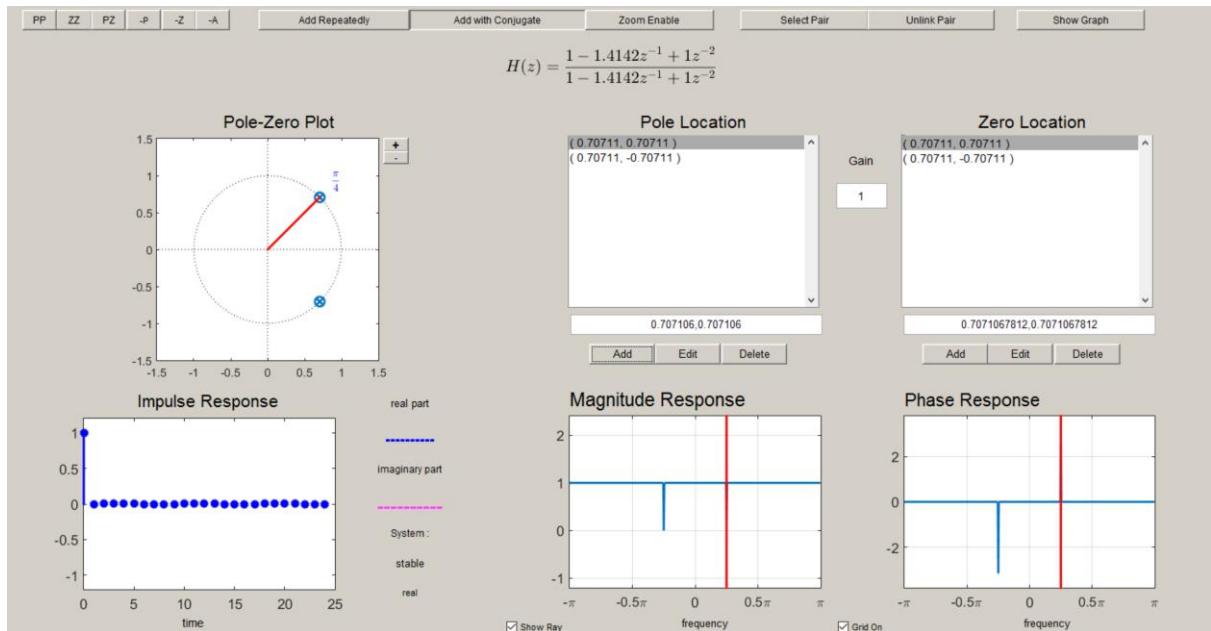
در ادامه به بررسی تغییر مکان صفر و قطب‌ها می‌پردازیم. البته با توجه به این‌که واضح است با تغییر مکان صفرها فیلتر notch دیگر فرکانس مورد نظر را حذف نمی‌کند، فقط مکان قطب‌ها را عوض می‌کنیم. ابتدا اثر دور شدن از صفرها را بررسی می‌کنیم. مانند قبل از pezdemo مکان‌ها را تعیین می‌کنیم و سپس ضرایب را از آن ذخیره می‌کنیم و با freqz پاسخ فرکانسی را محاسبه می‌کنیم. نهایتاً نیز آن را رسم می‌کنیم. تصویر و پاسخ فرکانسی این حالت در تصاویر زیر مشاهده می‌شود.





ملاحظه می شود با دور شدن بیشتر قطب‌ها از صفرها، خاصیت notch بودن کمتر شده و فرکانس‌های دیگری نیز فیلتر می‌شوند که از ایده‌آل بودن فیلتر می‌کاهد. همچنین تغییرات فاز نیز در بازه بزرگتری در حال اتفاق افتادن است.

اثر نزدیک کردن بیش از حد قطب‌ها به صفرها : انتظار می‌رود که در صورت نزدیکی بسیار زیاد قطب‌ها به صفرها، اثر notch تقریبا از بین برود یا به وضعیتی غیر از وضعیت مطلوب ما برسد. (در حالت حدی که روی هم باشند، اثر یکدیگر را به طور کامل خنثی می‌کنند پس می‌توان انتظار داشت از جایی به بعد خاصیت فیلتر notch از بین برود یا تغییر کند). مانند قبل ابتدا در pezdemo قطب‌ها را بسیار نزدیک کرده و ضرایب تابع تبدیل را از pezdemo به عنوان ضرایب فیلتر ذخیره می‌کنیم. (البته برای پایدار ماندن سیستم و فیلتر قطب‌ها را درون دایره واحد نگه می‌داریم). نهایتا با freqz پاسخ فرکانسی را محاسبه و سپس آن را رسم می‌کنیم. نتایج در تصاویر زیر مشهود است. (دقت می‌کنیم در این حالت باید خودمان ضرایب را حساب کنیم و ذخیره کنیم چون تعداد ارقام pezdemo کافی نمی‌باشد).



مشاهده می‌شود خاصیت notch تا حد زیادی از بین رفته است. همانطور که مشاهده می‌شود در واقع حذفی صورت نمی‌گیرد و تقریباً همه فرکانس‌ها با شدت یکسانی عبور می‌کنند.

قسمت دوم: تصحیح سیگنال صوتی با فیلترهای مورد نظر

۱. در واقع در روش fft سری فوریه به جای تبدیل فوریه محاسبه می‌شود. رابطه‌ای که مطلب با آن تبدیل فوریه را محاسبه می‌کند، از قسمت dct مطلب بدست‌آمده است:

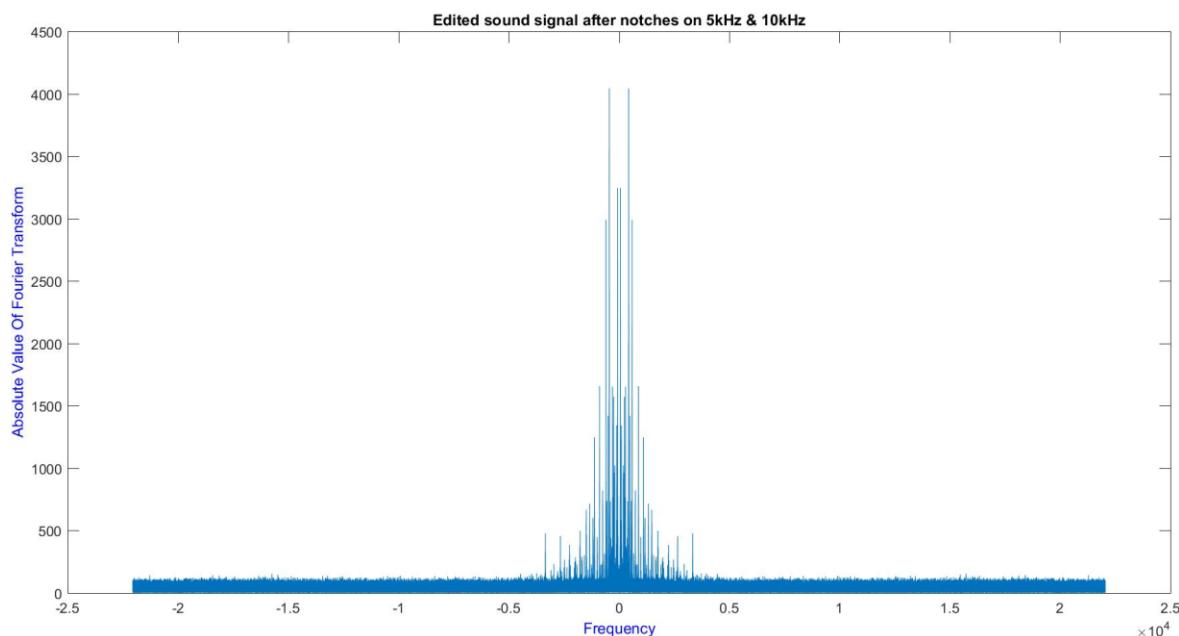
$$X(k) = \sum_{j=1}^N x(j) \omega_N^{(j-1)(k-1)}$$

$$x(j) = (1/N) \sum_{k=1}^N X(k) \omega_N^{-(j-1)(k-1)}$$

بنابراین به نظر می‌رسد مطلب سیگنال گستته در زمانی را که می‌گیرد به عنوان سیگنالی گستته و متناوب در نظر می‌گیرد و ضرایب سری فوریه آن را محاسبه می‌کند و در صورتیکه تعداد داده‌ها زیاد باشد، سری فوریه به تبدیل فوریه میل می‌کند چون فواصل ضرایب از هم کم شده و از نظر ما پیوسته به نظر می‌رسد. با توجه به رابطه بالا می‌توانیم بین ضرایب سری فوریه و آن چیزی که مطلب بدست می‌دهد رابطه‌ای بیاییم.

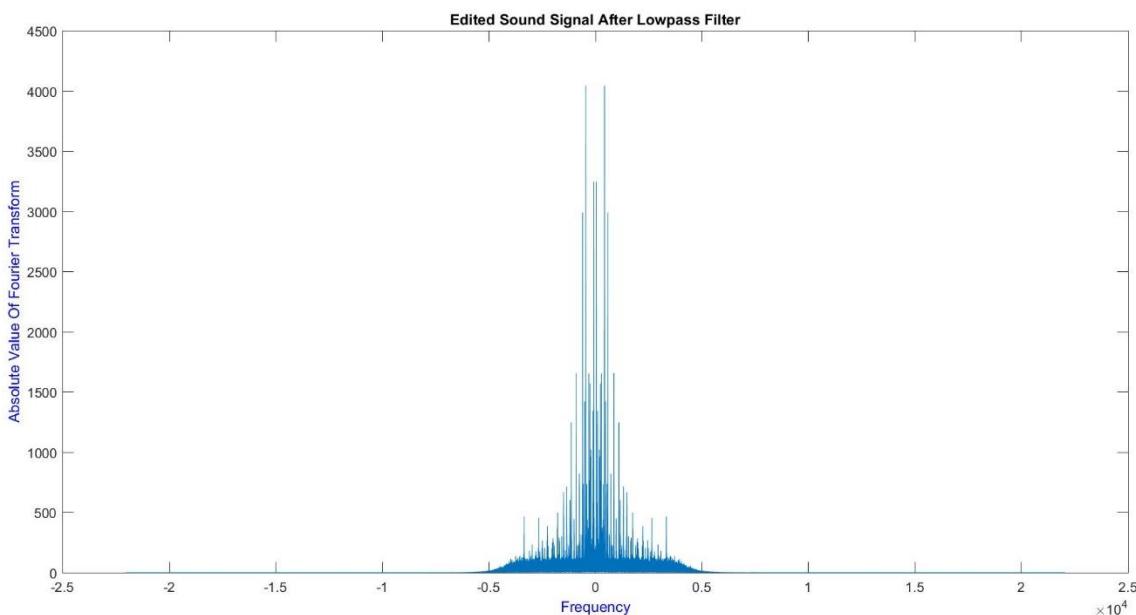
$$Na_k = X(k)$$

۲. با توجه به تبدیل فوریه سیگنال، که در قسمت قبل آورده شد، مشاهده می‌شود که تک فرکانس‌های موجود، روی فرکانس‌های ۵kHz و ۱۰kHz هستند که باید حذف شوند. برای حذف این فرکانس‌ها، به کمک فیلتر Notch، از تابع `iirnotch` در مطلب استفاده می‌کنیم. پارامترهایی که این تابع می‌پذیرد، فرکانسی که می‌خواهیم حذف شود و نیز پهنه‌ای باند -3dB است. پارامتر اول، باید به صورت نرم‌الایز شده باشد، به این



معنی که باید فرکانس مطلوب را بر نصف فرکانس نمونه برداری تقسیم کنیم و به تابع بدھیم؛ در انتخاب پارامتر دوم، با استفاده از روش آزمون و خطا، پهنانی باند -3dB را برابر فرکانس Notch نرمالایز شده، به ترتیب تقسیم بر 500 و 1000 قرار داده ایم؛ این اعداد با این معیار انتخاب شده اند که دامنه های بزرگ روی فرکانس های 5kHz و 10kHz و همچنین دامنه های بزرگ اطراف آنها حذف شوند؛ اگر این پهنانی باند را کمتر بگیریم، دامنه های بزرگ اطراف کاملا حذف نمی شوند و اگر بیشتر بگیریم، روی فرکانس های دورتر هم تاثیر کاهش دامنه خواهیم داشت که برایمان چندان مطلوب نیست.

۳. با توجه به تبدیل فوریه‌ی سیگنال که در دو سوال قبل رسم شد، تخمینی که از بیشینه‌ی فرکانس صوت می‌زنیم برابر 4kHz است. برای طراحی فیلتر پایین گذر از تابع butter استفاده می‌کنیم. مرتبه‌ی این فیلتر را برابر 8 و فرکانس قطع را (که باید به صورت نرمالایز شده باشد) برابر خارج قسمت 4kHz بر نصف فرکانس نمونه برداری (44100Hz) قرار می‌دهیم. علت اینکه مرتبه‌ی فیلتر را برابر ۸ قرار می‌دهیم این است که در مرتبه‌های پایین تر، دامنه‌های خارج از 4kHz مقدار قابل توجهی دارند و ایده آل ما، صفر کردن دامنه‌های خارج از 4kHz است. پس از اعمال این فیلتر تبدیل فوریه به صورت شکل زیر خواهد شد :



4. اگر $x[n]$ ورودی سیستم تاخیر دهنده و $y[n]$ خروجی آن باشد، رابطه‌ی آنها به صورت زیر خواهد بود:

$$y[n] = x[n] + \alpha \times x[n - \beta]$$

که در آن α ضریب (بدون واحد) و β زمان تاخیر (بر حسب تعداد نقطه) می‌باشد. لذا تبدیل Z سیستم تاخیر دهنده بصورت زیر خواهد شد :

$$H(z) = \frac{Y(z)}{X(z)} = (1 + \alpha \times z^{-\beta})$$

با توجه به تمرین اول، مقدار ضریب تاخیر برابر ۰.۳۸۶۴۴۸۵۹۸۱ و نیز مقدار زمان تاخیر برابر ۳ ثانیه شد. لذا مقدار α همان مقدار ضریب تاخیر خواهد شد اما مقدار β برابر زمان تاخیر ضربر فرکانس نمونه برداری خواهد شد.

با توجه به تبدیل زد بالا، تبدیل زد معکوس سیستم تاخیردهنده بصوت زیر خواهد شد :

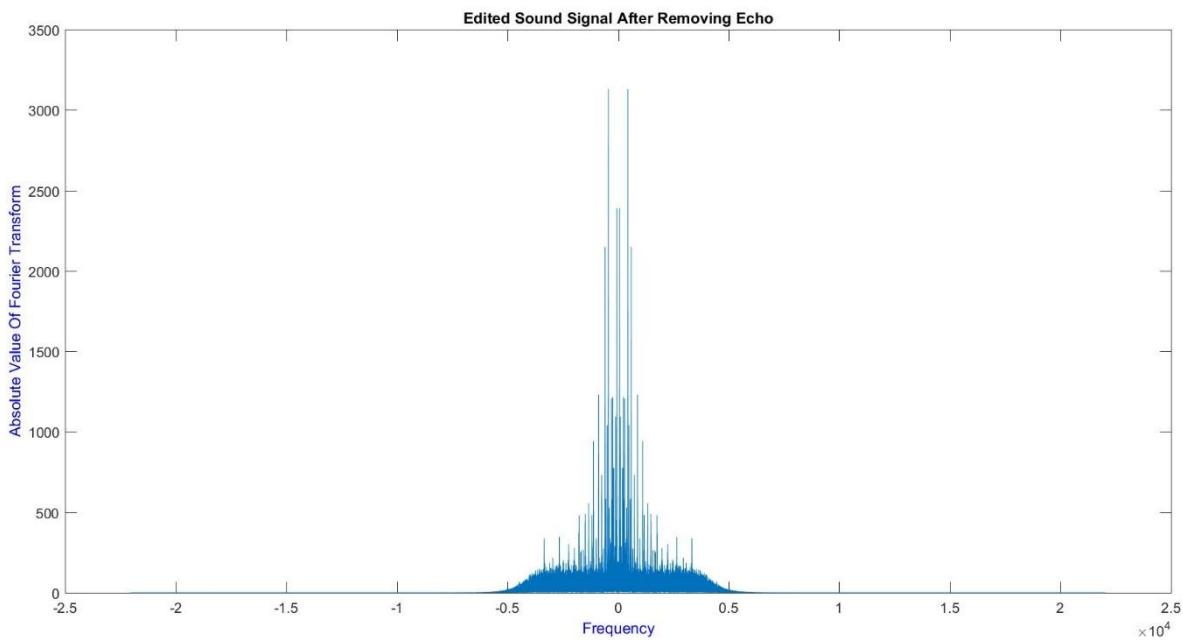
$$H^{-1}(z) = \frac{1}{1 + \alpha \times z^{-\beta}}$$

حال برای اعمال این فیلتر روی سیگنال، از تابع filter در متلب استفاده می‌کنیم. ورودی‌های این تابع، ضرایب چند جمله‌ای صورت و مخرج تابع تبدیل و نیز دنباله‌ای تحت عنوان سیگنال ورودی است. صورت کسر تابع تبدیل که ۱ است، ولی مخرج کسر چند جمله‌ای از مرتبه‌ی $f_s \times \beta$ می‌باشد که فقط جمله‌ی از همین مرتبه و مرتبه‌ی صفر را شامل می‌شود، لذا ضرایب چند جمله‌ای مخرج، به صورت یک بردار به طول $1 + \alpha \times f_s \times \beta$ می‌باشد که درایه‌ی اول آن ۱، درایه‌ی آخر آن α و بقیه‌ی درایه‌های آن صفر هستند. از آنجا که این فیلتر دارای قطب نیز هست، لذا فیلتر IIR می‌باشد.

