MATLAB HW1 Report

Ali Ghavampour

97102293

Sharif University of Technology

Signal and Systems

Dr. Hamid K. Aghajan

لطفا نکات زیر را مورد توجه قرار دهید:

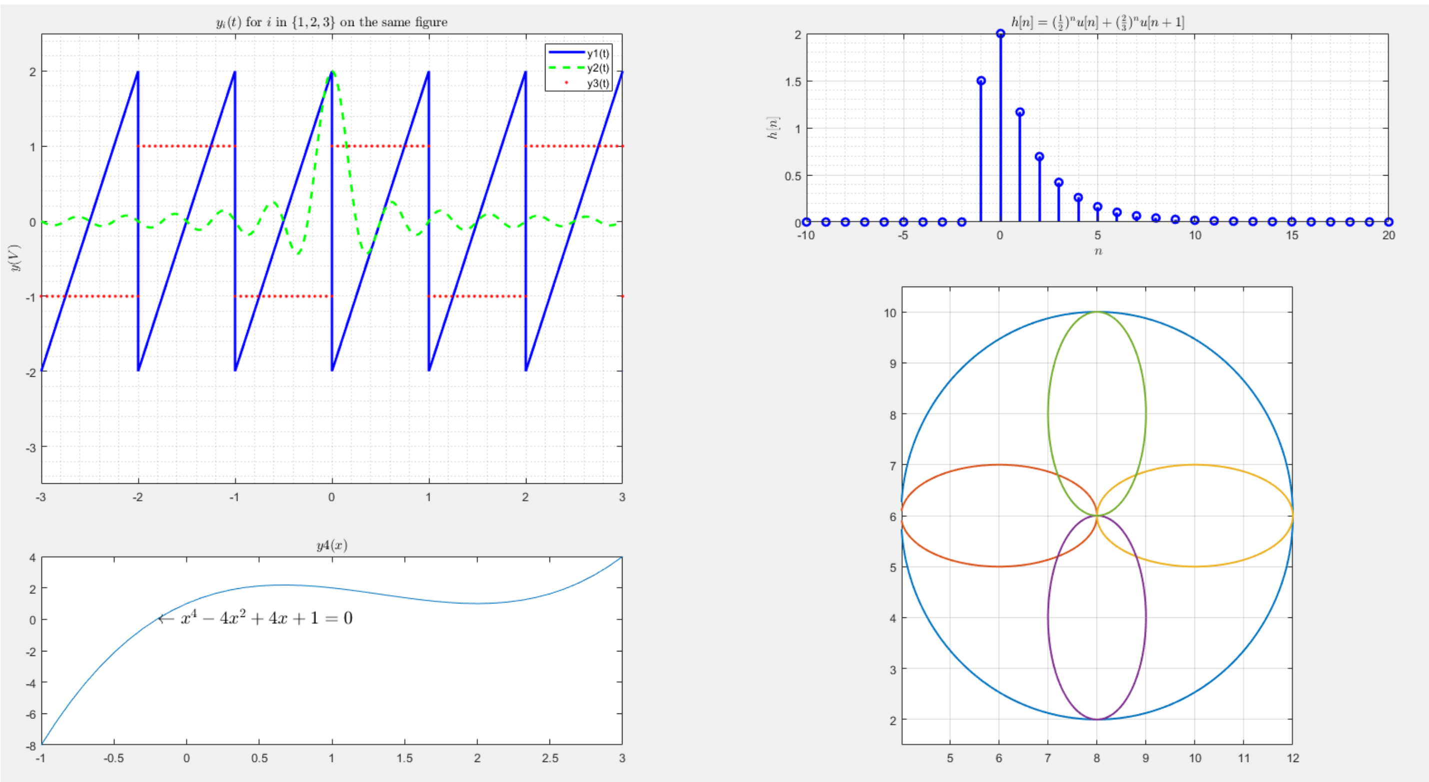
در این تمرین تمام فانکشن ها در فایل کد اصلی و در section فانکشن نوشته شده است.

همچنین در سوال 1، در بخش 3 از قسمت 1.2 از تابع groupcount استفاده شده است که از MATLAB 2019 به بعد در دسترس است!

در کد از توابعی مانند sum به صورت sum(matrix,'all') استفاده شده است که اصولا مواردی مانند 'all' از MATLAB 2018 به بعد در دسترس هستند.

