****

دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده برق و کامپیوتر

**گزارش آزمایش شماره 7**

آزمایشگاه پردازش بی‌درنگ سیگنال‌های دیجیتال

پاییز 1400

|  |  |
| --- | --- |
| علی ساعی زاده | نام و نام خانوادگی |
| 810196477 | شماره‌ دانشجویی |

­

فهرست

[چکیده 3](#_Toc92670527)

[2-4-7 قسمت اول: پیاده سازی تابع کسینوس به صورت Fixed-Point 4](#_Toc92670528)

[توضیح پیاده سازی 4](#_Toc92670529)

[پیاده سازی روش اول (fcos1) 4](#_Toc92670530)

[پیاده سازی روش دوم (fcos2) 4](#_Toc92670531)

[پیاده سازی روش سوم به صورت محاسبات ممیز ثابت (icos) 5](#_Toc92670532)

[نتایج 5](#_Toc92670533)

[4-4-7 قسمت سوم: پیاده سازی فیلتر IIR به صورت Fixed-Point 7](#_Toc92670534)

[توضیح پیاده سازی 7](#_Toc92670535)

[نتایج 8](#_Toc92670536)

# چکیده

پردازنده‌های دیجیتالی برای نمایش اعداد به ناچار از تعدادی بیت محدود استفاده می‌کنند که این موضوع منجر به ایجاد اثراتی در پیاده سازی الگوریتم‌ها می‌شود. این اثرات شامل کوانتزیاسون، سرریز، محاسبات صحیح و ... می‌گردد که در عمل چالش‌هایی به دنبال دارد. در این آزمایش به محاسبات صحیح و برخی از اثرات ناشی از نمایش اعداد با تعداد بیت محدود شامل کوانتزاسیون و سرریز بررسی می‌شود.

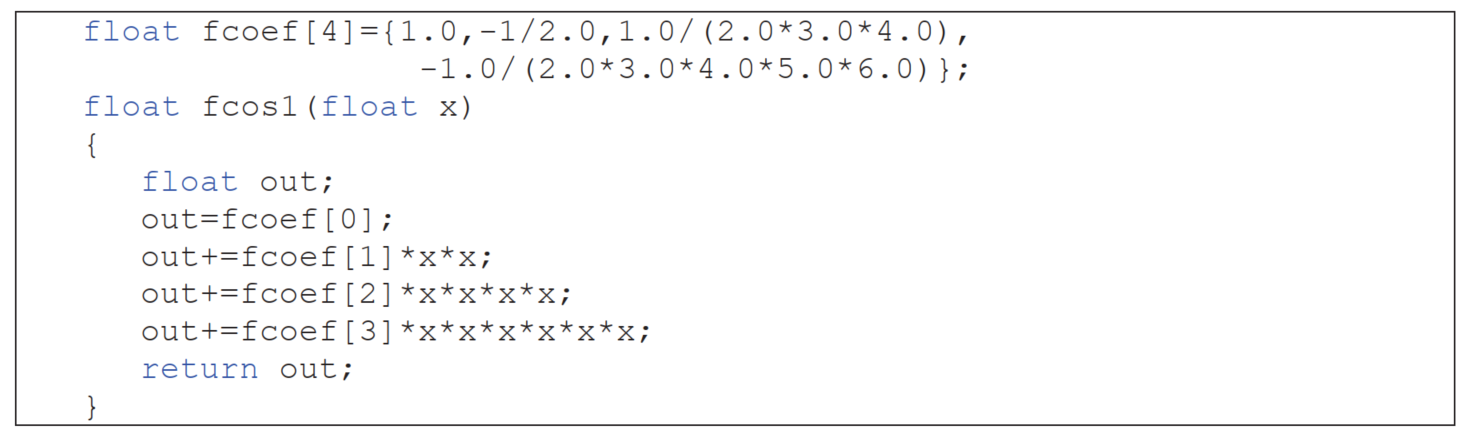
# 2-4-7 قسمت اول: پیاده سازی تابع کسینوس به صورت Fixed-Point

## توضیح پیاده سازی

به دلیل اینکه اغلب پردازنده‌های DSP به صورت fixed-point هستند در این قسمت تابع کسینوس را به کمک بسط مک لورن پیاده سازی می کنیم که رابطه آن بصورت زیر است:

### پیاده سازی روش اول (fcos1)

در این قسمت از قطعه کد زیر برای پیاده سازی استفاده شد.