پیاده‌سازی: با توجه به اینکه پاسخ ضریب فیلتر IIR شامل بینهایت جمله می‌باشد و برای یافتن خروجی فیلتر باید حاصل کانولوشن این بینهایت جمله با سیگنال محدود ورودی بدست آید، به نظر می‌رسد که پیاده‌سازی آن با مشکل مواجه باشد. روش متلب برای پیاده‌سازی آن یک روش تقریبی است. بدین صورت که فرض می‌کند خروجی برای زمان‌های منفی صفر بوده است.(و بنابراین به تعداد ورودی خروجی می‌دهد و از رابطه زیر به صورت بازگشتی با داشتن فرض‌های فوق فیلتر شده‌ی هر ورودی را بدست می‌دهد.

$$a_0 y[n] = \sum_{k=1}^{M-1} a_k y[n-k] + \sum_{k=0}^{N-1} b_k x[n-k]$$

پس از اعمال این فیلتر، تبدیل فوریه‌ی سیگنال به صورت شکل زیر خواهد شد :



قسمت سوم: تغییر نرخ نمونه برداری

۱. اگر تبدیل z سیگنال $X(z)$ باشد، طبق تعریف تبدیل z سیگنال $y[n]$ را می‌یابیم. چون $x_{(L)}[n]$ فقط به ازای $n = mL$ مقدار غیر صفر دارد:

$$Y(z) = X_{(L)}(z) = \sum_{n=-\infty}^{\infty} x_{(L)}[n]z^{-n} = \sum_{m=-\infty}^{\infty} x[m]z^{-mL} = X(z^L)$$

اگر R ناحیه همگرایی $X(z)$ باشد داریم:

$$R.O.C(Y) = R^{\frac{1}{L}}$$

در ادامه برای بدست آوردن تبدیل فوریه، به جای Z در رابطه تبدیل Z ، $e^{j\omega}$ قرار می‌دهیم. بدین ترتیب تبدیل فوریه خواهد بود:

$$Y(j\omega) = X(e^{jL\omega})$$

۲. اگر تبدیل z سیگنال $X(z)$ باشد، طبق تعریف تبدیل z سیگنال $y[n]$ را می‌یابیم. چون $y[n] = x[nL]$ است داریم (تغییر متغیر از n به $m = nL$)

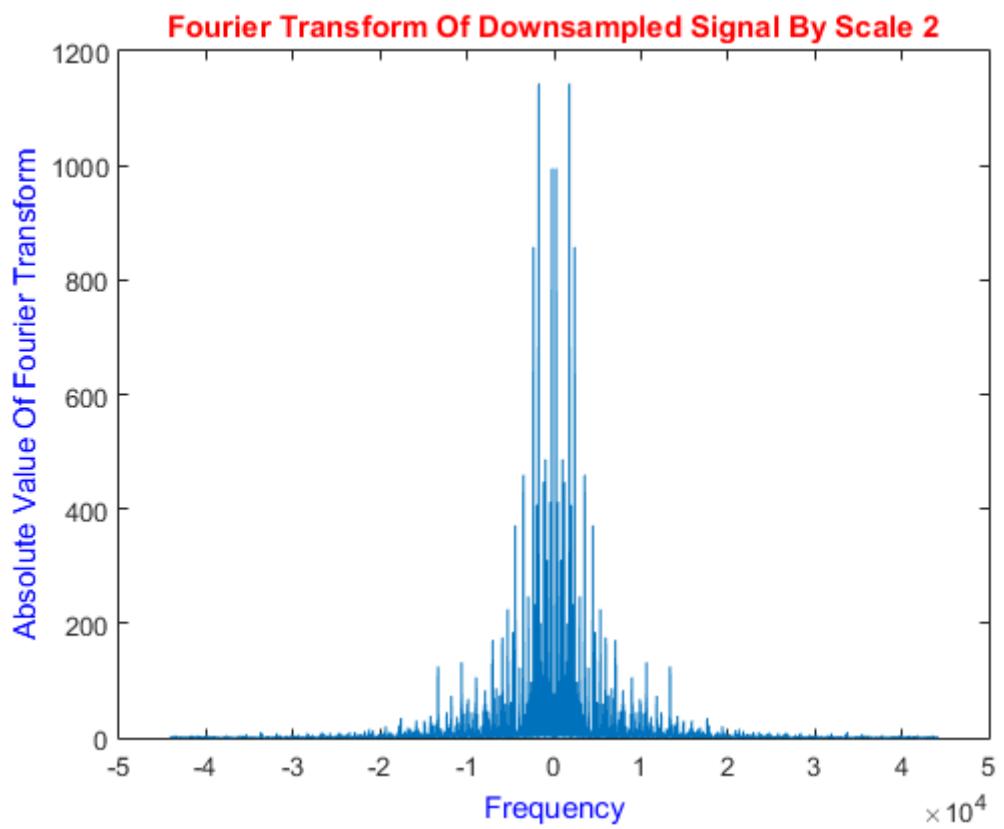
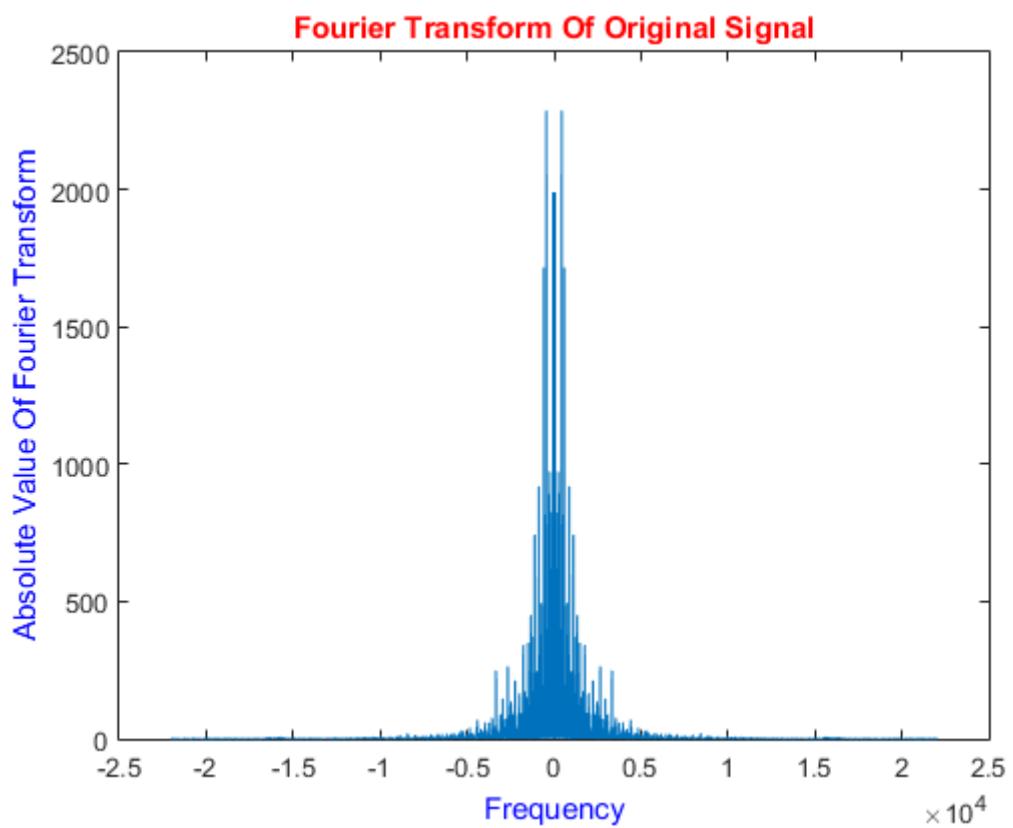
$$Y(z) = \sum_{n=-\infty}^{\infty} x[nL]z^{-n} = \sum_{m=-\infty}^{\infty} x[m]z^{-m/L} = X(z^{1/L})$$

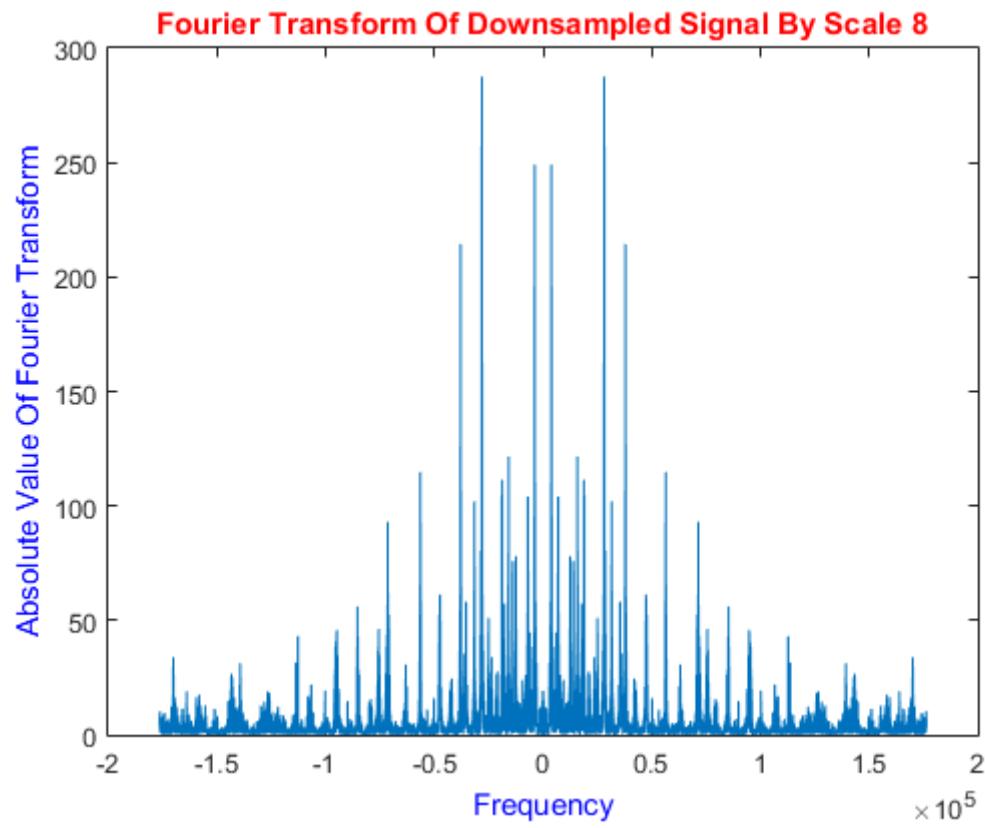
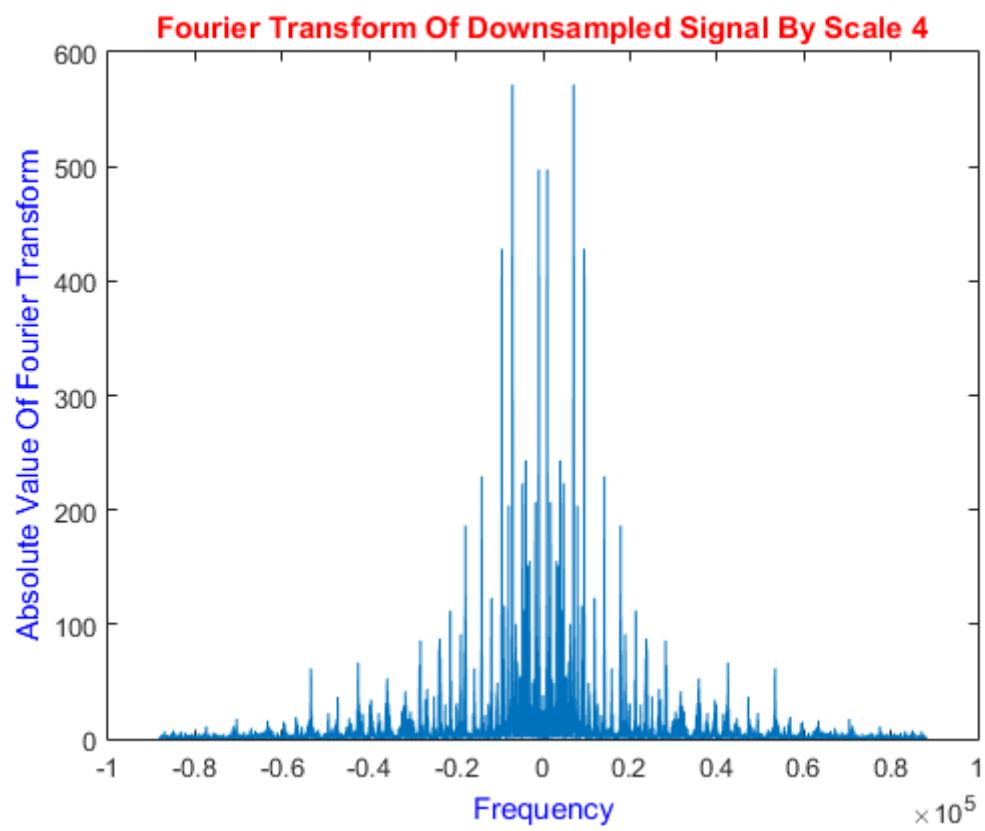
اگر R ناحیه همگرایی $X(z)$ باشد داریم:

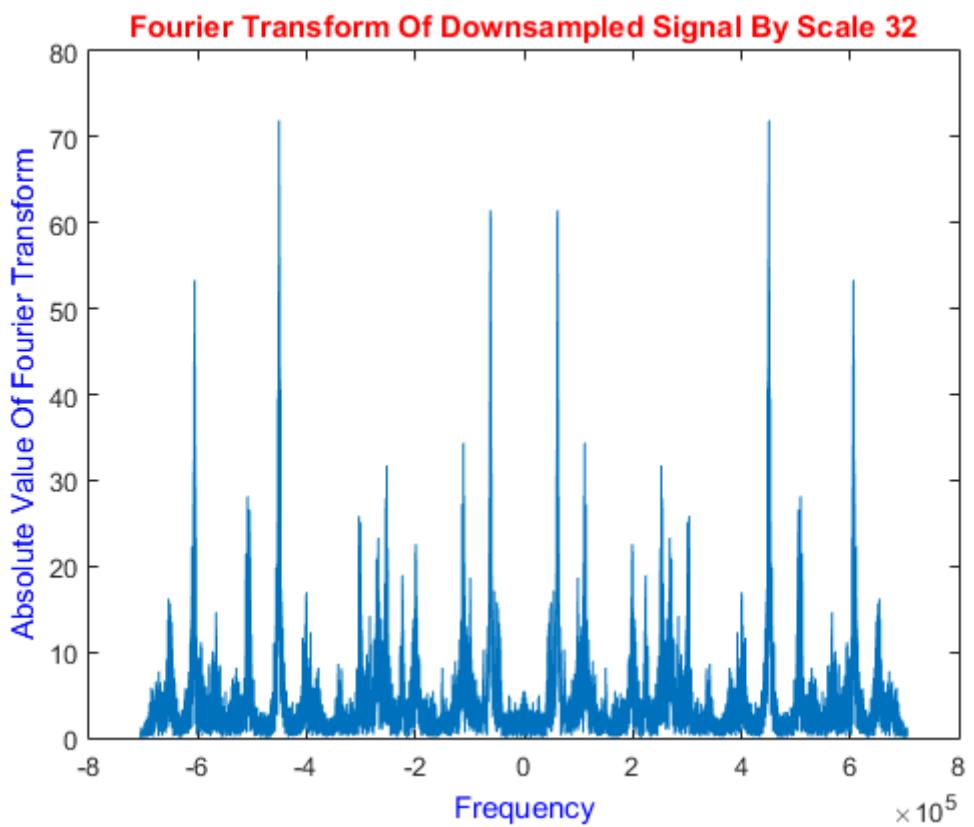
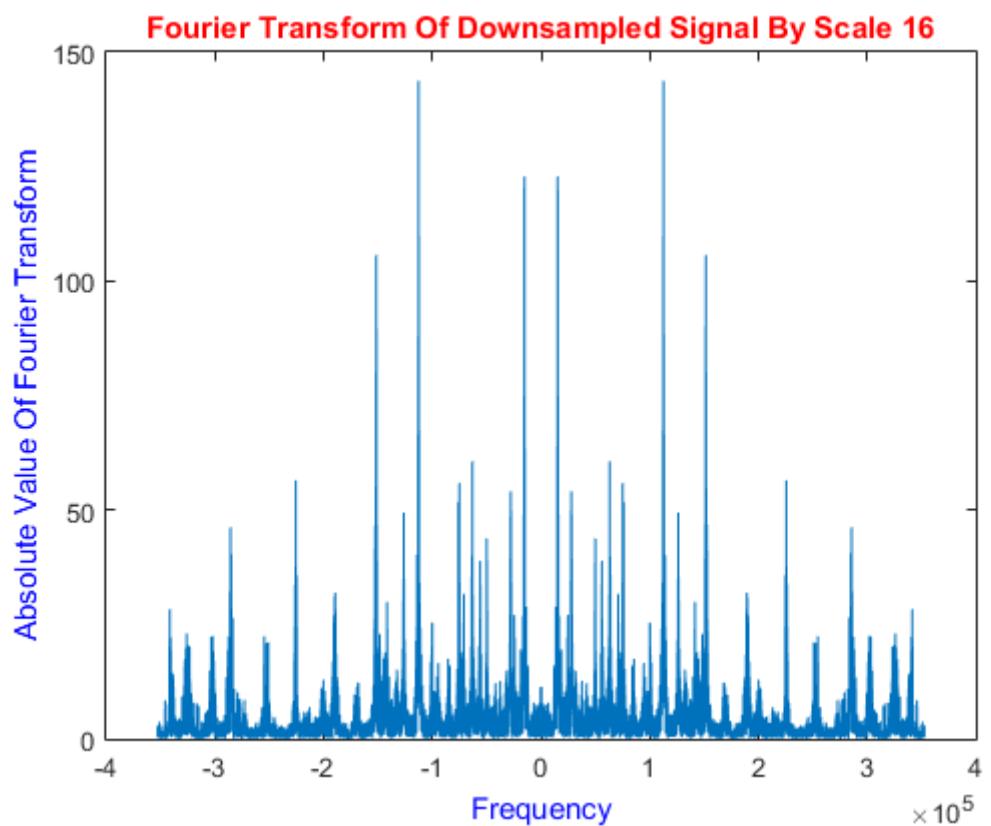
$$R.O.C(Y) = R^L$$

۳. دستور *downsample* سیگنال ورودی را به همراه عددی که باید به آن اندازه نرخ نمونه برداری را کاهش دهد می‌گیرد و اگر این عدد برابر L باشد، از هر L نمونه پشت سرهم فقط اولی را برمی‌دارد. تبدیل فوریه سیگنال *downsample* شده را در تصاویر زیر مشاهده می‌کنیم. برای بدستآوردن تبدیل فوریه و رسم آن با اندکی تغییر از تابعی استفاده می‌کنیم که در تمرین سری ۳ آنرا ساختیم. (تابع *FFT*) این تابع در ورودی خود سیگنال و فرکانس نمونه برداری آنرا به همراه یک *flag* برای رسم کردن یا نکردن تبدیل فوریه دریافت می‌کند. طول سیگنال را بدست آورده و اولین توان ۲ بعد از طول سیگنال را می‌یابد. سپس تبدیل فوریه را با طول عدد اخیر بدست می‌دهد. (با دستور *fft*) سپس برای مقایرن نمودن تبدیل فوریه از دستور *fftshift* استفاده می‌نماییم. فرکانس‌های متقارن را نیز تعریف می‌کنیم. در نهایت اگر یک *flag* باشد تبدیل فوریه متقارن نیز رسم می‌شود.

توضیح کد: ابتدا با دستور *audioread* فایل صوتی خواسته شده خوانده می‌شود. سپس برداری برای مقیاس‌های جدید ساخته می‌شود. فرکانس نمونه برداری جدید نیز برای *downsample* پخش سیگنال‌های جدید ساخته می‌شود. (با تقسیم فرکانس نمونه برداری اصلی بر مقیاس *downsample* سپس نرخ نمونه برداری سیگنال‌ها با دستور *downsample*، به مقیاس گفته شده کاهش می‌یابد. در ادامه نیز تبدیل فوریه سیگنال اصلی و سیگنال‌های جدید ساخته شده از آن رسم می‌شوند. دقت می‌کنیم که فرکانس نمونه برداری برای رسم تبدیل فوریه هر بار عوض می‌شود. (ضرب می‌شود در مقیاس *downsampling*) نتایج آن در تصاویر زیر قابل مشاهده است. در انتهای کد این بخش نیز با کنترل‌هایی که به کاربر داده شده امکان پخش و شنیدن صوت هریک از سیگنال‌ها فراهم شده است.