Section 1

1.1. Plot



همانطور که مشاهده می‌شود با دقت بسیار بالا شکل خواسته شده رسم شده است.

برای تایتل ها و لیبل های نمودار از LaTex استفاده شده است.

برای رسم بیضی تابع ellipse به صورت دستی نوشته شده است که در قسمت فانکشن های کد اصلی موجود است.

1.2. Matrix Calculations

در این قسمت همانطور که خواسته شده بود از هیچگونه دستور for و while استفاده نشده است.

جای گذاری اعداد خواسته شده در قسمت پایین و بالای قطر اصلی ماتریس به کمک جمع ماتریس با ماتریس های پایین مثلثی و بالا مثلثی انجام شده که تولید این ماتریس ها با توابع tril() و triu() انجام شده است.

برای نرمالایز کردن سطر ها به 4، ماکسیموم هر سطر به کمک تابع max() به صورت یک آرایه در آمده و با دستور تقسیم المان به المان یعنی ./ کل سطر به ماکسیموم خودش تقسیم شده و در 4 ضرب شده است.

برای صفر کردن قطر فرعی ماتریس نیز از تابع flip() استفاده شده که به نوعی ماتریس را می‌چرخاند.

برای جایگذاری میانگین خانه های اطراف به جای NaN ابتدا ایندکس های خانه هایی که این المان در آنها ها قرار دارد به کمک تابع isnan پیدا شده و به تابع avger که به صورت دستی نوشته شده است داده می‌شود. عملکرد این تابع اینگونه است که با دریافت ایندکس هایی که بیان شد، خانه های اطراف خانه NaN را به صورت یک ماتریس 3x3 در می‌آورد و با صفر قرار دادن خانه NaN جمع بقیه خانه ها محاسبه می‌شود و میانگین از آن به دست می‌آید.

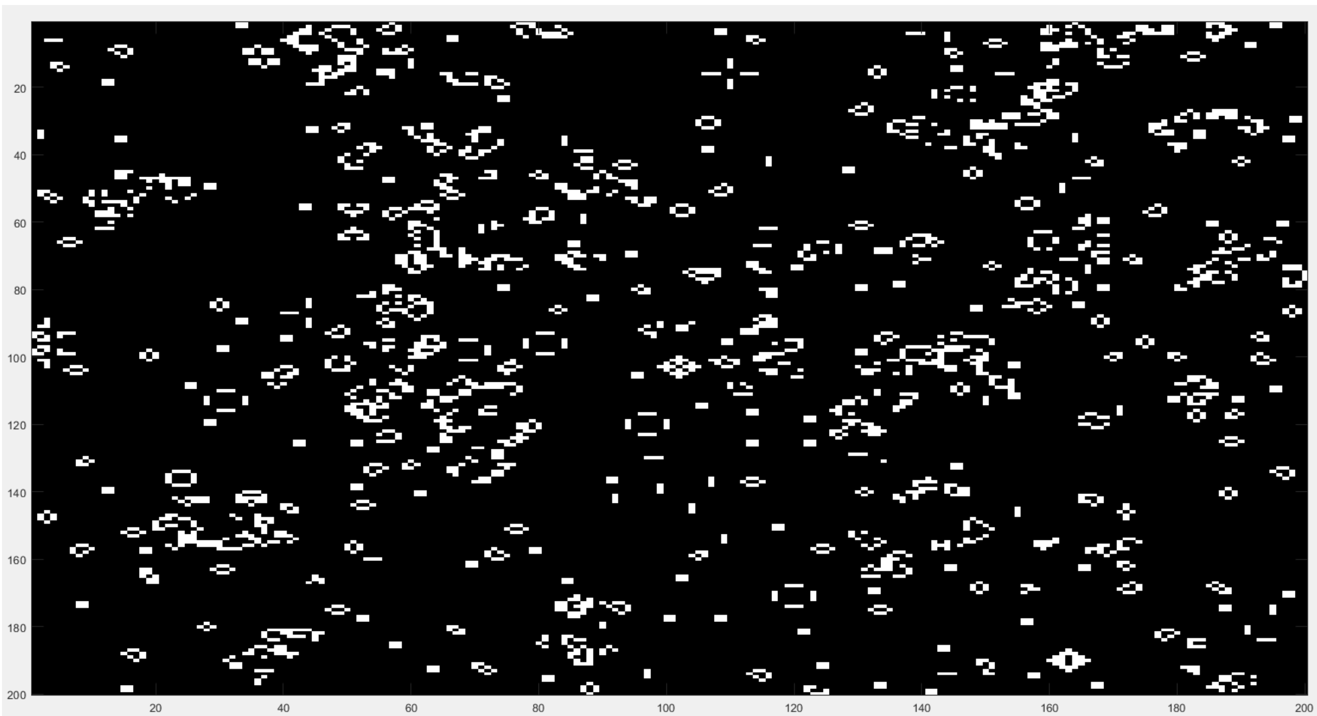
همچنین همانطور که در ابتدای داک گفته شد، برای بخش 3 که باید ستون هایی که بیشتر از چهار مورد در آن ها عدد 1 تکرار شده است را حذف کنیم از تابع groupcounter استفاده شده است که طبق مطالعه ای که داشتم تنها در متلب 2019 به بعد وجود دارد. نحوه عملکرد این تابع در کد به صورت کامنت توضیح داده شده است.

