COMPUTER HOMEWORK1
SIGNAL AND SYSTEMS

2021

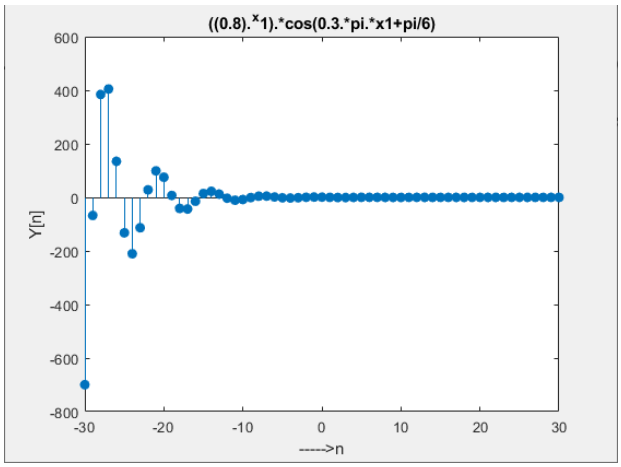
رادمهر کریمیان ۹۸۱۰۳۵۵۶

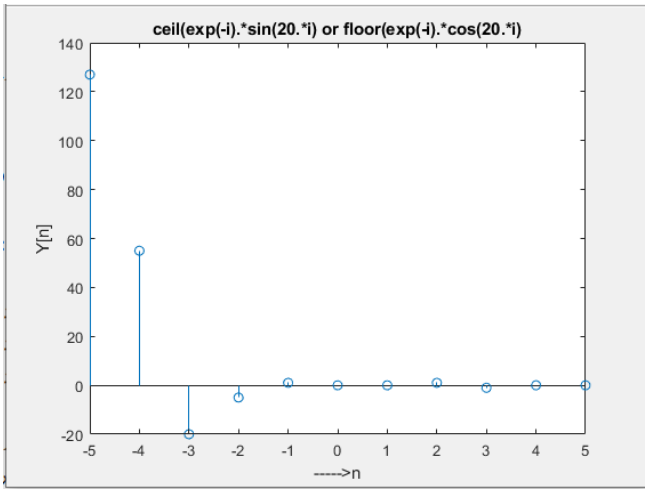
*قسمت هایی از کد باید به شکل کلین کد نوشته شده است و برای مطالعه به آن مراجعه شود.

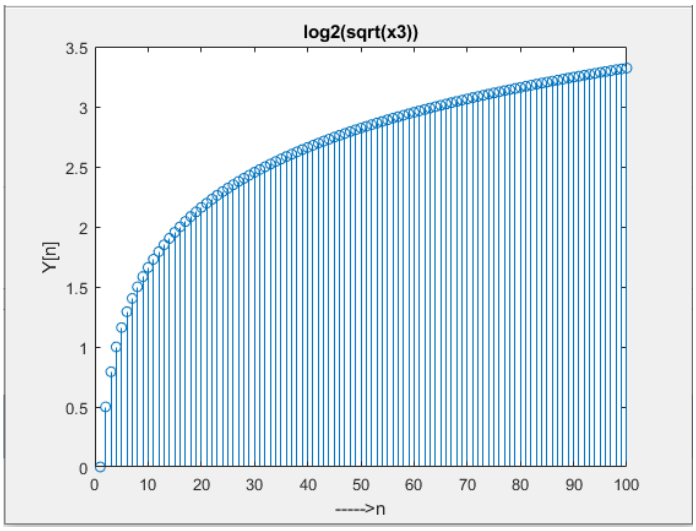
قسمت الف؛تولید سیگنال و رسم آن(در قسمت SS_HW1_part1)