همان‌طور که مشاهده می‌شود با هر بار کاهش نرخ نمونه برداری، دامنه مقادیر نصف شده و محدوده فرکانسی دو برابر می‌شود. علت نصف شدن بهره نصف شدن تعداد مجموعهایی است که برای تولید پاسخ فرکانسی هر

فرکانس در رابطه تبدیل فوریه ظاهر می‌شوند. علت تغییر محدوده فرکانس نیز عوض شدن فرکانس نمونه برداری متناسب با کاهش نرخ نمونه برداری است.

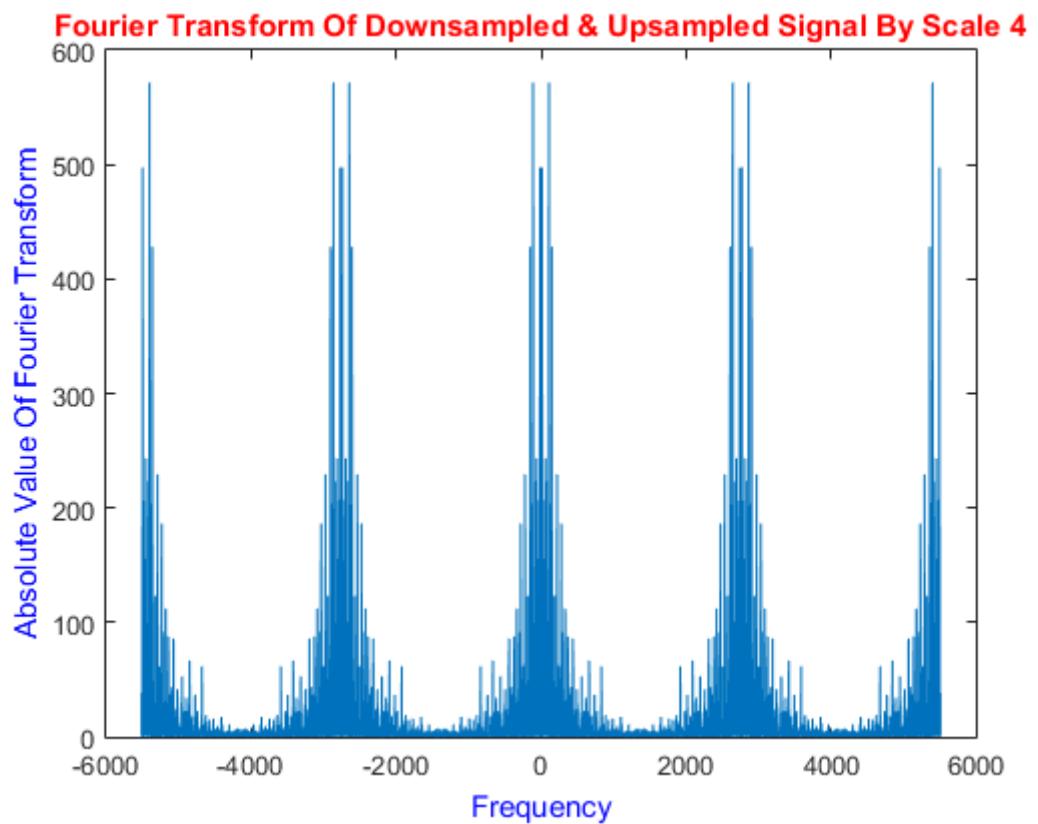
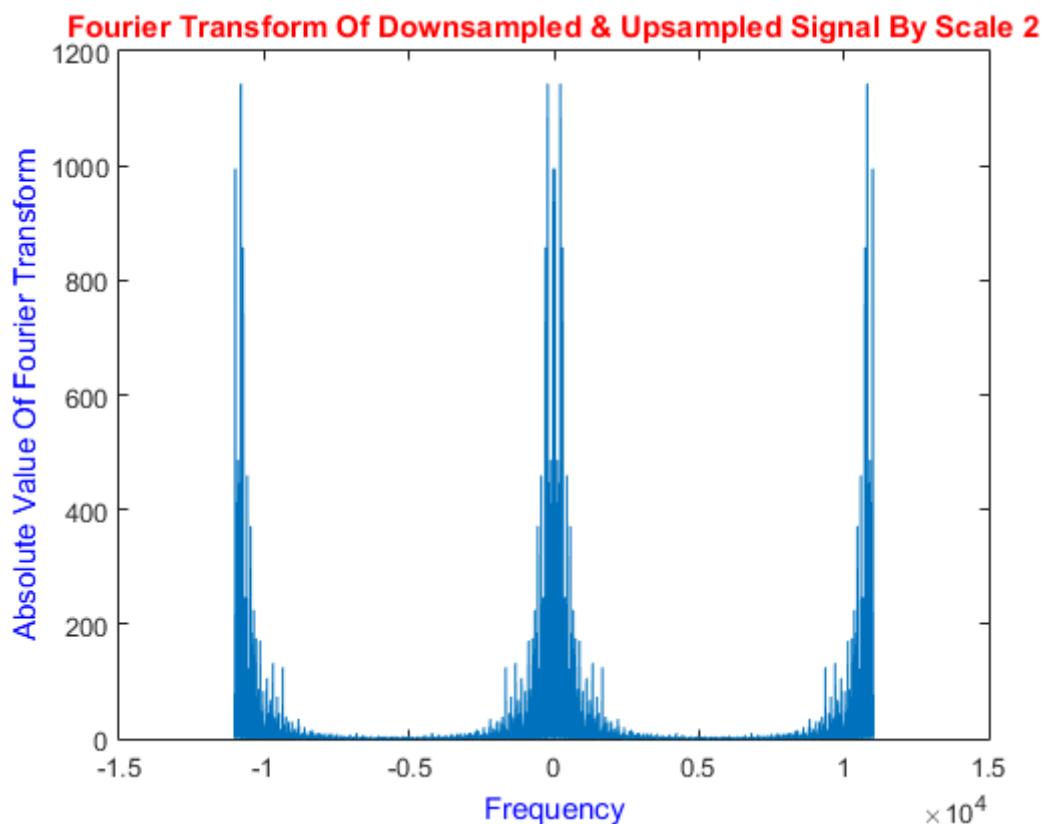
می‌دانیم که با *downsampling* در واقع نرخ نمونه برداری را کاهش می‌دهیم. لذا در صورتیکه که این نرخ آن قدر کاهش یابد که شرط نایکوییست نقض شود، پدیده اختلاط فرکانسی رخ می‌دهد و صوت شنیده شده خراب خواهد بود و هرچه اختلاط فرکانسی بیشتر باشد تفاوت صوت اصلی و صوت نمونه برداری شده با نرخ کمتر بارزتر می‌شود. با توجه به اینکه اکثر فرکانس‌های دارای محتوا در کمتر از ۴ یا ۵ کیلوهرتز قرار دارند، بنابراین نرخ نایکوییست حدود ۱۰ کیلوهرتز بوده و با توجه به فرکانس نمونه برداری اصلی (44100) اگر بیش از ۲ بار نرخ نمونه برداری را کاهش دهیم، صدای شنیده شده دچار اختلاط فرکانسی نسبتاً زیادی خواهد شد. در اینجا به ازای کاهش نرخ نمونه برداری به ازای مقیاس ۸ و بیشتر اثرات نامطلوبی دارد که می‌توان با گوش دادن به آن‌ها پی برد.

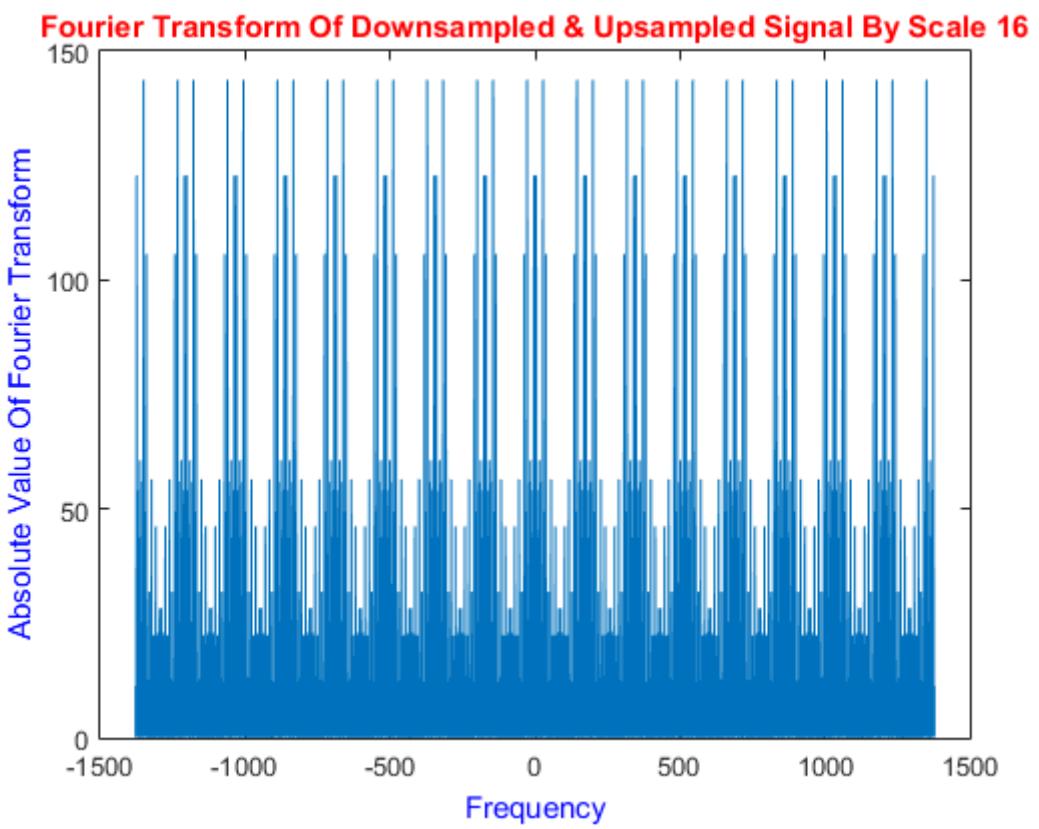
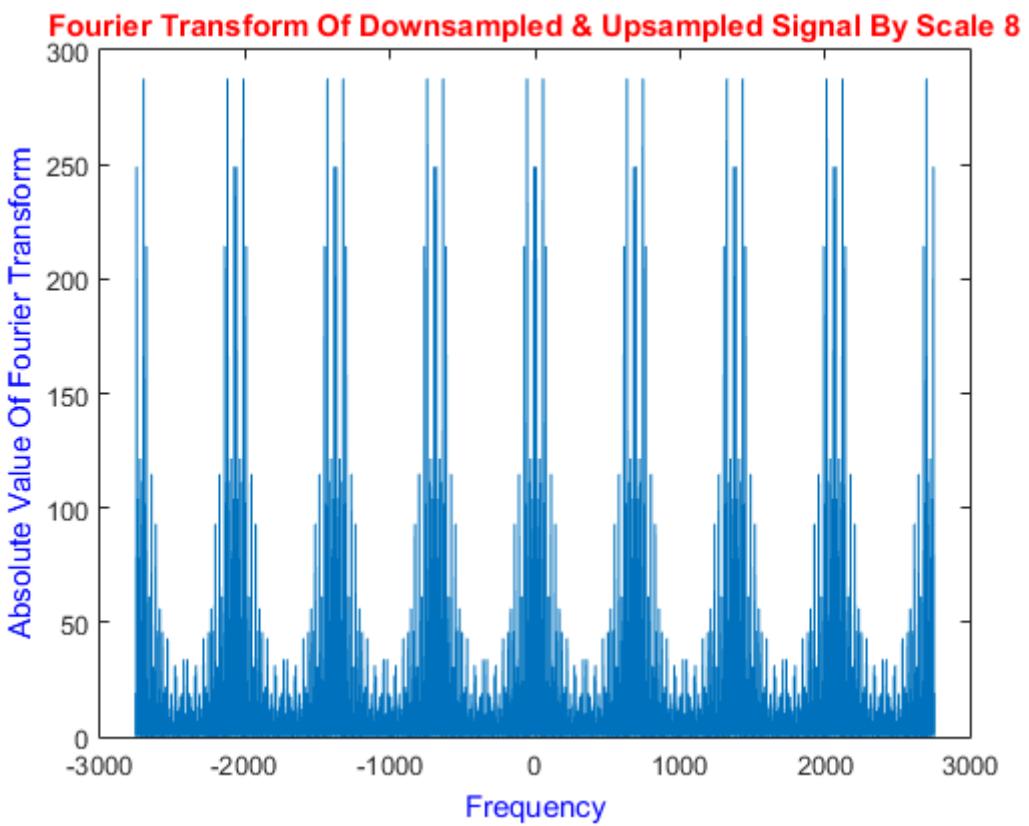
مشاهده می‌شود در هر بار کاهش نرخ نمونه برداری، اندازه‌های تبدیل فوریه تقریباً نصف می‌شوند. علت این است که اگر با مقیاس ۲ نرخ نمونه برداری را کاهش دهیم، از هر ۲ نمونه یکی برداشته می‌شود و بنابراین در رابطه تبدیل فوریه تعداد داده‌ها نصف شده و از آنجایی که دو داده نزدیک هم مقادیر نزدیکی دارند، در رابطه تبدیل فوریه برای هر فرکانس تقریباً به نصف مقدار قبلی می‌رسیم. (از هر دو جمله نزدیک به هم تشکیل دهنده مقدار تبدیل فوریه در یک نقطه، یکی حذف شده است). در واقع به نظر می‌رسد داریم محور افقی را می‌کشیم و در عین حال با نصف کردن نرخ نمونه برداری، مقادیر تبدیل فوریه حدوداً نصف می‌شوند.

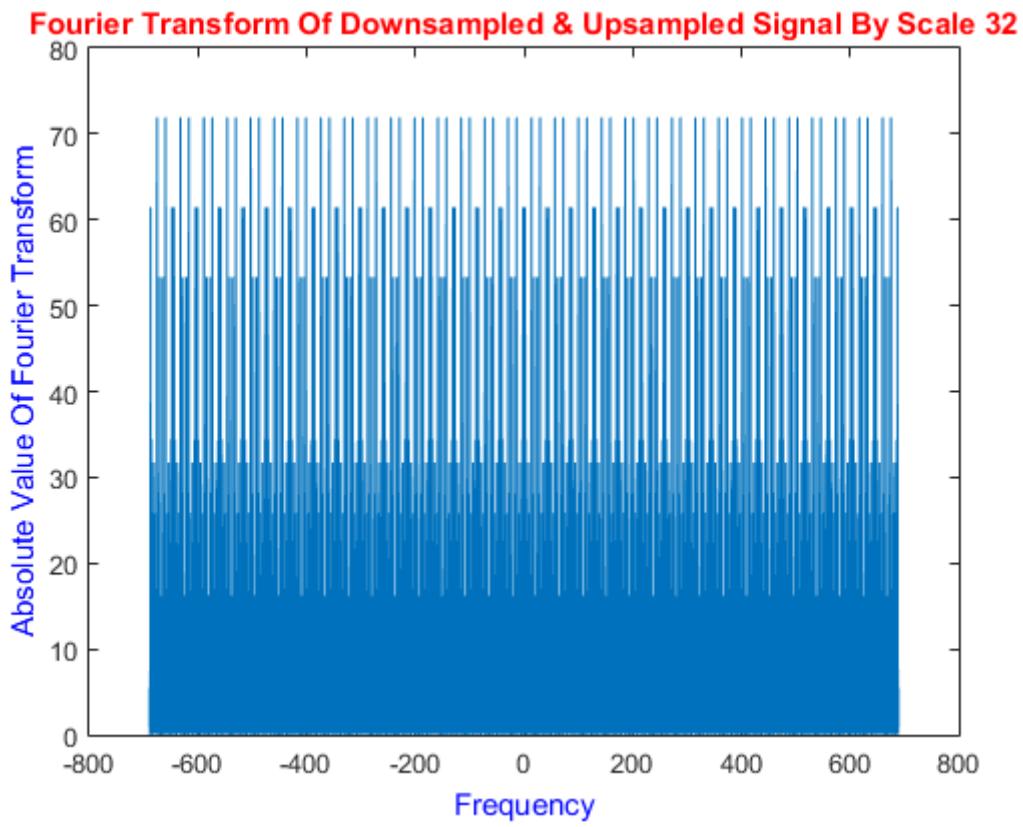
برای گوش دادن هر صوت باید یک شی *audioplayer* تعریف کرد. این شی در ورودی خود سیگنال را به همراه فرکانس نمونه برداری آن می‌گیرد و امکان پخش و نگهداشتن و قطع و دیگر موارد را به کاربر می‌دهد. برای این‌که یک صوت *downsampled* را گوش دهیم، لازم است فرکانس نمونه برداری متناظر همان را به شی بدهیم. در غیر این صورت سیگنال با سرعتی غیر صحیح پخش خواهد شد. برای مثل درصورتیکه صوت *downsampled* با مقیاس ۴ را با فرکانس نمونه برداری صوت *downsampled* با مقیاس ۲ پخش کنیم، سرعت آن دو برابر حالت معمولی خواهد بود.

در صورتیکه صوت‌ها را با فرکانس نمونه برداری مناسب آن‌ها پخش کنیم، مشاهده می‌کنیم هرچه مقیاس *downsample* بیشتر باشد، صدا بمتر و ضعیفتر است. علت ضعیفتر شدن همان تقسیم بر ۲ شدن‌هایی است که در بالا توضیح داده شد. در هر بازه انرژی که می‌گیریم، حدوداً نصف حالت قبل است. علت بمتر شدن نیز این است که با هر بار *downsampling* محتوای فرکانس‌های بالا کم می‌شود و در نتیجه‌ی آن، صدای بم شنیده می‌شود. اثر تضعیف (تقسیم بر ۲ شدن‌های متوالی) بر فرکانس‌های زیاد در تصاویر فوق نیز مشاهده می‌شود که بمتر شدن نتیجه را توجیه می‌کند.

۴. توضیح کد: ابتدا همه سیگنال‌ها به مقدار گفته شده *upsample* می‌شوند. سپس رسم شده و در نهایت با کنترل قرار داده شده قابل پخش هستند. تصویر نتایج در زیر مشاهده می‌شود.