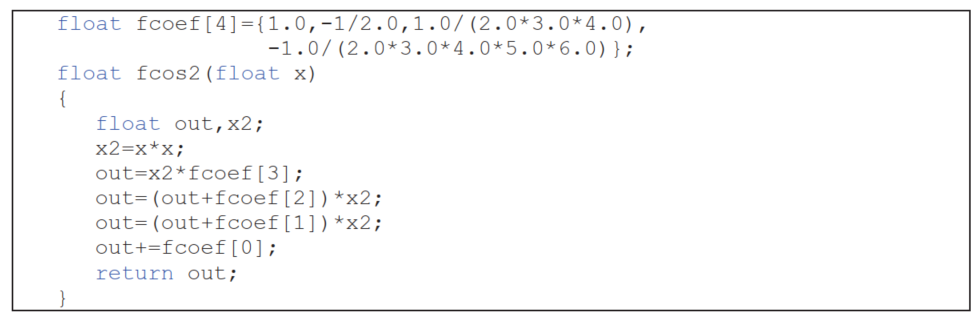


شکل 1 کد روش اول پیاده سازی تابع کسینوس

با توجه به نحوه پیاده سازی این قسمت، به این دلیل که از جملات محدود بسط مک لورن استفاده شده است بنابراین تنها آرگومان ها با مقادیر کوچک قابل محاسبه هستند برای محسابه آرگومان‌های بزرگ تر باید از جملات بیشتر بسط مک‌لورن استفاده شود.

### پیاده سازی روش دوم (fcos2)

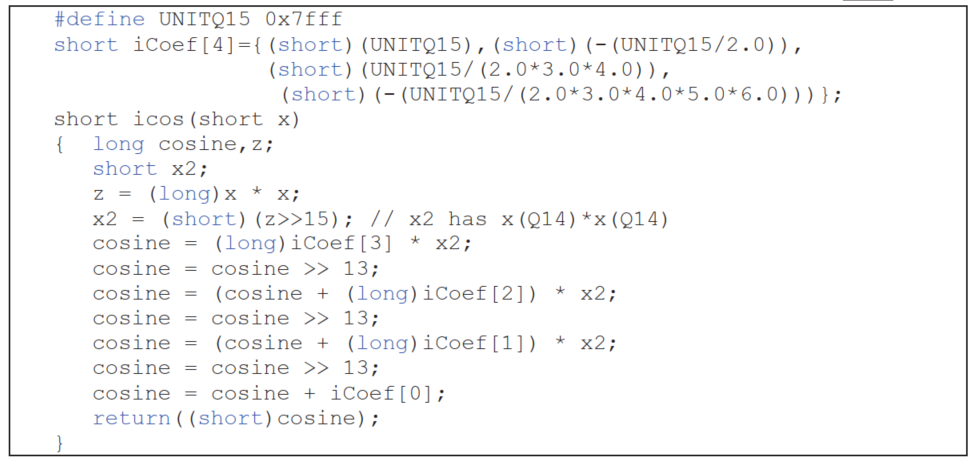
از کد زیر برای پیاده سازی قسمت دوم استفاده شد.



شکل 2 کد روش دوم پیاده سازی تابع کسینوس

### پیاده سازی روش سوم به صورت محاسبات ممیز ثابت (icos)

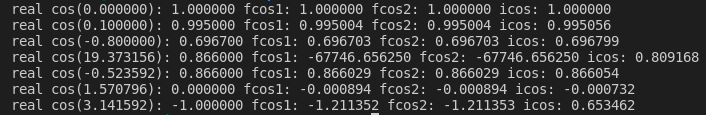
در این روش مقدار متغیر cosine 13 بیت به راست انتقال پیدا کرده است. دلیل آن است که به علت ضرب متغیر به فرمت 28 بیتی تبدیل می‌شود. اما فرمت استاندارد ما برای این قسمت 15 بیتی است. بنابراین پس از هر ضرب، 13 شیفت به راست می‌دهیم تا دوباره به فرمت 15 بیتی برسیم.



شکل 3 کد روش سوم پیاده سازی تابع کسینوس

## نتایج

خروجی کد برای ورودی های مختلف به شکل زیر بدست آمد که به کمک جدول کامل می‌شود.



شکل 4 خروجی کد پیاده سازی روش های مختلف

جدول 1 نتیجه تست پیاده سازی‌های مختلف برای تابع کسینوس

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| icos | fcos2 | fcos1 |  |  |
| 1.000000 | 1.000000 | 1.000000 | 1.0000 | 0 |
| 0.995056 | 0.995004 | 0.995004 | 0.9950 | 0.1 |
| 0.696799 | 0.696703 | 0.696703 | 0.6967 | -0.8 |
| 0.809168 | -67746.656250 | -67746.656250 | 0.8660 | 19.373155 |
| 0.866054 | 0.866029 | 0.866029 | 0.8660 | -0.523592 |
| -0.000732 | -0.000894 | -0.000894 | 0.0000 | 1.5707963 |
| 0.653462 | -1.211353 | -1.211352 | -1.0000 | 3.141592 |

همانطور که در قست های قبل اشاره شد، دو روش اول به علت اینکه از جملات محدود بسط مک لورن استفاده می‌کنند در آرگومان های بزرگ دچار مشکل می‌شوند که در آرگومان 19.373155 این موضوع مشخص شده است.