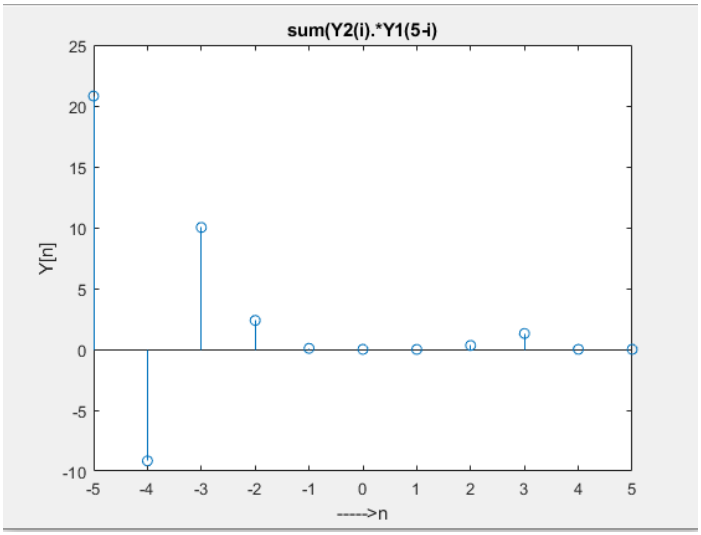
1.برای رسم دستور ابتدا مقادیر n را ست میکنیم.سپس در هر nمقدار تابع را کشیده و سپس با رسم آن به وسیله دستور stem آن را رسم میکنیم.

تصاویر نمودار ما به شرح زیر است:

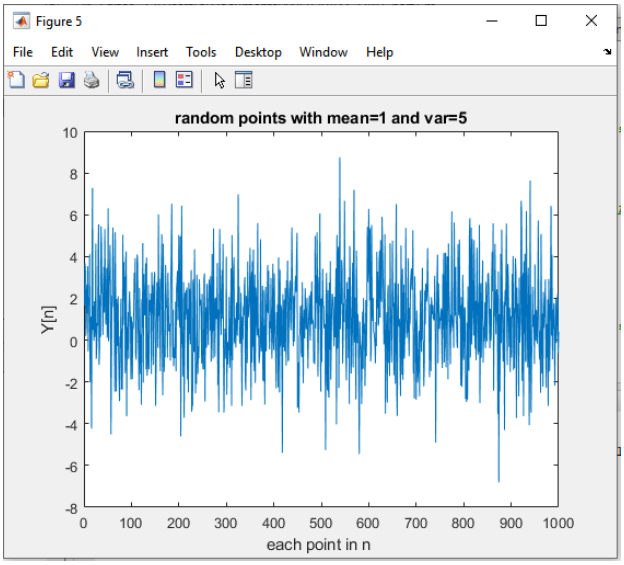




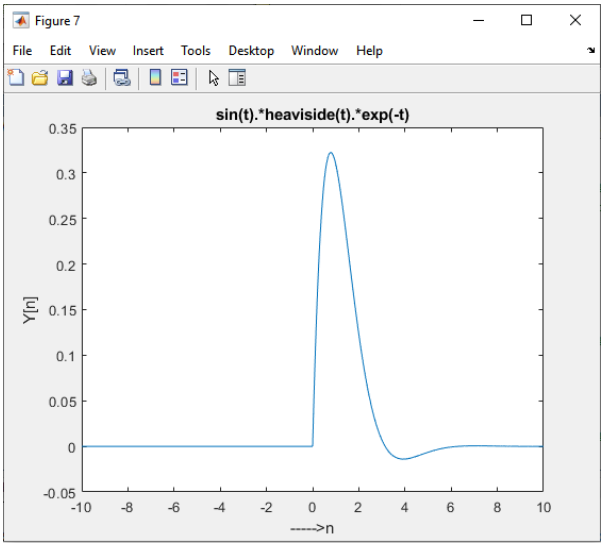




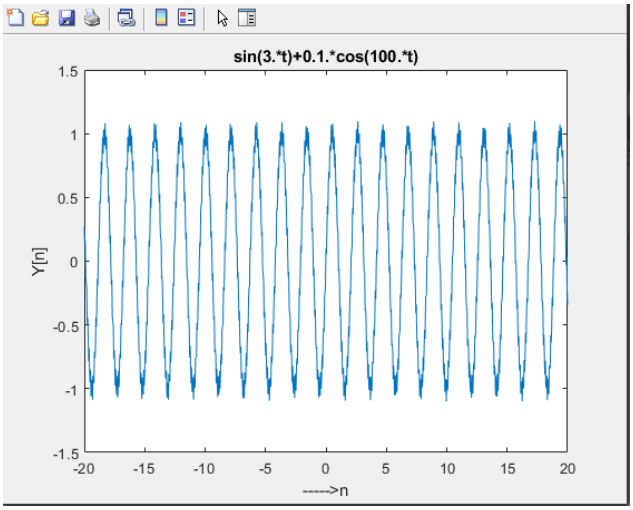
۲.برای تابع randnداریم که اعدادی تصادفی با توضیع نرمال میانین صفر و وارینس یک تولید می شود با توجه به تعریف توضیع نرمال کافیست که فقط مقدار واریانس را در همان مقدار تولید شده ضرب کرده و سپس با میانگین دلخواه جمع کرده تا میانگین مورد نظر ما تولید شود.



3.برای رسم توابع پیوسته به این شکل عمل میکنیم که تابعمان را ابتدا پارمتری تعریف کرده و سپس با نمونه برداری به تعداد نقاط لازم در طول بازه و رسم مقدار تابع در آن نقطه به شکل تابع می رسیم:

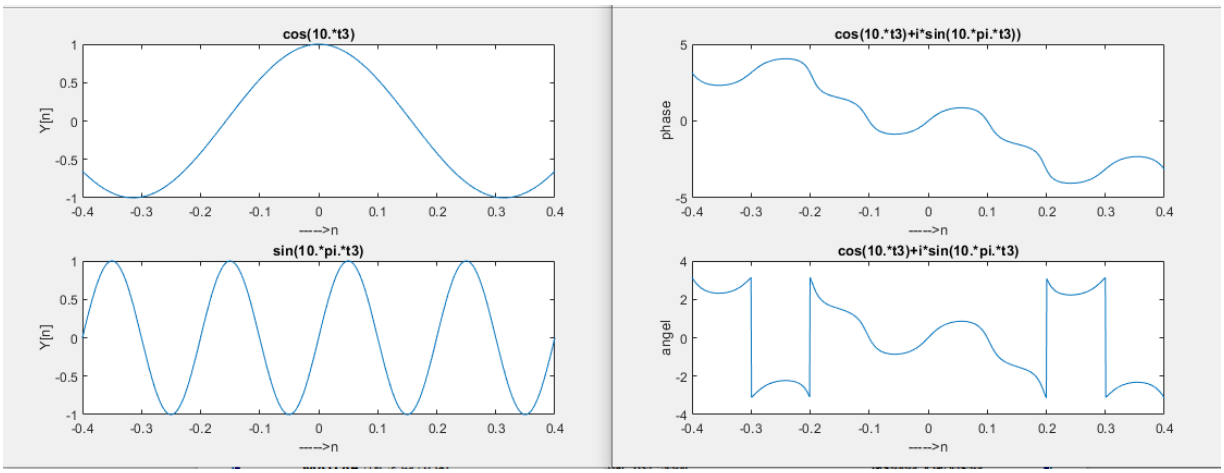


دقت کنید درمواردی مانند شکل زیر اگر تعداد نقطه برداری های ما کم باشد شکل ما کمی مشکل دارد:



که با بالابردن تعداد نقاط این مشکل رفع می شود.