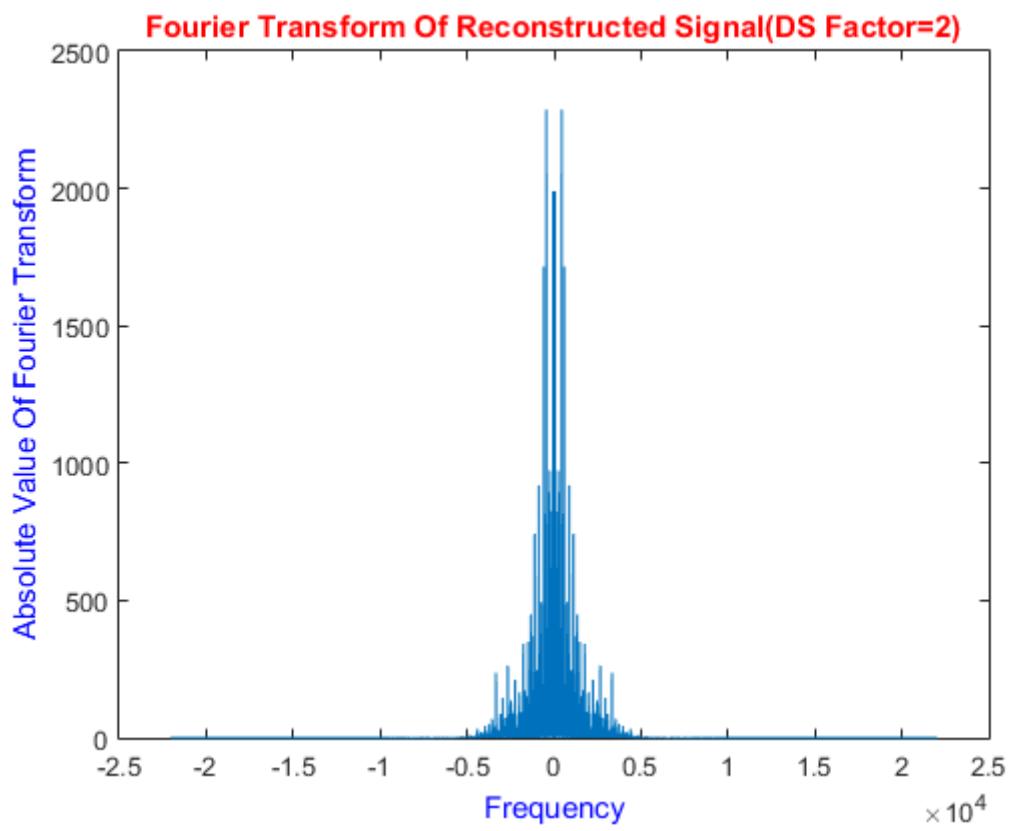
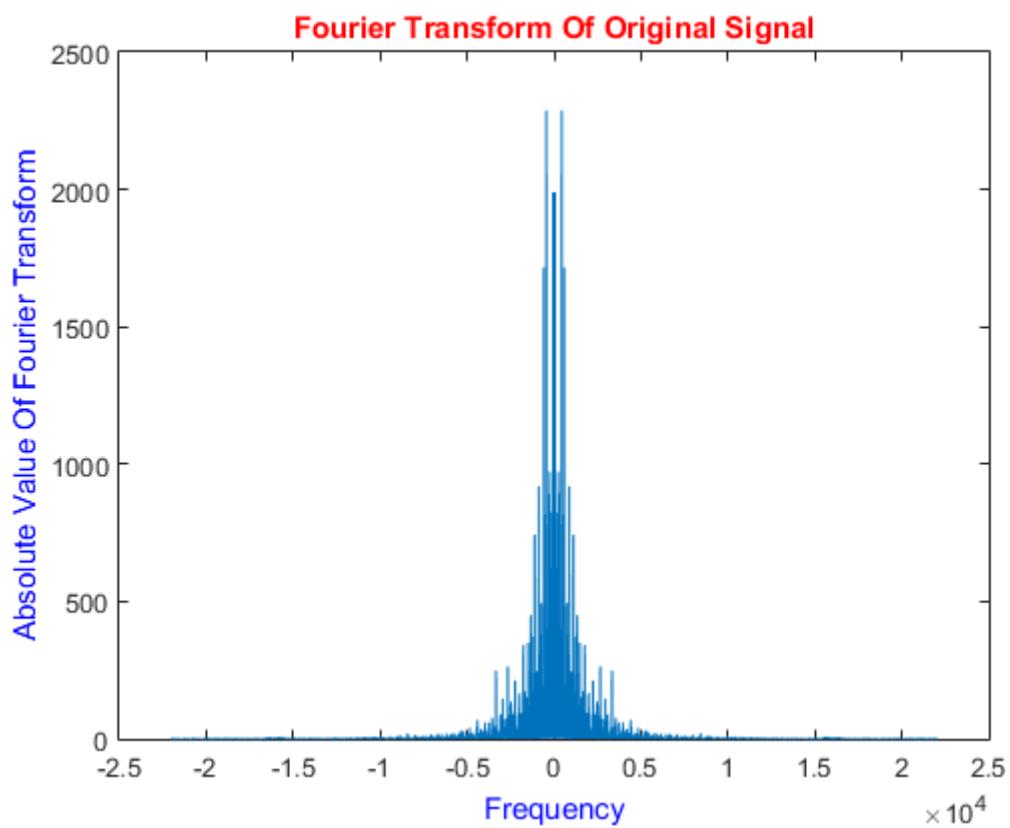


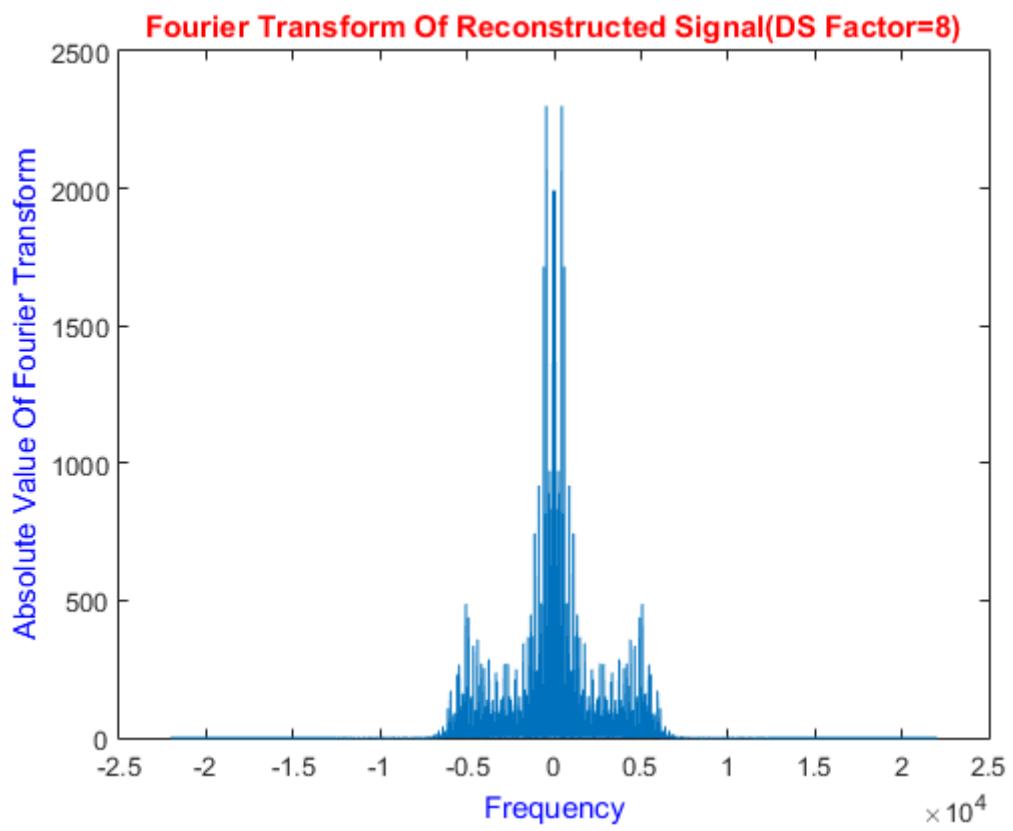
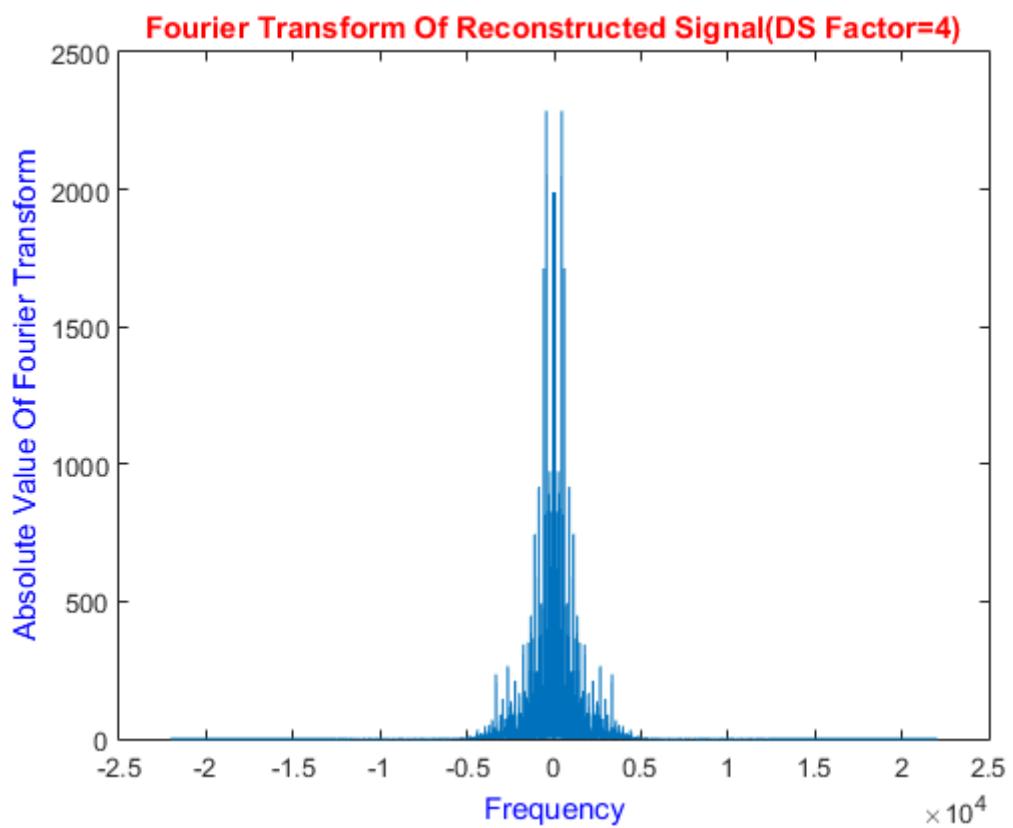
همان‌طور که می‌دانیم در اثر *upsampling* دامنه مقادیر تغییری نمی‌کند و تنها تبدیل فوریه به مقدار مقیاس *upsampling* جمع می‌شود. در تصاویر فوق نیز این مطلب مشهود است چرا که تفاوت دامنه‌ها به این علت است که از سیگنال‌های با دامنه متفاوتی گرفته شده‌اند. هم‌چنین اثر جمع شدن سیگنال را با توجه به محدوده فرکانسی می‌توان مشاهده کرد.

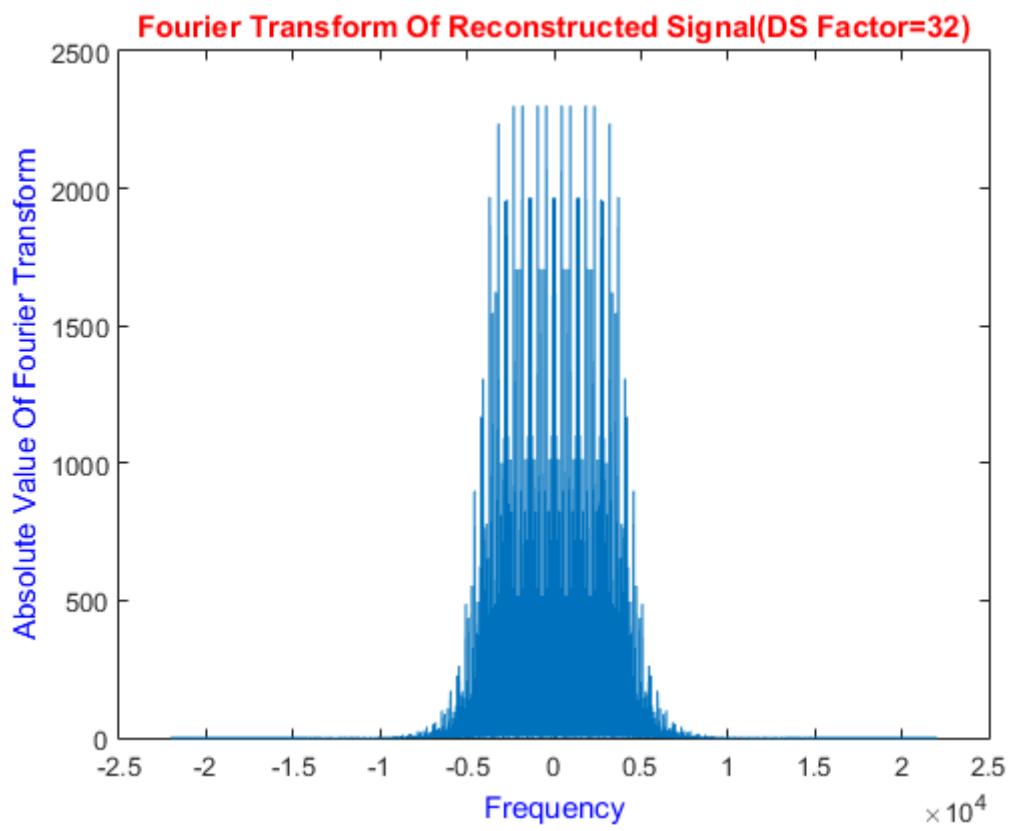
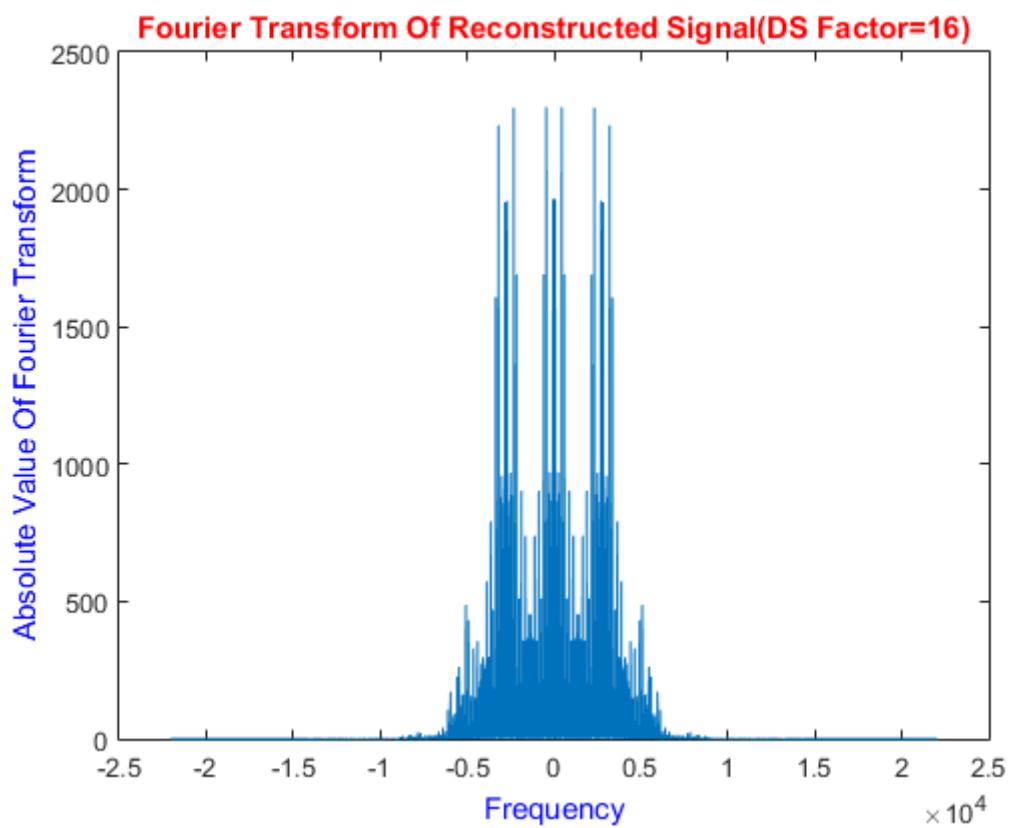
در اینجا باید دقت کرد که در اثر *upsampling* و جمع شدن تبدیل فوریه، فرکانس‌های بعد از π می‌توانند وارد ناحیه اصلی و بنابراین وارد تبدیل فوریه شده و صدا را از سیگنال اصلی متفاوت کنند. برای حذف این اثرات باید از فیلتر پایین‌گذر استفاده نمود. در این مثال هم به ازای افزایش نرخ نمونه برداری با مقیاس ۲، همانطور که در تبدیل فوریه در صفحه‌های قبل مشاهده می‌کنیم، فرکانس‌های بالایی به سیگنال اضافه می‌شوند که با توجه به زیاد بودنشان(حدود ۲۰ کیلوهرتز) گوش انسان قادر به شنیدن آن‌ها نبوده و همینطور کامپیوتر و لپتاپ توانایی تولید فرکانس‌هایی به این بالایی را ندارند. اگر سیگنالی داشتیم که در اثر چنین وضعیتی فرکانس‌های بالایی جدید آن در محدوده شنوازی باشند، آن‌گاه به اثر *upsampling* و تفاوت سیگنال خروجی آن با خروجی گذشته از فیلتر پایین‌گذر به راحتی پی می‌بردیم. در بقیه افزایش نرخ نمونه برداری‌ها فرکانس‌های بالای بیشتری وارد ناحیه $[\pi, \pi]$ شده و سیگنال خروجی را به شدت خراب می‌کنند.

۵. توضیح کد: ابتدا تخمین مان از فرکانس بیشینه را به عنوان ماکریم فرکانس در نظر می‌گیریم.(فرکانس قطع) سپس آن را بر نصف فرکانس نمونه برداری تقسیم کرده تا به فرکانس قطع نرمال شده قابل استفاده برای

دستور *butter* برسیم. همچنین مرتبه فیلتر را 6 انتخاب می‌کنیم. ضرایب فیلتر با ترویرت مورد نظر را می‌گیریم، همه سیگنال‌های *upsample* شده را در بهره مناسب خود(به خاطر دفع اثر *downsample*) ضرب کرده و سپس از فیلتر پایین‌گذر عبور می‌دهیم. در ادامه آن تبدیل‌های فوریه را برای سیگنال‌های نهایی رسم می‌کنیم. همچنین برای پخش صدا کنترل آن به کاربر داده شده است. نتایج نهایی تبدیل فوریه سیگنال بازسازی شده در زیر آورده شده است.