همچنین در آرگومان 3.141592 خطا برای همه روش به نسبت بقیه آرگومان ها زیاد است اما در روش سوم خطا قابل توجه تر است دلیل آن است که همانطور در دستور کار اشاره شد ورودی باید بین -2 و 2 باشد تا با فرمت استفاده شده قابل محاسبه باشد.

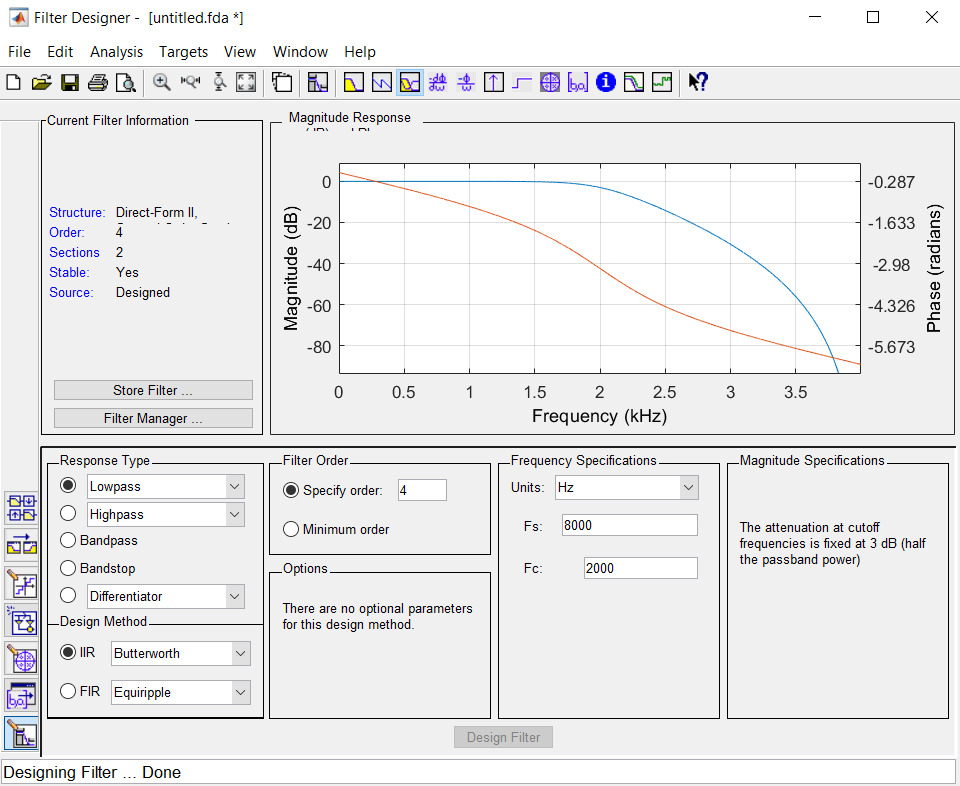
تمامی روش ها، به ازای آرگومان های کوچک با دقت خوبی عمل می‌کنند.

# 4-4-7 قسمت سوم: پیاده سازی فیلتر IIR به صورت Fixed-Point

## توضیح پیاده سازی

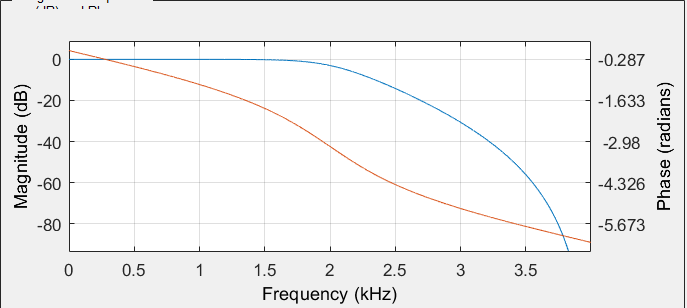
در این قست یک فیلتر IIR به صورت fixed-point طراحی و تست شد. مشخصات این فیلتر یک فیلتر پایین گذر با فرکانس قطع 2KHz است و فرکانس نمونه برداری نیز 8 KHz می‌باشد.

این فیلتر مرتبه 4 درمتلب به کمک ابزار fdatool طراحی به شکل زیر طراحی شد.



شکل 5 طراحی فیلتر در محیط fdatool

پاسخ دامنه و فاز این فیلتر به شکل زیر بدست آمد. مشخصات و ضرایب آن در فایل IIR\_Filter.fcf ذخیره شد.