برای نمودار تابع مختلط داریم:



در مورد تفاوت angelو phase داریم که angelجواب همواره بین pi–تا pi است

ولی درمورد phase **میتواند بیشتر از پی و کمتر از منفی نیز باشد که در شکل قابل مشاهده است.**

همچنین با دستور real**و** imag **نیز میتوان بخش مختلط و حقیقی را جدا کرد و نیز با دستور سابپیلوت چند نمودار در یک فیگور رسم میکنیم.**

قسمت دوم؛ماتریس اعداد تصادفی(در قسمت SS_HW1_part2)

در مورد تفاوت rand**و** randn **داریم که** rand **توضیع یونیفرم اعداد تصادفی است ولی** randn **توضیع نرمال اعداد تصادفی می باشد.**

در مورد تایم گزارش شده داریم:

```
Elapsed time is 0.000499 seconds.
Elapsed time is 0.001710 seconds.
```

که میبینیم تقریبا تایم الگوریتم جمع ما 3.5 **برابر خود تابع جمع متلب می باشد که علت این زمان فاحش به علت الگوریتم هایی است که در تابع سام خود متلب از نظر زمانی آن را پیشتاز میکند.**

قسمت سوم؛elementwise **و** vectorization **در متلب**(در قسمت SS_HW_part3)

برای انجام عملیات به صورت elementwise **فقط کافیست که قبل عملیات که در اینجا ضرب است . بگذاریم یعنی بنویسیم** *. **همچنین اگر از** * **استفاده کنیم درواقع به صورت عادی ضرب کرده ایم.**

در مورد تایم محاسبه شده داریم:

```
>> SS_HW1_part3
Elapsed time is 0.000257 seconds.
Elapsed time is 0.000710 seconds.
>> SS_HW1_part3
Elapsed time is 0.000698 seconds.
Elapsed time is 0.000659 seconds.
>> SS_HW1_part3
Elapsed time is 0.000175 seconds.
Elapsed time is 0.000639 seconds.
>> SS_HW1_part3
Elapsed time is 0.000536 seconds.
Elapsed time is 0.001277 seconds.
```

دقت کنید که ممکن هست حتی الگورتیم ضرب با حلقه ما نیز بعضا سریع تر باشد اما این اختلاف اندک باعث می شود که بگوییم که الگوریتم ساده و بیسیک ما همواره در کند تر مساوی الگوریتم ضرب متلب باشد که به صورت کلی میتوان گفت الگوریتم متلب بین 2 تا 3 برابر سریع تر است که علت آن استفاده از الگوریتم های پیچیده جبر خطی می باشد.

قسمت چهارم؛کانولوشن(در قسمت SS_HW_part4)

دو تابع ما در فایل های conv_loop و cov_noloop تعریف شده است:

در قسمت conv_loop داریم که به این شکل تابع تعریف می شود که ابتدا دو ماتری و آرایه خودمان رو هم طول میکنیم به این شکل که طول دو سیگنال برابر جمع دو طولشان بشود.در ادامه طول دو سیگنال آن را صفر قرار می دهیم تا به طول دلخواه برسیم.

در ادامه ماتریس کانولوشن رو صفر قرار میدهیم.سپس لوپی به شکل سیگماو تعریف کانولوشن تعریف میکنیم و مقدار کانولوشن را سیو میکنیم. در ادامه با لوپی دیگر در ماتریس کانولوشن جلو میرویم.

در قسمت conv_noloop از قضیه کانولوشن استفاده میکنیم که مثال و شبیه آن را در سوال 3 تمرین داشتیم.

میدانیم که :

$$h(x) = \{f * g\} = \mathcal{F}^{-1}\{F.G\}$$

حال با استفاده از دستور fftو ifft ابتدا تبدیل فوریه دو سیگنال را حساب کرده در هم ضرب کرده و سپس فوریه معکوس میگیریم تا به تابع کانولوشن برسیم.

در مورد تایم هر کدام از توابع داریم:

```
>> SS_HW1_part4
Enter x(with[]):    [2 3 4 2 2 4]
Enter h(with[]):    [2 4 2 5 2 5 2 5 3 2]
Elapsed time is 0.008749 seconds.
Elapsed time is 0.009281 seconds.
Elapsed time is 0.003613 seconds.
```

که در مورد تایم ران ها به ترتیب داریم که تایم تابع بدون حلقه ا همه کمتر سپس تایم تابع کانولوشن در متلب و سپس و در نهایت تایم کانولوشن با حلقه را داریم.