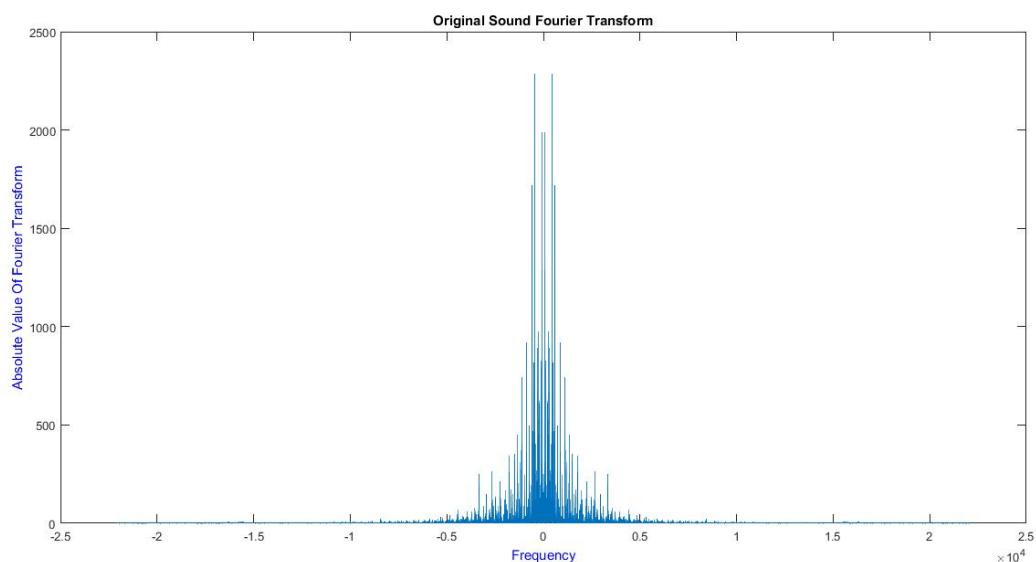


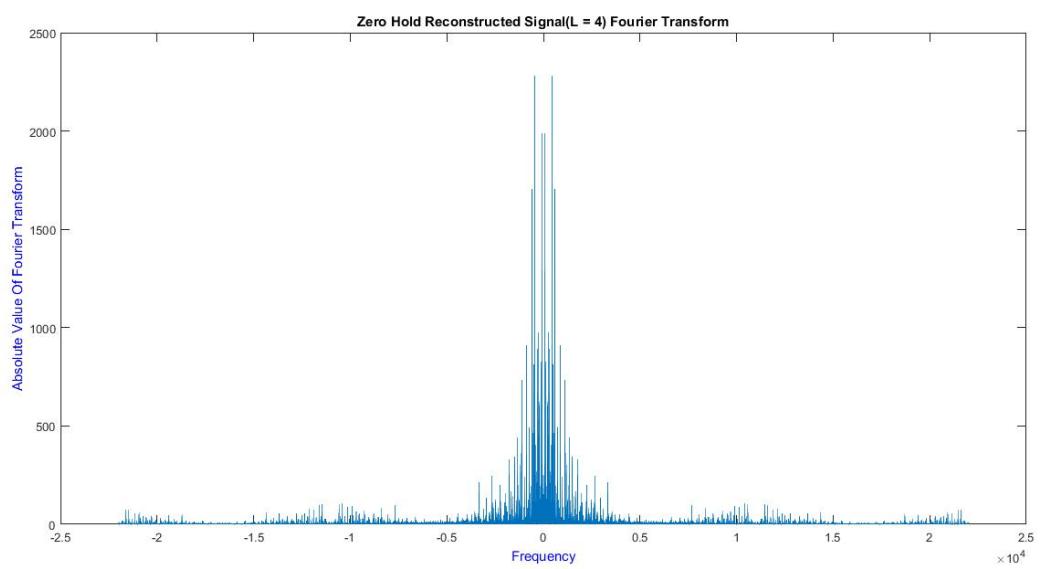
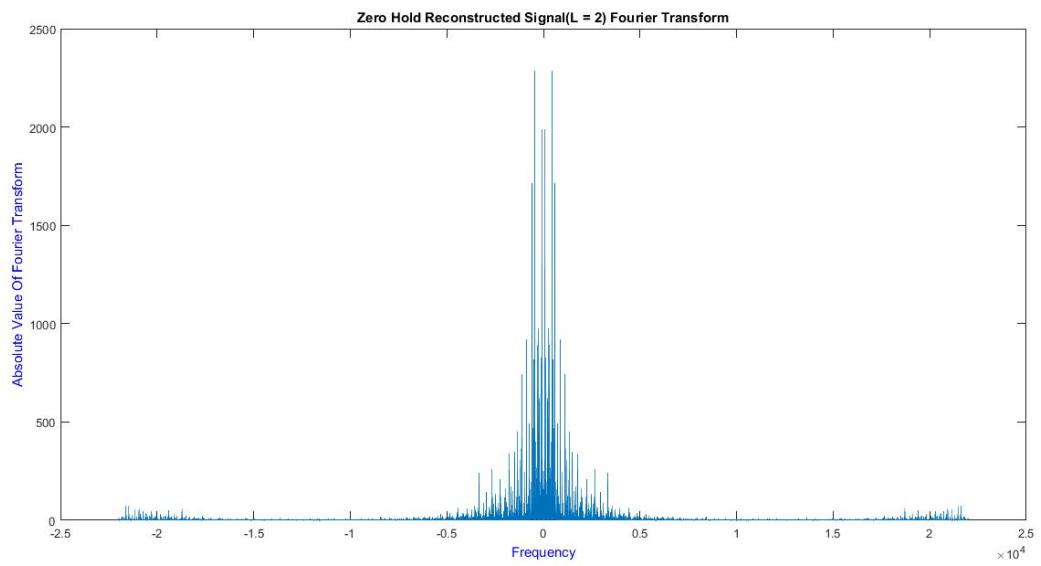


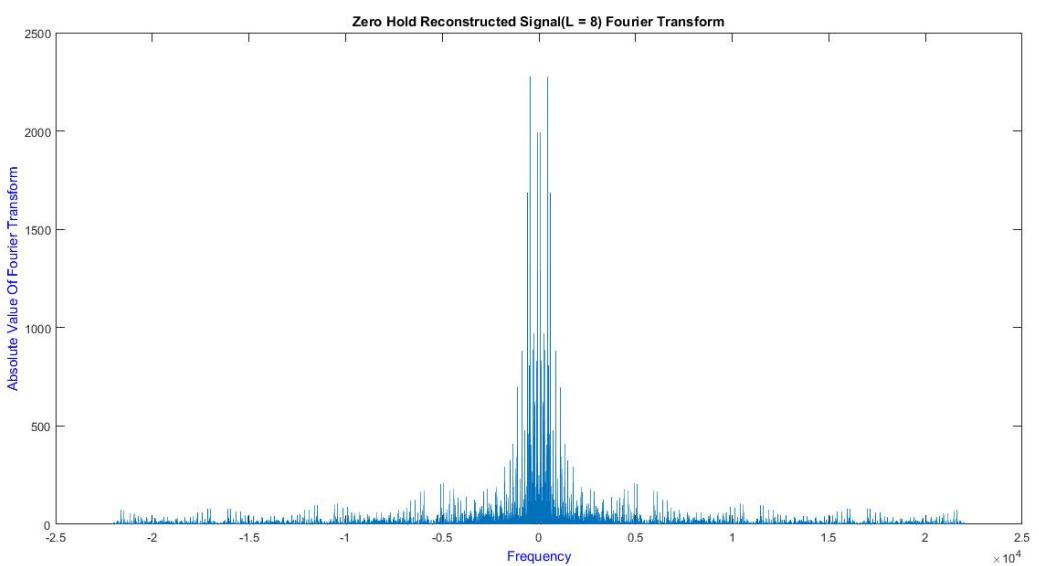
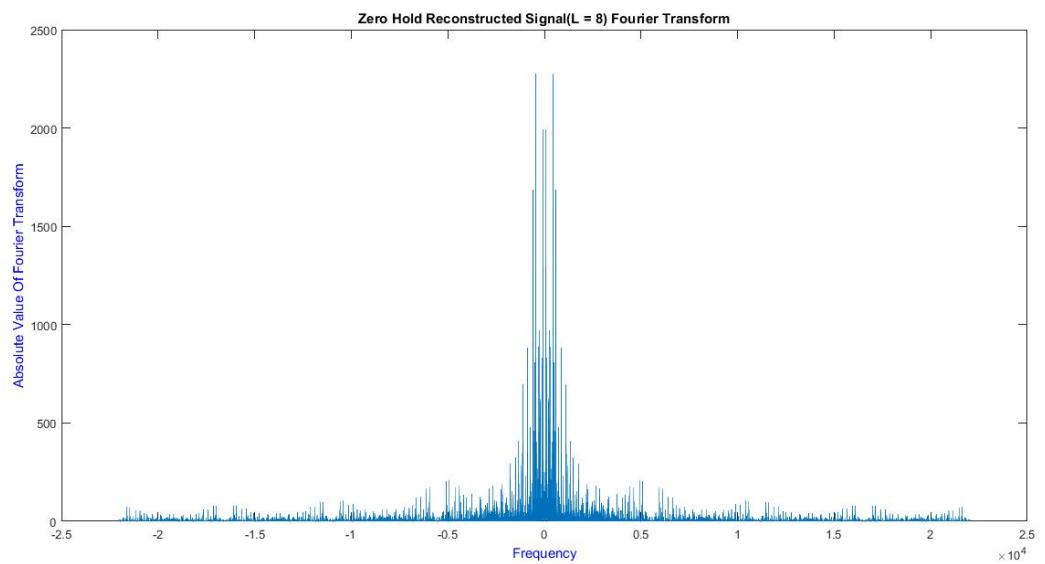


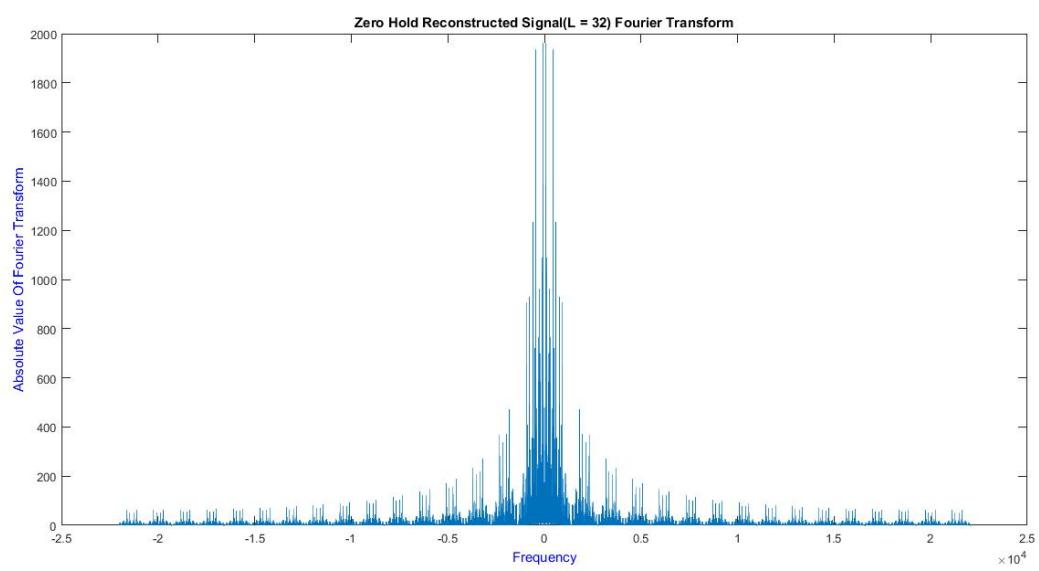
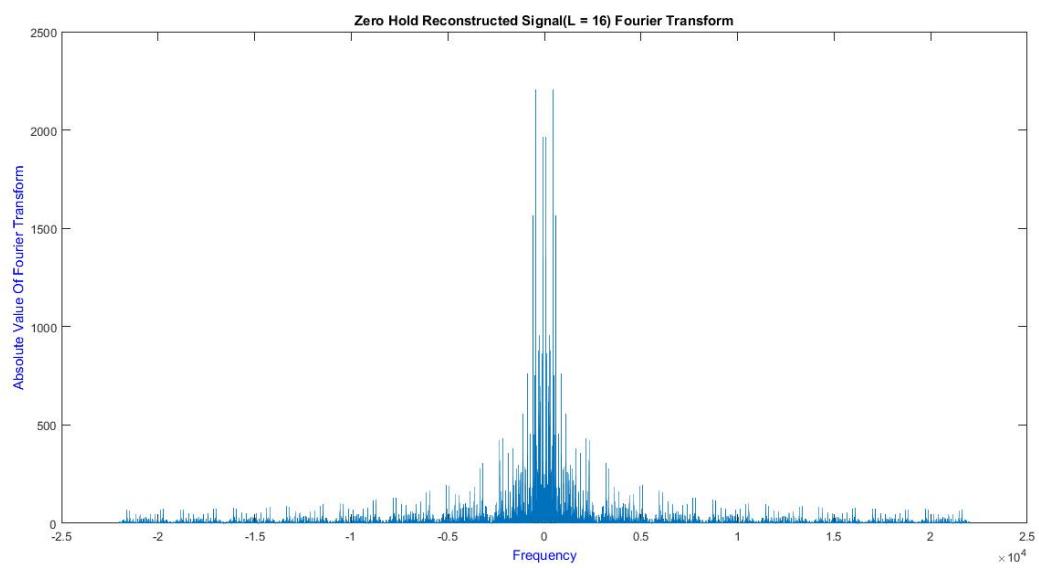
همان طور که در تصاویر بالا مشاهده می شود به ازای کاهش نرخ نمونه برداری با مقیاس ۲ و ۴، تبدیل فوریه سیگنال بازسازی شده تفاوت چندانی با تبدیل فوریه سیگنال اصلی ندارد. اما با توجه به اینکه نرخ نایکوییست با توجه به تخمین ما از بیشترین فرکانس موجود در سیگنال، حدود ۸ کیلوهرتز است، اگر نرخ نمونه برداری ۸ برابر یا بیشتر کم شود، دیگر قضیه نمونه برداری برقرار نیست و مسلماً سیگنال بازسازی شده با سیگنال اصلی برابر نخواهد بود. با پخش سیگنال‌ها نیز دقیقاً می‌توان به موضوع گفته شده پی برد. در واقع از ۸ برابر به بعد اختلاط فرکانسی شدید و شدیدتر شده تا جایی که دیگر شباهتی به سیگنال صوتی اصلی نخواهد داشت.

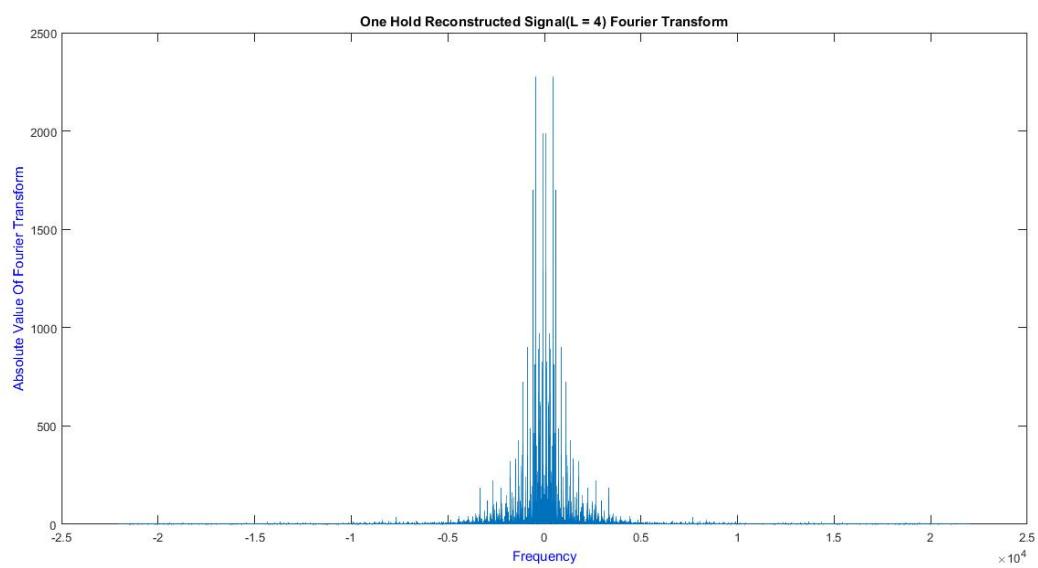
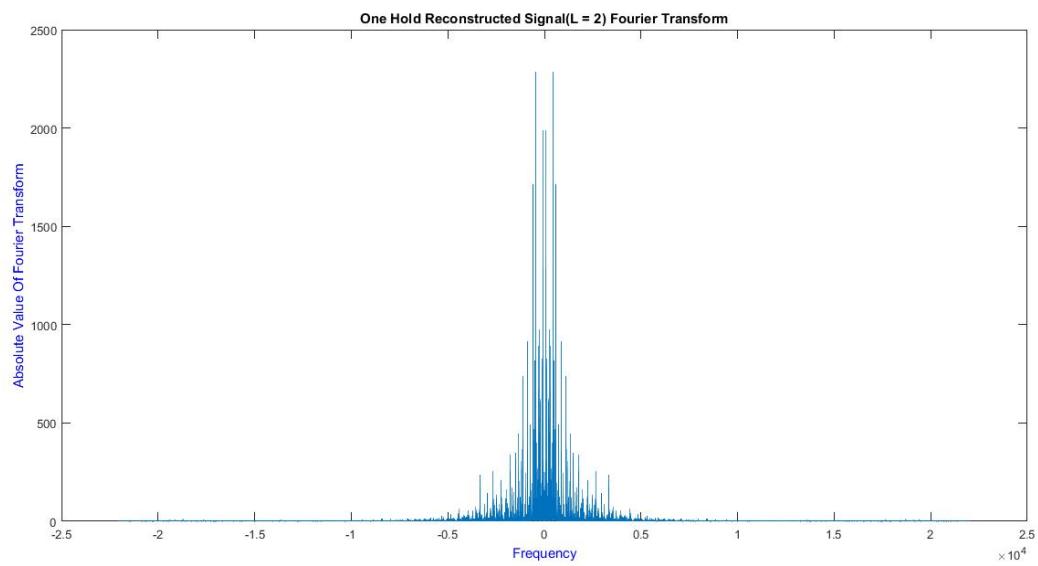
۶. ابتدا توابع oneHold و zeroHold را برای بازسازی با تقریب مرتبه‌ی صفر و مرتبه‌ی یک طراحی می‌کنیم؛ این توابع در ورودی خود، سیگنال مربوطه و عدد n را می‌گیرند و تابع اول بین هر دو نمونه‌ی سیگنال، $1 - n$ نسخه از نمونه‌ی سمت چپ را قرار می‌دهد و تابع دوم، بین هر دو نمونه‌ی سیگنال، $1 - n$ نمونه را که دامنه‌ی آن‌ها به صورت خطی از نمونه‌ی سمت چپ تا نمونه‌ی سمت راست افزایش می‌یابد، قرار می‌دهد. اگر این دو تابع را با جایگذاری مقادیر L سوال ۳، بجای n روی سیگنال‌های downsample شده اعمال کرده و سپس تبدیل فوریه بگیریم، خواهیم داشت :