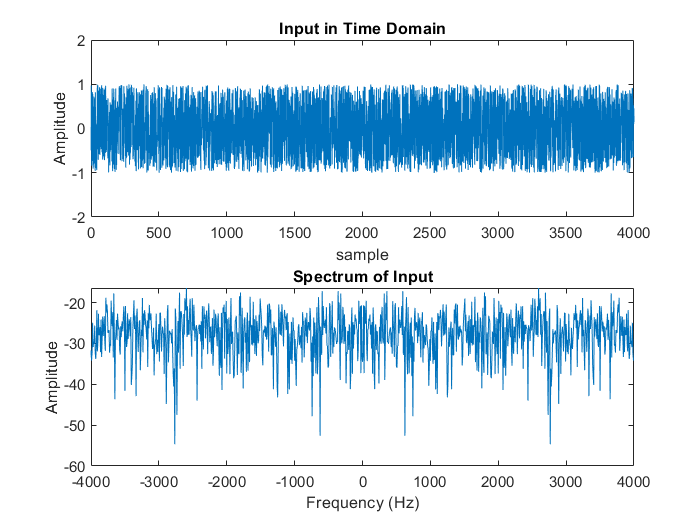
شکل 6 اندازه و پاسخ فیلتر بدست آده

برای تولید سیگنال از کد gen\_random.py استفاده شد که 4000 مقادیر رندوم را بین -1 و 1 تولید و در فایل input.txt جهت ورودی دادن به کد C ذخیره می‌کند.

کد فیلترینگ سیگنال در فایل Lab7\_2.c قابل مشاهده است. خروجی این قسمت در فایل out.txt ذخیره می‌شود تا در نرم‌افزار MATLAB بررسی شود.

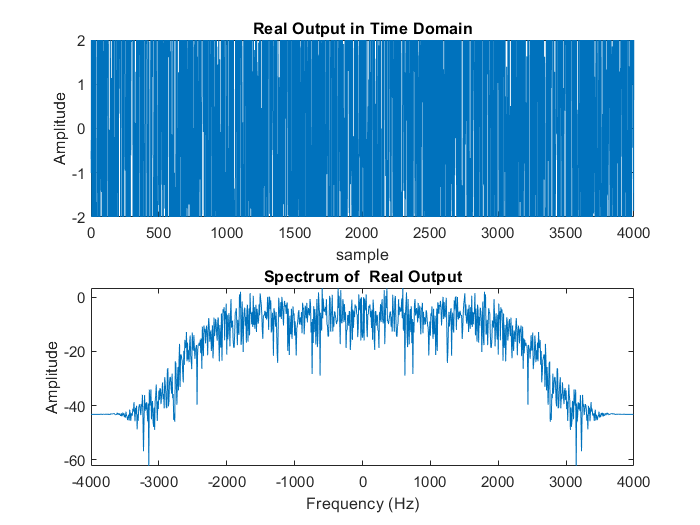
## نتایج

سیگنال ورودی در حوزه زمان فرکانس به شکل زیر می‌باشد.



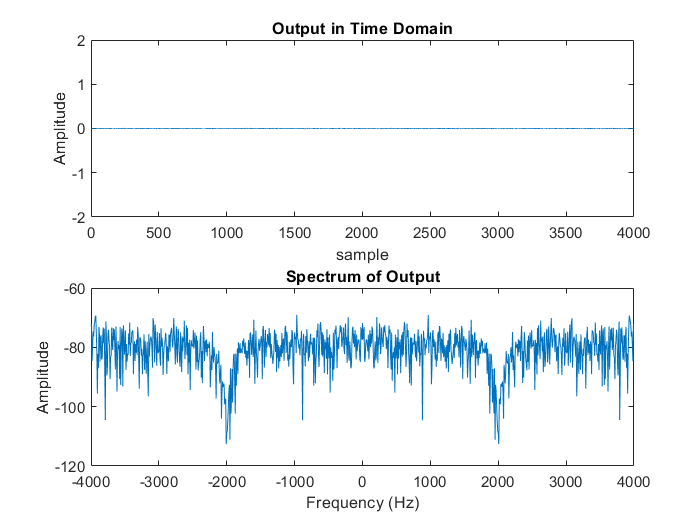
شکل 7 سیگنال ورودی در حوزه زمان و فرکانس

همچنین خروجی به کمک دستور sosfilt متلب و ساتفاده از فیلتر محاسبه شد که در سکل زیر قابل مشاهده است. همانطور که انتظار می‎رفت فرکانس‌های بالا فیلتر شده اند.



شکل 8 سیگنال فیلتر شده توسط متلب

اما خروجی به کمک فیلتر fixed-point که در زبان C پیاده سازی شد به شکل زیر است که کاملا کارایی خود‌را از دست داده است.



شکل 9 سیگنال فیلتر شده توسط کد fixed-point