توجه کنید که درست است تابع نوشته شده ما سریع تر است اما اگر به خروجی خواسته شده نگاه کنیم متوجه می شویم که در نقاطی که انتظار مقدار صفر را داشتیم مقدار ما اندکی با صفر فاصله دارد(در اورد ده به توان منفی شانزده) که علت آن خطای ما در تبدیل فوریه و تبدیل فوریه معکوس است. اما با تقریب خیلی خوبی جواب کانولوشن ما درست است اما این مقدار خطای کم ناشی از عملگر های fft و ifft است که استفاده کرده ایم.

در ادامه و در فایل متلب نیز دو ورودی گرفته و کانولوشن آن ها محاسبه و به عنوان خروجی داده می شود.

قسمت پنجم،خواص کانولوشن(در قسمت SS_HW1_part5)

با مشاهده خروجی ها به شکل کاملا مشهودی می توان مشاهده کرد که خواص به درستی قابل مشاهده هستند.

اما در تعریف و محاسبه کانولوشن با تابع دلتا برای تابع دلتای یک شیفت به راست محاسبه شده است اما دقت کنید که در متلب تابعک دلتای دیراک بدین شکل تعریف شده است که در نقطه ما مقدار inf را اختیار می کند.

پس ما در خود تابعک دلتا چک میکنیم که چه نقطه ای مقدار inf را گرفته و تابعی مشابه تابع دلتای دیراک تعریف میکنیم با ابن تفاوت که در نقطه مطلوب مقدار یک را به جای بینهایت یا inf گرفته است.

بقیه خواص نیز با مشاهده خروجی ها قابل بررسی هستند.

قسمت ششم؛سیستم(در قسمت SS_HW1_part6)

در مورد جواب قسمت اول ابتدا تابع ورودی را تعریف میکنیم که f نامیدیم.

دقت کنید که h1 یک سیگنال نامتناهی خواهد بود اما از آنجایی که ما میخواهیم در یک بازه آن را ببینیم پس x را به هرگونه که بخواهیم و در هر بازه ای تعریف میکنیم.

مشکلی پیش نمیاد.

دقت کنید که در تابع Heaviside که تابع پله خود متلب می باشد به علت اینکه خوش تعریف باشد مقدار در صفر را نیم تعریف کرده است.

ما در قسمت های قبلی این را پذیرفتیم اما در این قسمت برای راحتی کار مقدار آن را دست کاری کرده و به یک تغییر داده ایم.

همچنین داریم که:

$$x1\ [n\ ] = -2x\ 1\ [n\ -\ 1] + x\ [n\ ] + 3x\ [n\ -\ 1];$$

که به سادگی قابل مشاهده است که ماتریس b,a ما برای فیلتر به شکل زیر است:

$$a = [1 \quad 2], b = [1 \quad 3]$$

در مورد تابع تعریف شده x که به آن نام f را دادیم نیز مقادیر را میدانیم پس با دستور فیلتر جواب سیستم را میابیم.

برای سیستم شماره دو نیز داریم:

$$x\,2\,[n\,] = -3x\,2\,[n\,-\,1] + x\,1\,[n\,] + 3x\,1\,[n\,-\,1] + x\,1[n\,-\,2]$$

که در آن داریم:

$$a = [1 \quad 3 \quad 2], b = [1 \quad 2]$$

و به راحتی به دستور impz جواب آن را بدست می آوریم.

در مورد کانوالو کردن نیز به ترتیب سیستم کانوالو کرده اما در قسمت جمع کردن به این شکل عمل میکنیم که ابتدا f4 و f2 را هم طول میکنیم که طول آن برابر طول سیگنال بزرگ تر است و ادامه f2 را صفر میگذاریم.