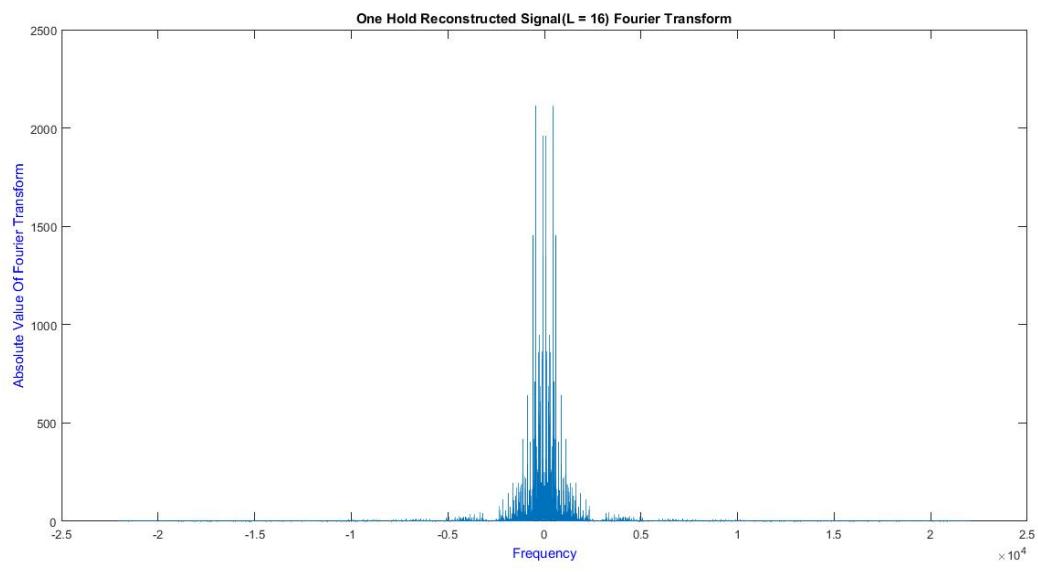
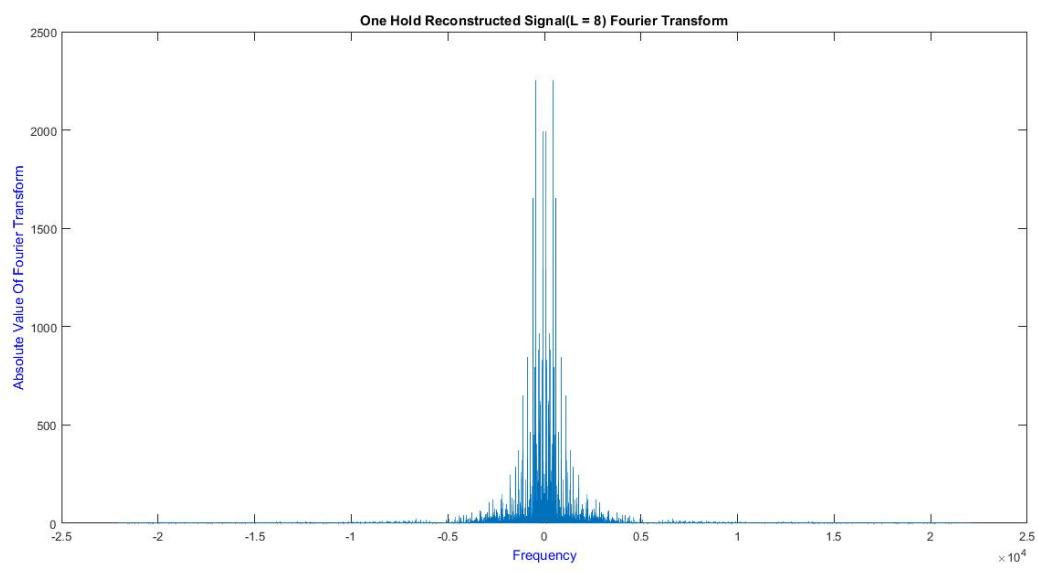


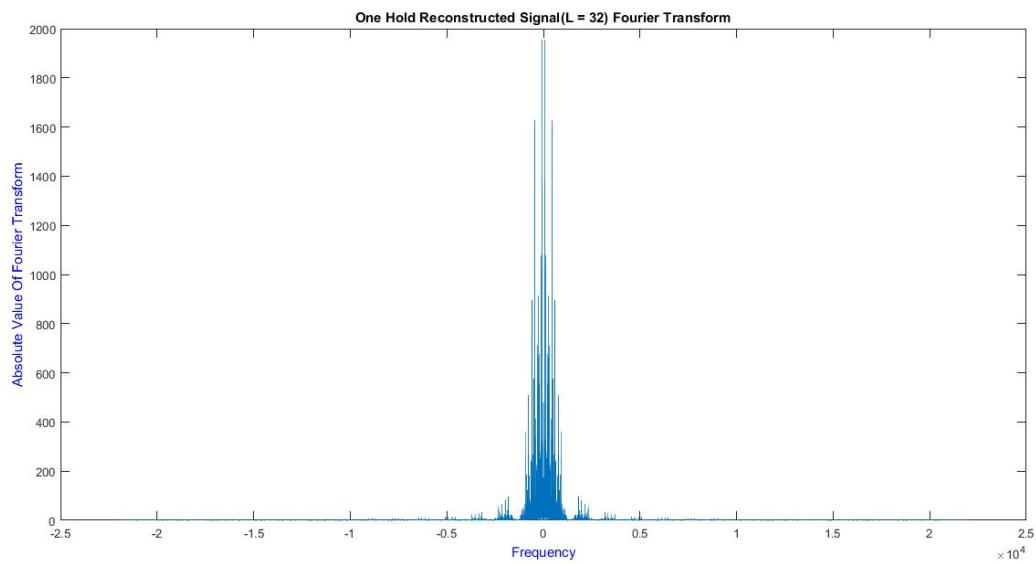






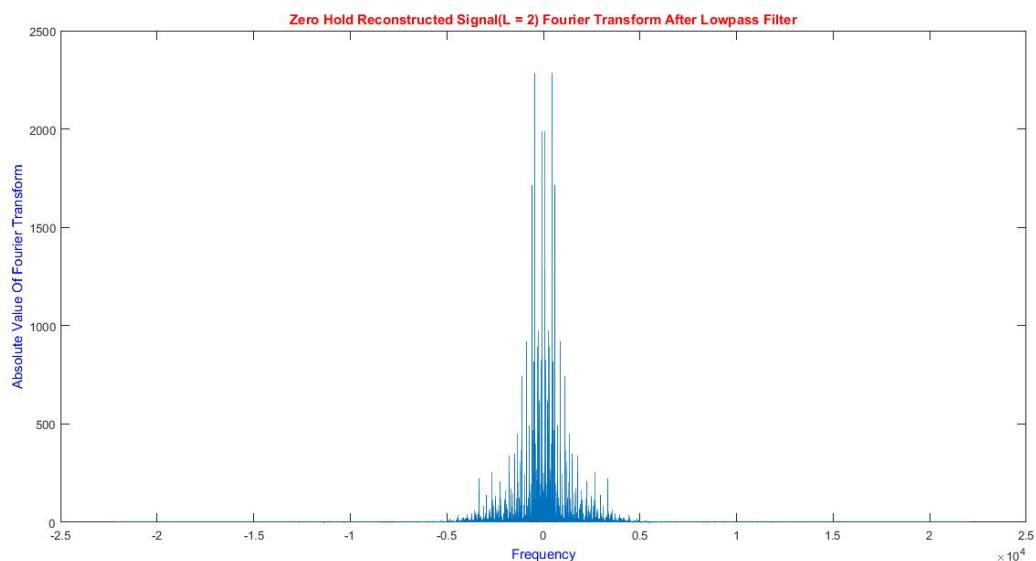


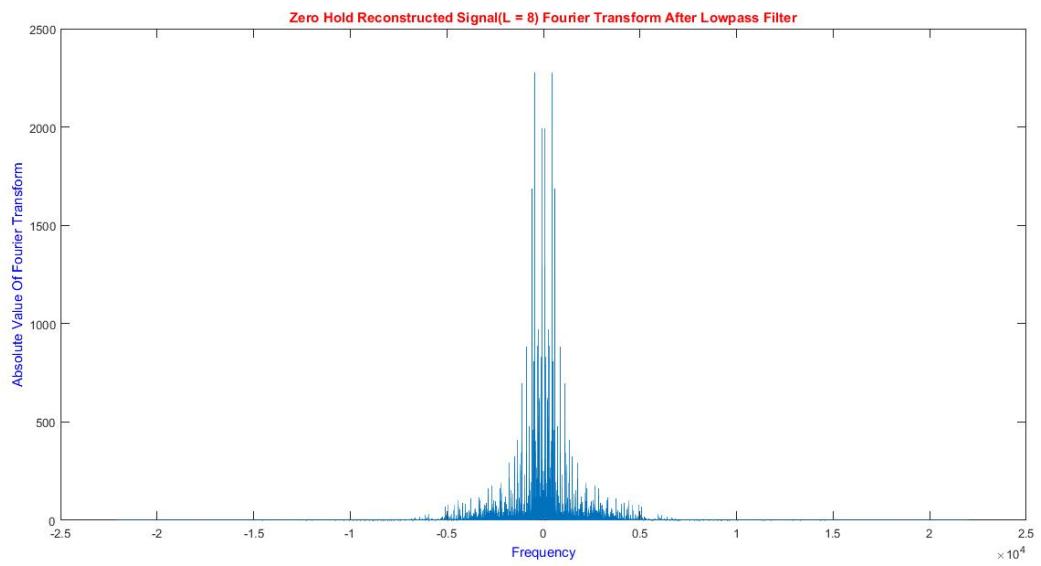
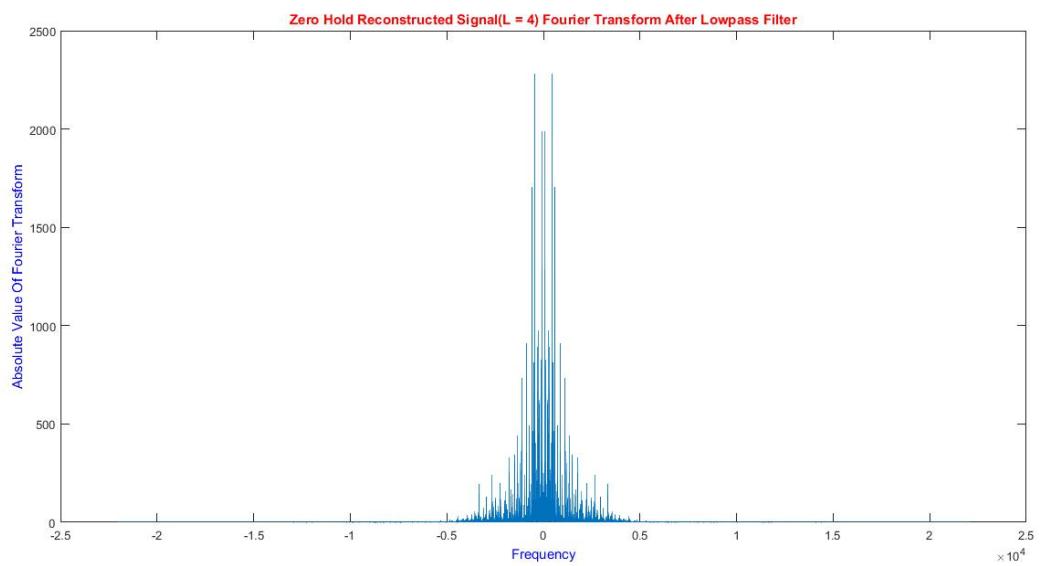


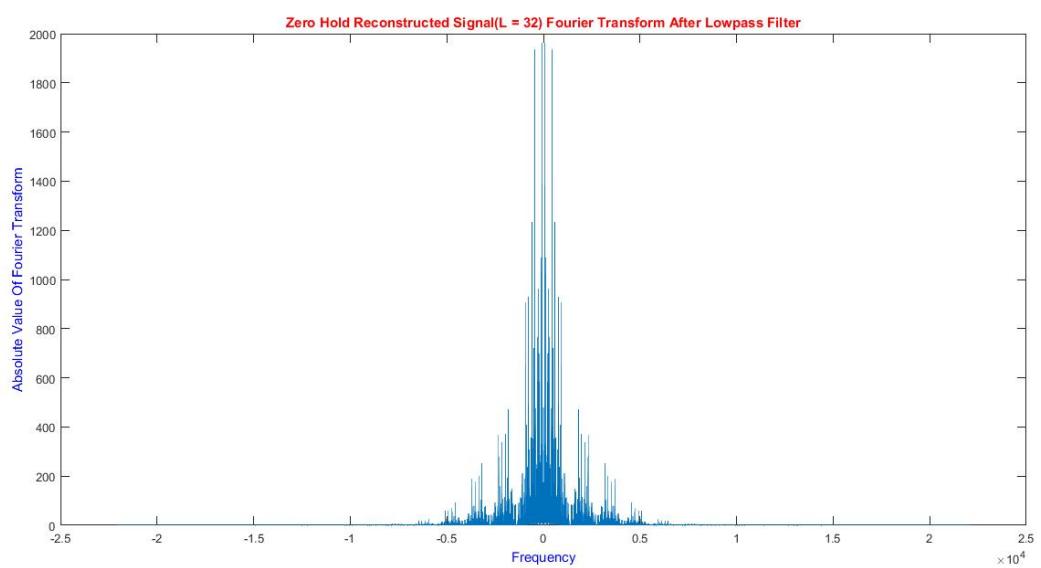
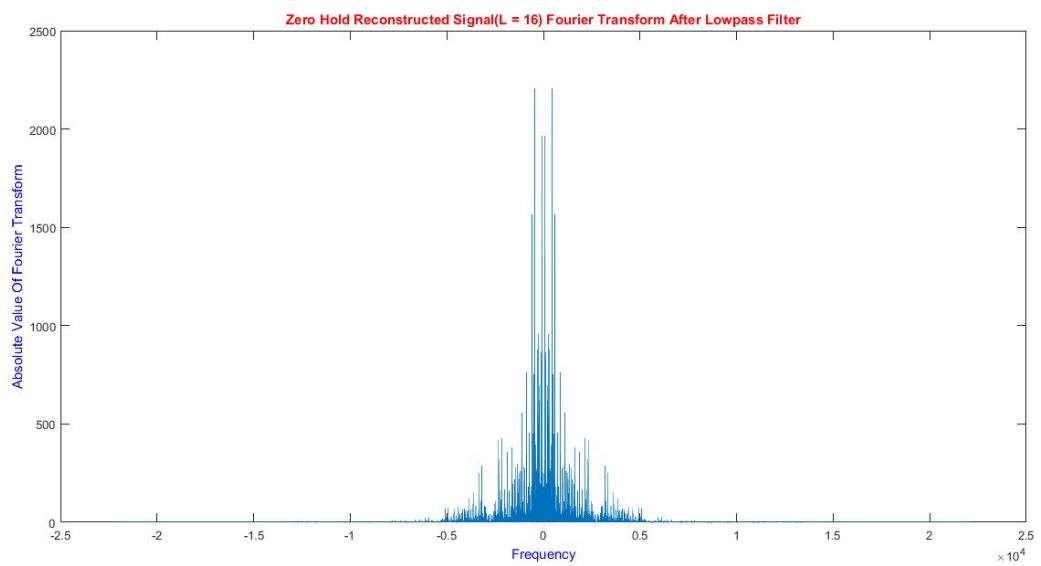


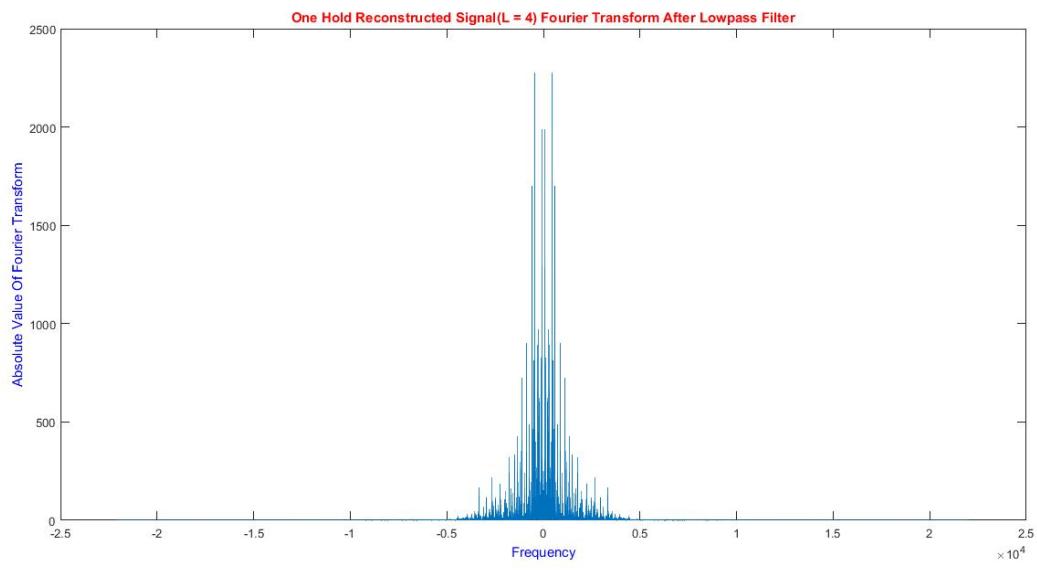
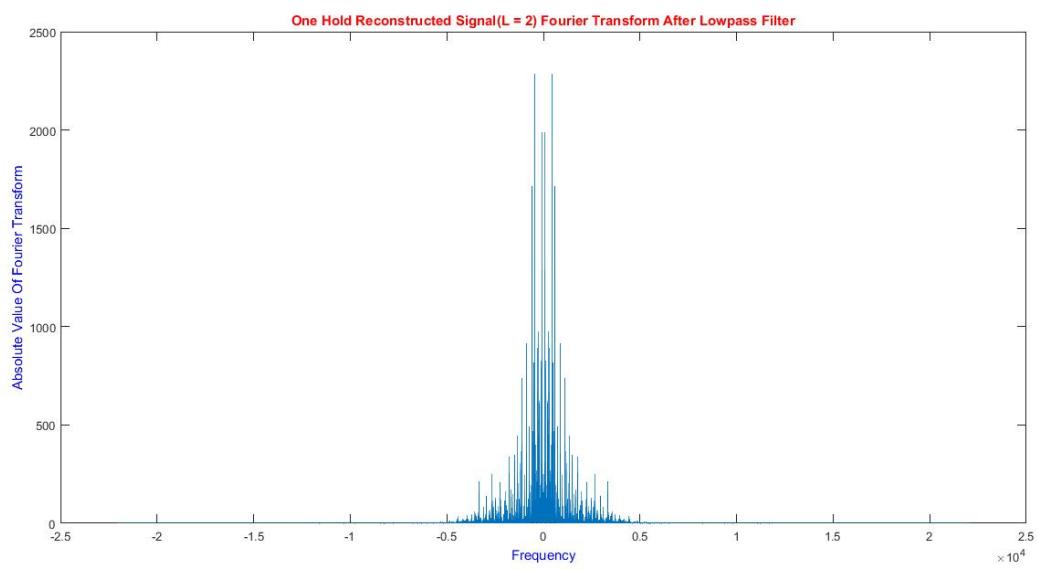
همان‌طور که مشاهده می‌شود، مانند قسمت‌های قبل، در این قسمت نیز پس از نمونه برداری، تبدیل فوریه متناوب می‌شود، با این تفاوت که دامنه‌ی تناوب‌ها با دور شدن از مبدأ، کم می‌شود.

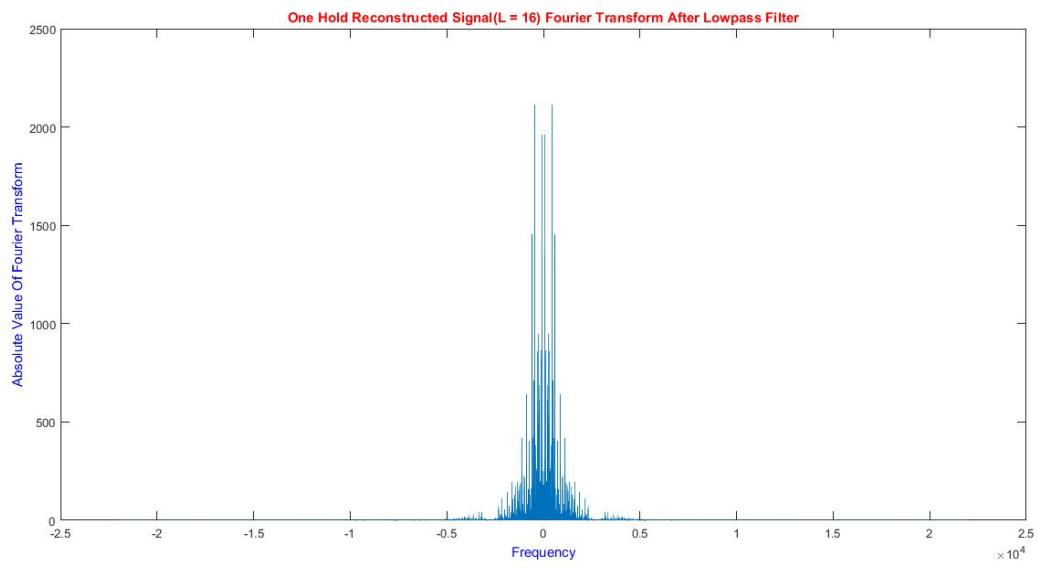
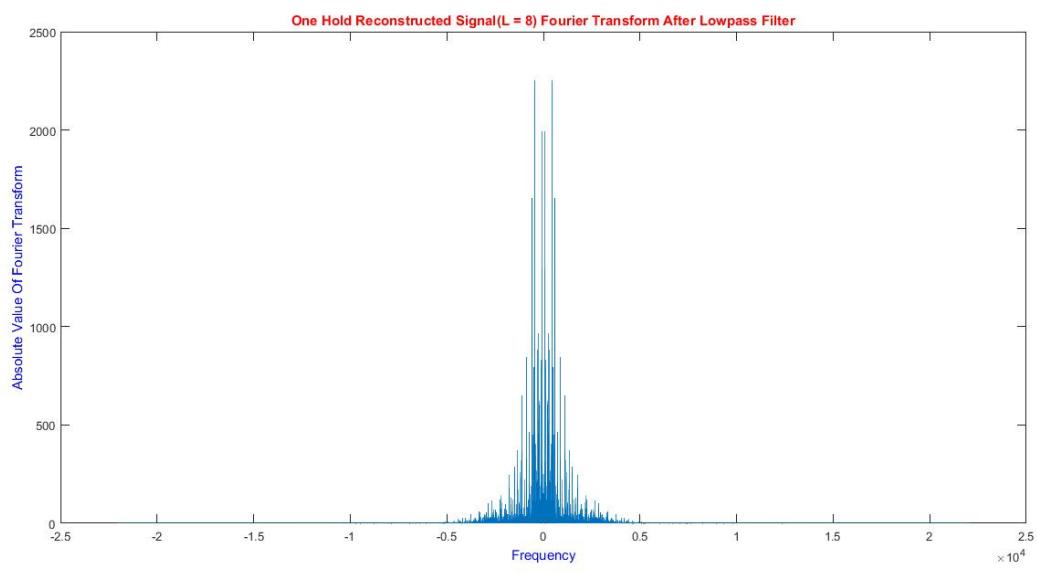
۷. در این قسمت نیز برای طراحی ضرایب چندجمله‌ای‌های صورت و مخرج کسر فیلتر پایین‌گذر از تابع استفاده می‌کنیم؛ فرکانس قطع را برابر 4kHz در نظر می‌گیریم که طبیعتاً باید نرمالایز (تقسیم بر نصف فرکانس نمونه برداری) شود، مرتبه‌ی فیلتر را نیز 4 در نظر می‌گیریم. سپس با استفاده از تابع filter، سیگنال‌های بازسازی شده‌ی قبل را فیلتر می‌کنیم. پس از انجام عملیات مذکور تبدیل فوریه‌ی سیگنال‌ها به صورت زیر خواهد شد :

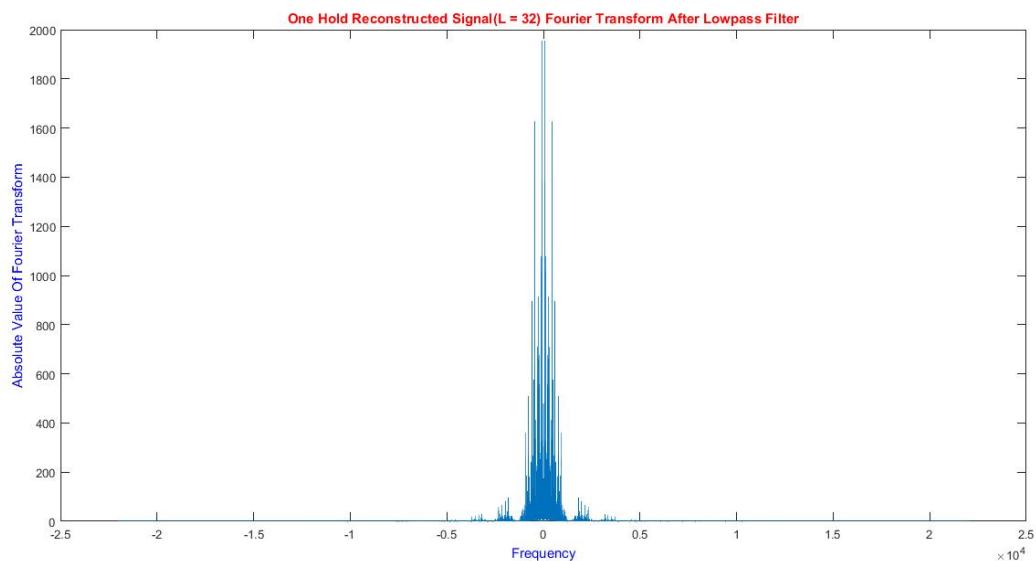








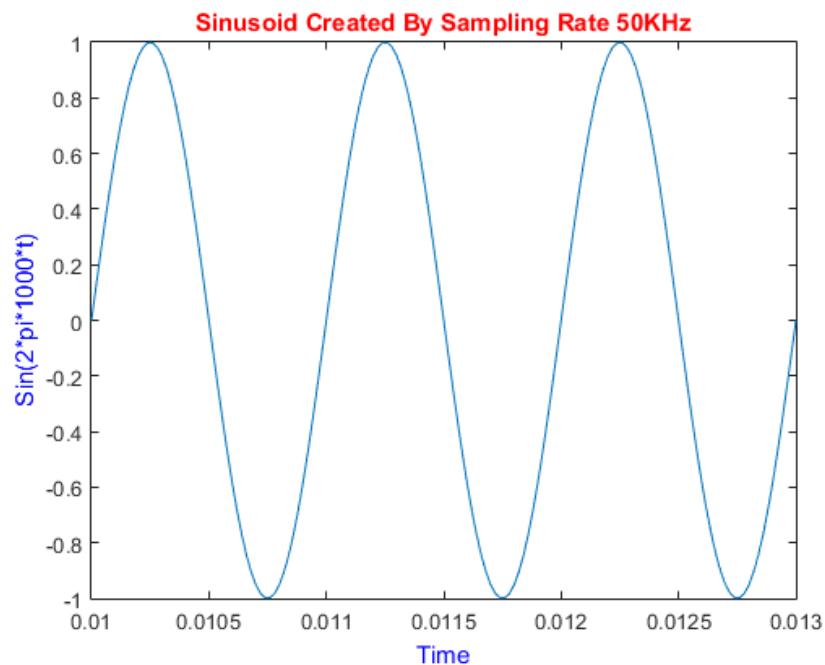




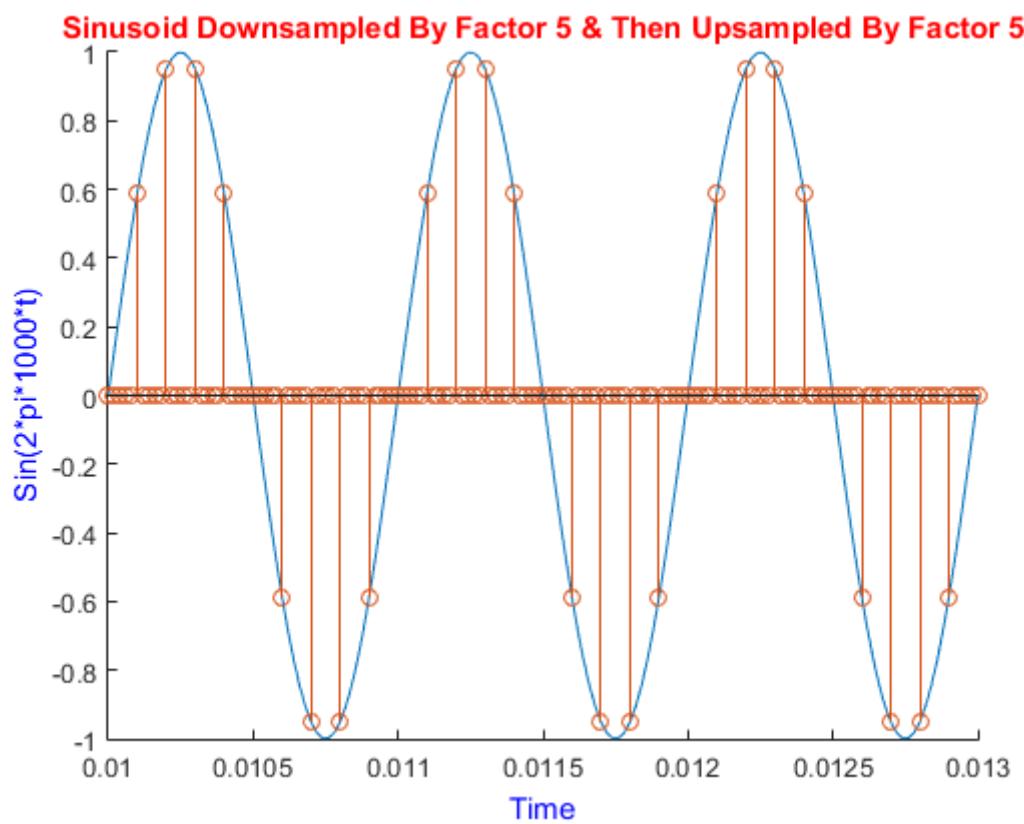
با مقایسه ای نتایج این قسمت با قسمت های قبل، اولا مشاهده می کنیم که افت شدید دامنه ای که در بازسازی با استفاده از upsample وجود داشت در onehold و zerohold وجود ندارد، ثانیاً وضوحی که در صوت وجود دارد، حتی پس از بازسازی کردن سیگنال downsample شده با مقیاس های بالا، پس از بازسازی با کمک onehold و پس از آن zerohold به مراتب بیشتر از بازسازی با استفاده از upsample است (برای پخش کردن صوت، کنترل player ایجاد شده است و کافی است در پنجره فرمان، (player) تایپ شود). نتیجه هی دوم از لحاظ شهودی هم تا حدودی واضح است، زیرا در بازسازی صوت با استفاده از upsample، ما صرفا بدون هیچ اطلاعاتی از سیگنال، صرفا بین هر دو نمونه تعدادی صفر قرار می دهیم و این کار حین بازسازی سیگنال downsample شده با مقیاس های بالا، شدیداً وضوح صوت را خراب می کند، اما در بازسازی به کمک onehold و یا zerohold، برای بازسازی، از اطلاعاتی که خود سیگنال به ما می دهد هم استفاده می کنیم لذا بازسازی بهتری خواهیم داشت.

قسمت چهارم: درون یابی

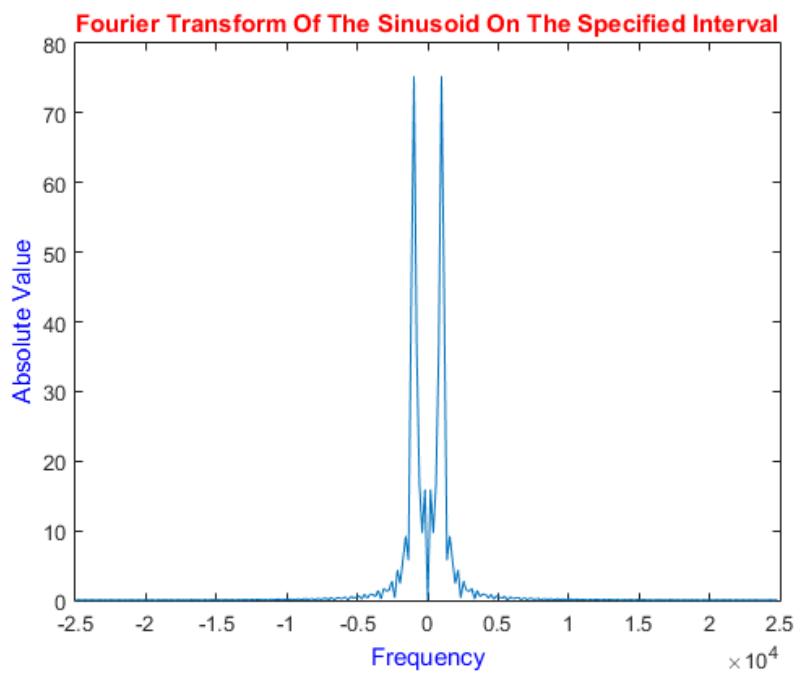
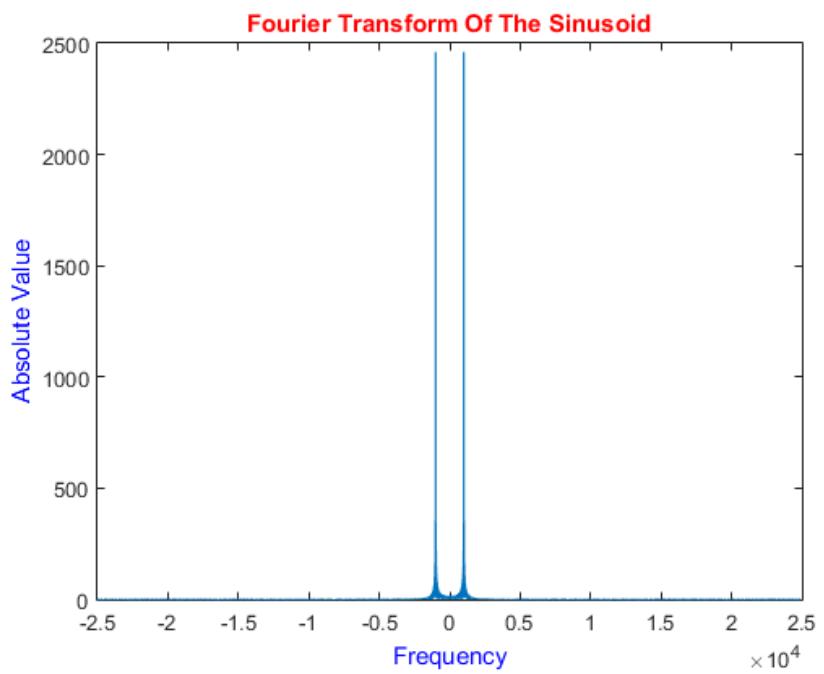
۱. سیگنالی با فرکانس ۱ کیلوهرتز را با فرکانس نمونه برداری ۵۰ کیلوهرتز می‌سازیم. نتیجه رسم در بازه مذکور در شکل زیر مشخص است.

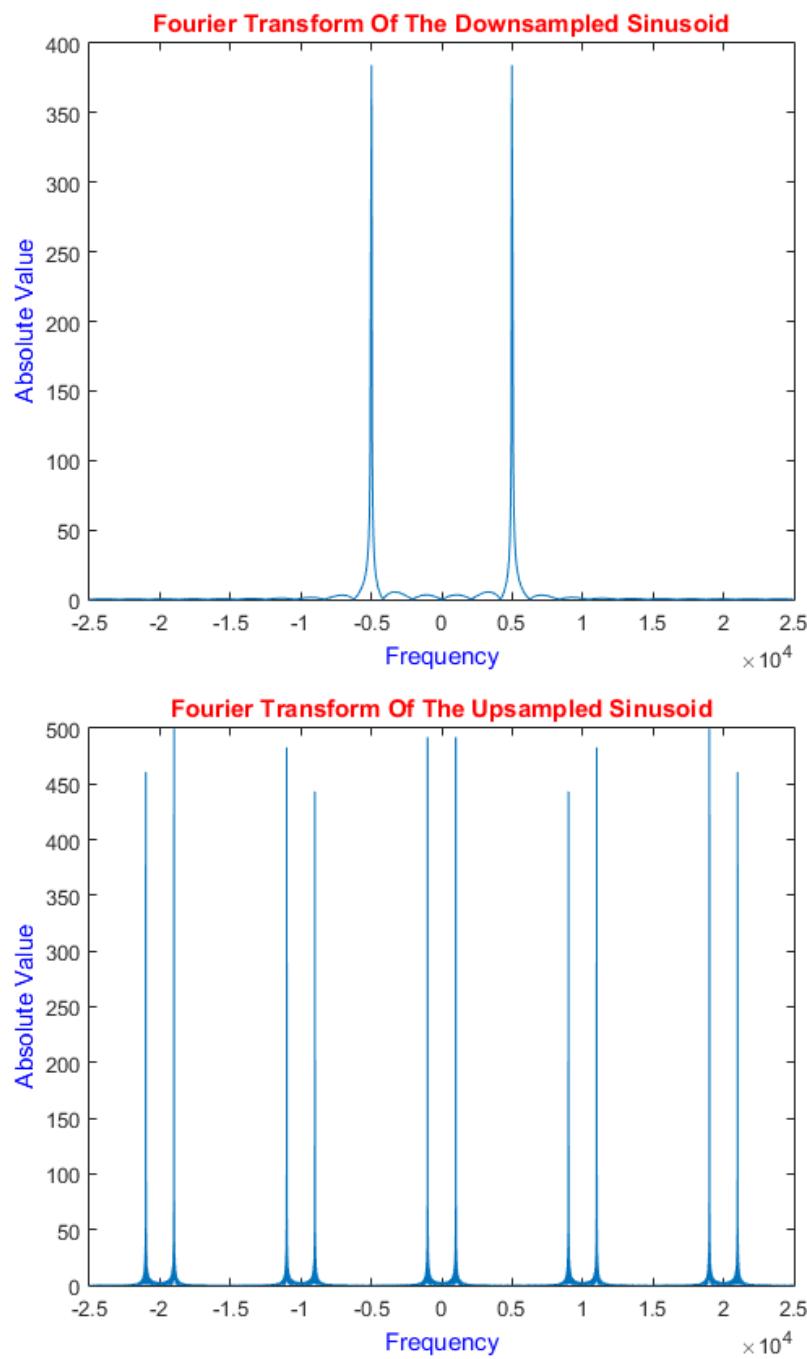


۲. برای نمونه برداری با فرکانس ۱۰ کیلوهرتز باید با مقیاس $\frac{50KHz}{10KHz} = 5$ downsample کنیم. سپس همان‌طور که گفته شده به ازای هر نمونه ۴ صفر قرار می‌دهیم. نتیجه در تصویر زیر مشاهده می‌شود.(برای عملیات‌های بعدی متغیر جدید y_0 را تعریف کرده‌ایم که فقط شامل بازه مورد نظر است).



۳. نتایج در تصاویر قابل رویت‌اند.

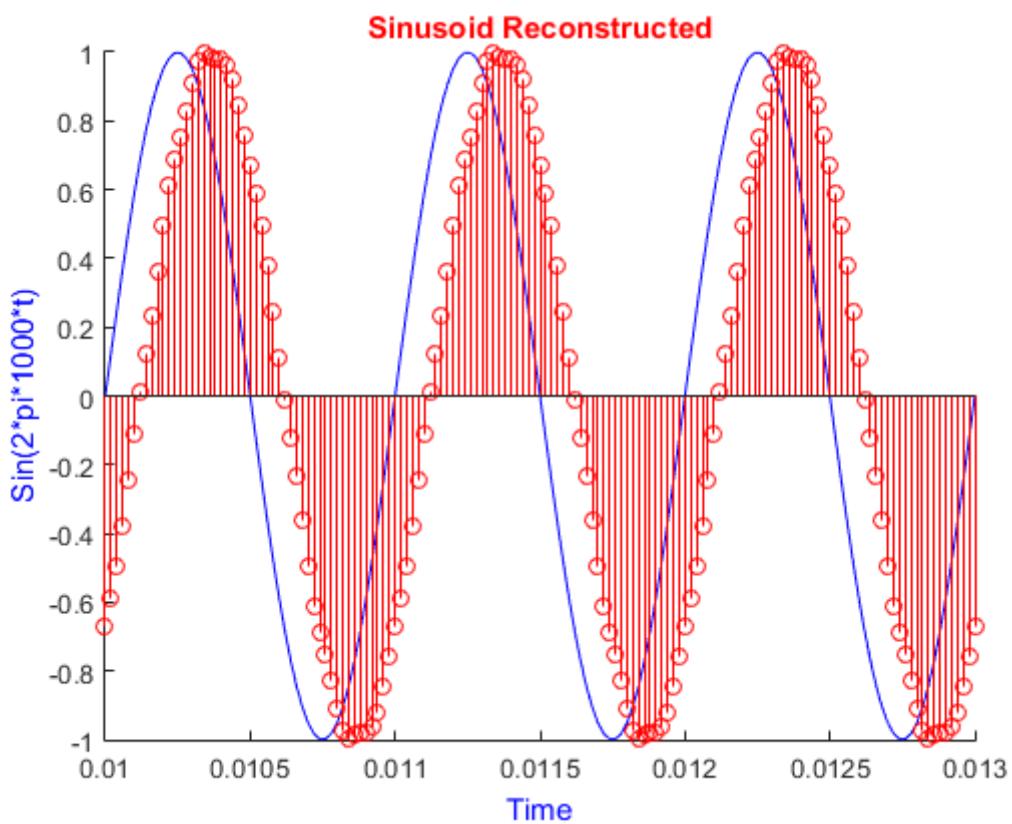




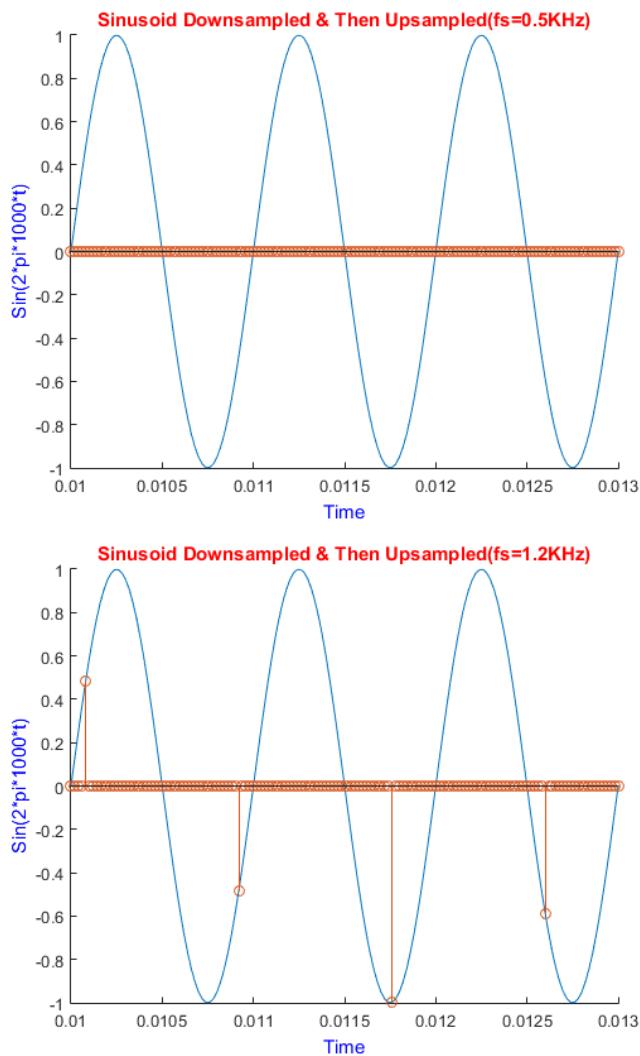
توضیح : همانطور که مشاهده می شود در تبدیل فوریه سینوسی، به جای ضربه، شبیب بسیار تند داریم که این با توجه به این واقعیت که سینوسی تولید شده از $-\infty$ تا ∞ نیست، قابل انتظار است و در واقع هرچه طول سینوسی بیشتر باشد، تبدیل فوریه آن نیز به ضربه می گراید و بر عکس هرچه طول سینوسی کمتر باشد از ضربه ایدهآل دورتر می شویم. این واقعیت را در تبدیل فوریه بخشی از سینوسی که در بازه مشخصی تعریف شده است می توان ملاحظه نمود. شکل سوم مربوط به *downsampling* است که در این حالت می دانیم مقادیر تبدیل فوریه بر مقیاس *downsampling* نقسیم و فرکانسها در آن ضرب می شوند. این نتیجه را نیز در شکل سوم می توان مشاهده کرد. نهایتا در شکل چهارم تبدیل فوریه سیگنال *upsample* شده از

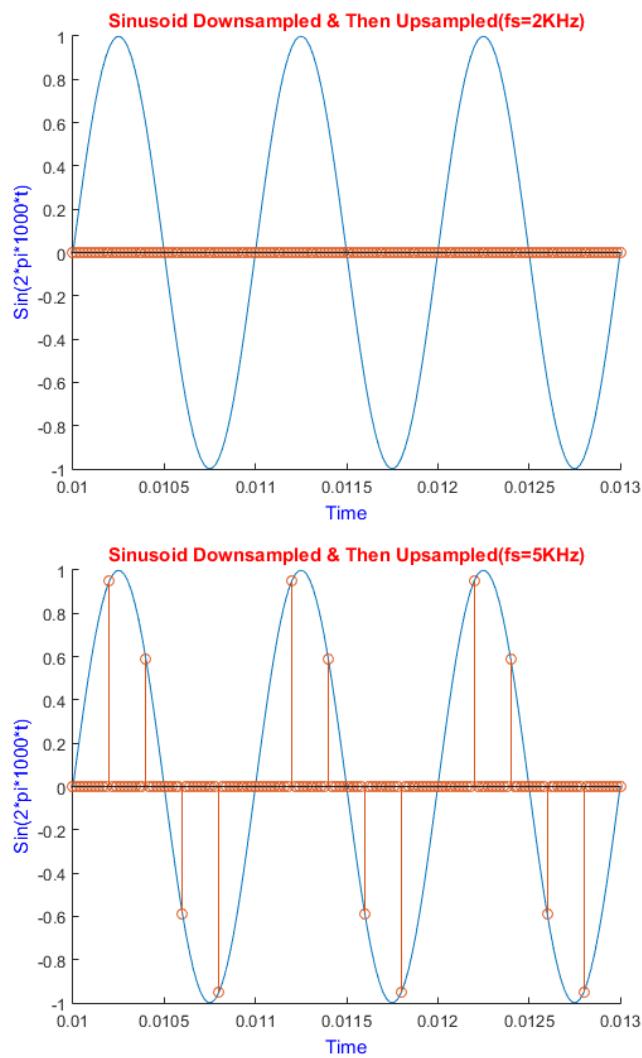
سیگنال قبلی را ملاحظه می‌کنیم که با توجه به قسمت‌های قبل می‌دانیم در اثر این عملیات، تبدیل فوریه سیگنال جدید، منقبض شده تبدیل فوریه سیگنال قبلی با ضریب مقیاس *upsample* است. این واقعیت نیز در تصویر چهارم به راحتی قابل تشخیص است.

۴. برای طراحی فیلتر پایین‌گذر از فیلتر باترورث استفاده می‌کنیم. (دستور butter) مرتبه آن با آزمون و خطا بدست آمد و مقدار ۶ مناسب به نظر می‌رسد. همان‌طور که می‌دانیم، اگر فرکانس قطع فیلتر پایین‌گذر را نصف فرکانس نمونه برداری ($10KHz$) در نظر بگیریم، تا زمانی که اختلاط فرکانسی پیش نیاید، فیلتر ما مناسب خواهد بود. بنابراین همین مقدار را به عنوان فرکانس قطع فیلتر پایین‌گذر ایده‌آل خود در نظر می‌گیریم و با تقسیم بر نصف فرکانس نمونه برداری سیگنال اصلی ($50KHz$) آن را نرمال می‌کنیم. سپس با داستن *upsample* و فرکانس قطع نرمالایز شده، ضرایب فیلتر باترورث مورد نظر را می‌گیریم. سپس سیگنال *downsample* شده را از همین فیلتر و با بهره ۵ عبور می‌دهیم، علت ضرب شدن در ۵ این است که پس از *upsample* شدن بهره به مقدار ۵ افت می‌کند. برای جبران آن باید در ۵ ضرب شود. نتیجه نهایی در شکل زیر مشاهده می‌شود. علت تفاوت مکانی به اندازه یک انتقال به این خاطر است که فیلتر باترورث فاز خطی به تبدیل فوریه اضافه می‌کند که می‌دانیم این فاز خطی باعث یک انتقال سیگنال در حوزه زمان می‌شود. همچنان ملاحظه می‌کنیم به علت اینکه از نرخ نایکوییست نگذشته‌ایم، سیگنال به درستی بازسازی شده است.

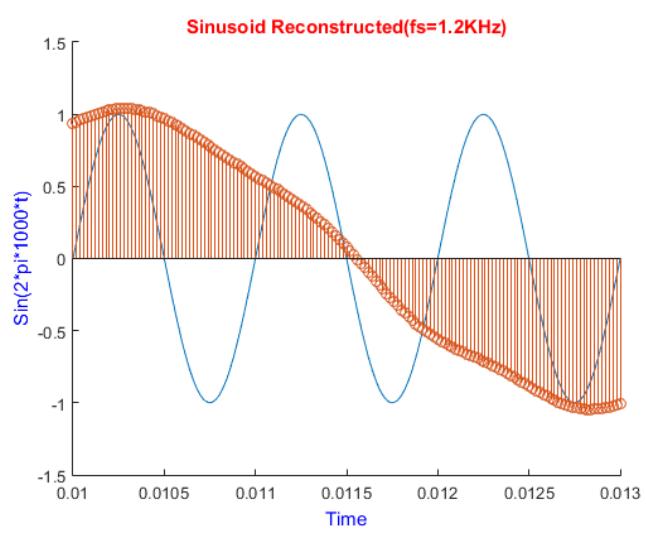
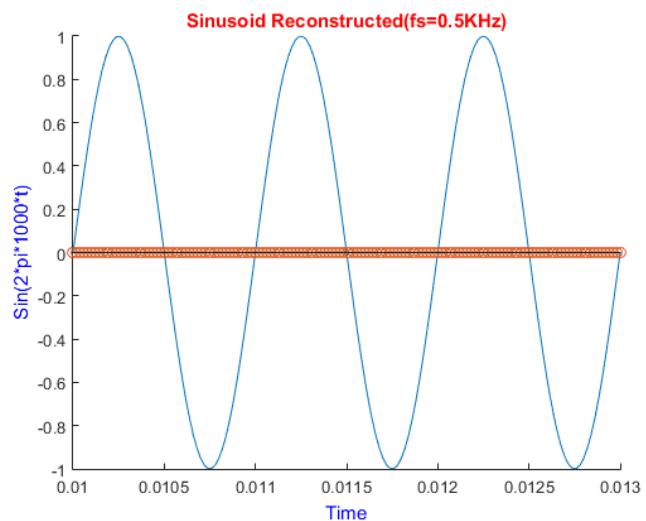


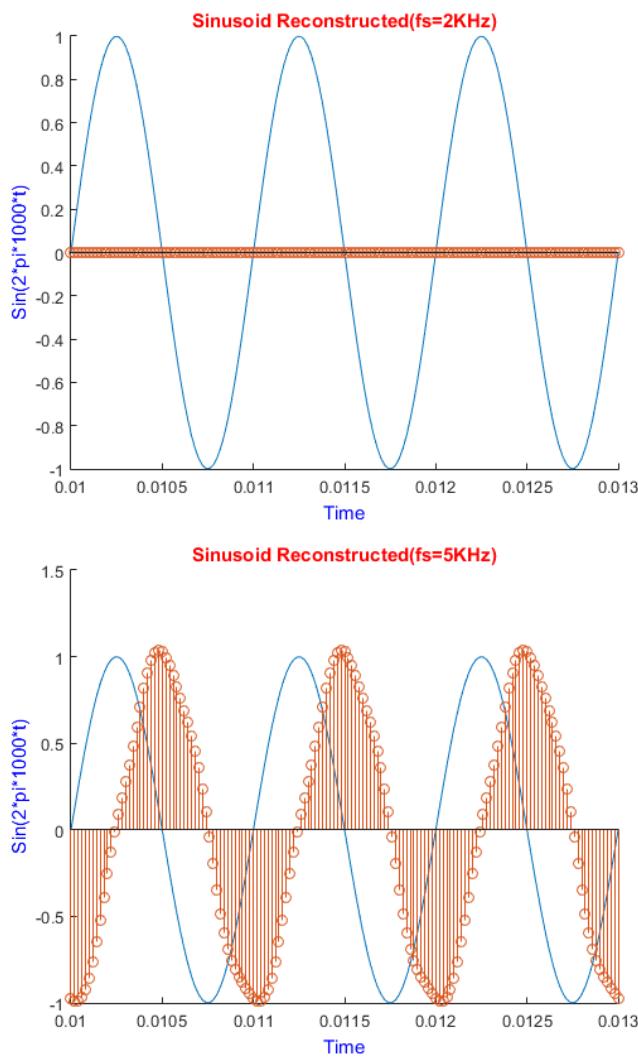
۵. توضیح کد: ابتدا برداری برای فرکانس‌های نمونه برداری ساخته می‌شود. پس از آن مقیاس *upsample* با گرفتن سقف(*ceil*) حاصل تقسیم فرکانس نمونه برداری کل سیگنال بر فرکانس نمونه برداری خاص هر مورد بدست می‌آید. مقیاس *downsample* نیز با مقدار اخیر برابر است. در ادامه برای هر یک از فرکانس‌های نمونه برداری گفته شده عملیات *upsampling* و *downsampling* صورت می‌گیرد. پس از آن برای هر کدام از فرکانس‌ها عملیات سوال ۲ یعنی ابتدا کاهش نرخ نمونه برداری و سپس افزایش آن صورت می‌گیرد و نتیجه همراه سیگنال مربوط به آن فرکانس بر روی یک شکل رسم می‌شوند. این نتایج در تصاویر زیر قابل مشاهده‌اند.





در ادامه مانند قسمت ۴ فرکانس‌های قطع نرمالایز شده را بدست می‌آوریم و با انتخاب مرتبه فیلتر برابر ۶، ضرایب صورت و مخرج فیلترها را می‌یابیم و با دستور *filter* و عبور سیگنال‌های *upsample* شده از این فیلترها، به سیگنال‌های بازسازی شده می‌رسیم. در ادامه این سیگنال‌ها رسم می‌شوند. نتایج در تصاویر زیر مشاهده می‌شوند.





با توجه به تصاویر مشخص است از آنجایی که فرکانس سیگنال اصلی $1KHz$ می باشد، نرخ نایکوییست برابر $2KHz$ بوده و اگر فرکانس نمونه برداری کمتر از این مقدار باشد، سیگنال به درستی بازسازی نخواهد شد. در دو تصویر اول به علت کمتر بودن فرکانس نمونه برداری از نرخ نایکوییست عدم بازسازی درست سیگنال امری مورد انتظار است. در مورد تصویر سوم، هنگامی که با این نرخ نمونه برداری کنیم، مطابق سومین شکل قسمت قبل به تمام نمونه های ما برابر صفر خواهند بود و سیگنال نمونه برداری شده سیگنالی متعدد با صفر است که در اثر عبور از فیلتر پایین گذر نیز صفر می ماند و به این دلیل سیگنال بازسازی نشده است. در تصویر چهارم از آنجایی که فرکانس نمونه برداری از نرخ نایکوییست بیشتر است، بازسازی به درستی انجام گرفته و اعوجاج های مشاهده شده ناشی از اعمال فاز به تبدیل فوریه سیگنال نمونه برداری شده در اثر عبور از فیلتر با تورث است که علاوه بر اعمال فاز خطی بر تبدیل فوریه که منجر به انتقال در حوزه زمانی می شود، با اعمال فاز غیر خطی باعث غیرخطی شدن خروجی نیز شده است. اما سیگنال به درستی بازسازی شده و با یک انتقال تا حد بسیار خوبی بر سیگنال اصلی منطبق می گردد.