همچنین برای ساخت My_Cell به جای انجام اعمال اضافه با استفاده از تابع struct2cell()، استراکت ساخته شده به Cell تبدیل شده است.

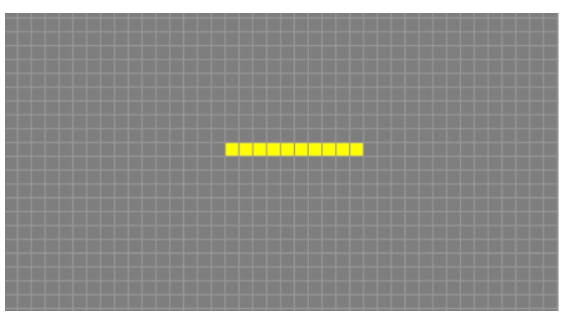
1.3. Conway’s Game of Life

برای این بخش یک تابع به همین اسم نوشته شده که در section توابع کد قرار دارد. عملکرد محاسبه تعداد سلول های زنده اطراف هر خانه دقیقا مانند تابع avger که در بالا بیان شد انجام شده است به طوری که خانه های اطراف را به صورت ماتریس 3x3 در می‌آورد و با جمع کردن آن ها تعداد زنده های اطراف هر خانه مشخص می‌شود. همچنین در این الگوریتم برای اینکه برای خانه های Edge به ارور Bound Limit ماتریس نخوریم، در ابتدا اطراف ماتریس City اصلی صفر اضافه شده است.

خروجی برنامه برای شهر 200x200:



همچنین برای بررسی صحت الگوریتم استفاده شده از مثال معروفی به نام 10 Cell Row استفاده شد که حالت اولیه آن به صورت زیر است:



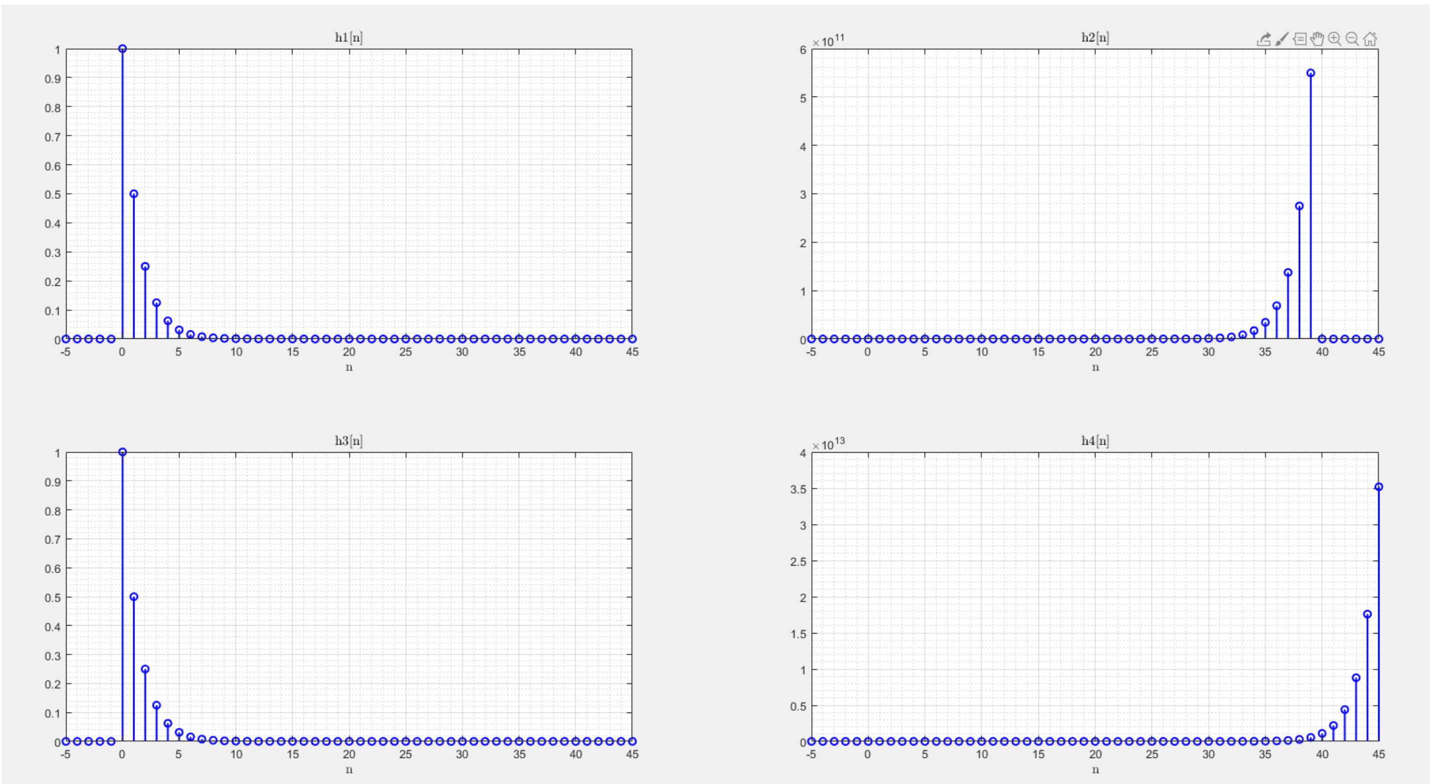
با مقایسه خروجی ای که باید برای این ستاپ دریافت کنیم با کد خودم، متوجه میشویم که الگوریتم کد درست است و به خوبی کار می‌کند. این مثال خاص به صورت کامنت در کد قرار داده شده تا در صورت تمایل برای بررسی صحت کد آن را امتحان کنید.

Section 2

2.1. Z-Transform and Zero-Pole Map

بخش 1)

خروجی متلب برای Plot چهار سیگنال خواسته شده:



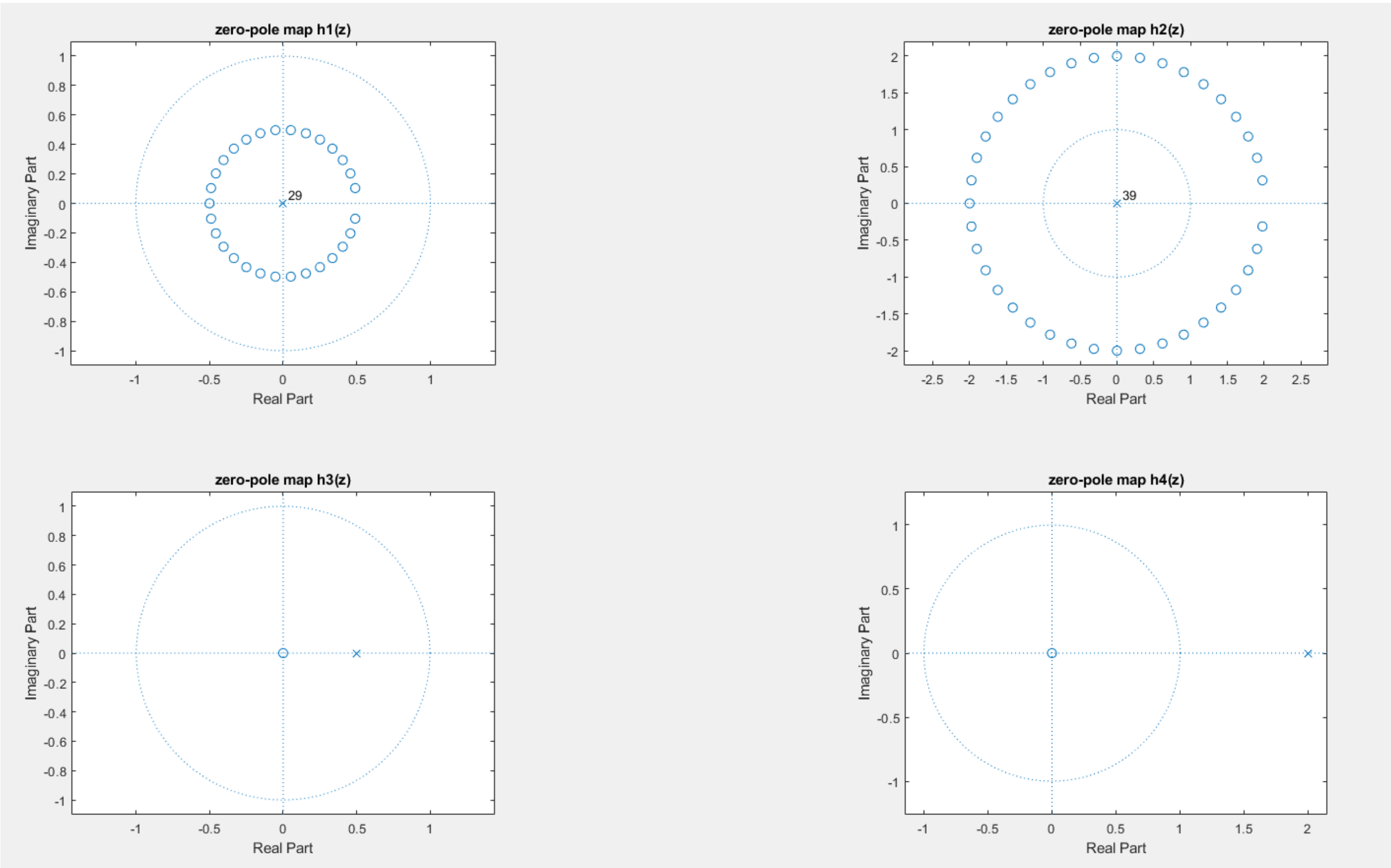
با استفاده از تابع ()ztrans تبدیل زد این سیگنال ها، مطابق زیر به دست آمده است:

```
H1 is:
   1                 1
------- - ----------------------- + 1
2 z - 1     29
          z    (2 z - 1) 536870912

H2 is:
  2      1099511627776
----- - ------------- + 1
z - 2      39
         z    (z - 2)
```

```
H3 is:
   1
------- + 1
2 z - 1

H4 is:
  2
----- + 1
z - 2
```

برای رسم نمودار صفر-قطب دو تابع ()zplane و ()pzmap وجود دارد که در اینجا ار تابع ()zplane استفاده شده زیرا این تابع مرتبه صفر یا قطب را هم کنار آن می‌نویسد و در کل خروجی مرتب تری می‌دهد.

نمودار های صفر-قطب برای این 4 سیگنال:



تحلیل پایداری h1(z):

یک راه برای بررسی پایداری h1(z) بررسی مطلقا جمع پذیر بودن این سیگنال است که در صورت نوشتن به سادگی به دست می‌آید که این سیگنال مطلقا جمع پذیر است و در نتیجه پایدار است. همچنین قضیه ای دیگر برای بررسی پایداری از طریق تبدیل زد سیگنال و ROC آن وجود دارد که اگر ROC این سیگنال شامل دایره |z|=1 باشد، در این صورت پایدار است. برای این سیگنال مشاهده می‌شود که ROC شامل این دایره می‌باشد پس این مورد هم نشان دهنده پایداری آن است. وجود دایره واحد در ناحیه همگرایی شرط لازم و کافی برای پایداری سیستم و سیگنال است.

تحلیل پایداری h2(z):

این سیگنال هم از n=40 به بعد صفر می‌شود و مطلقا جمع پذیر است پس پایدار است. همچنین ROC شامل دایره واحد می‌باشد.

تحلیل پایداری h3(z):

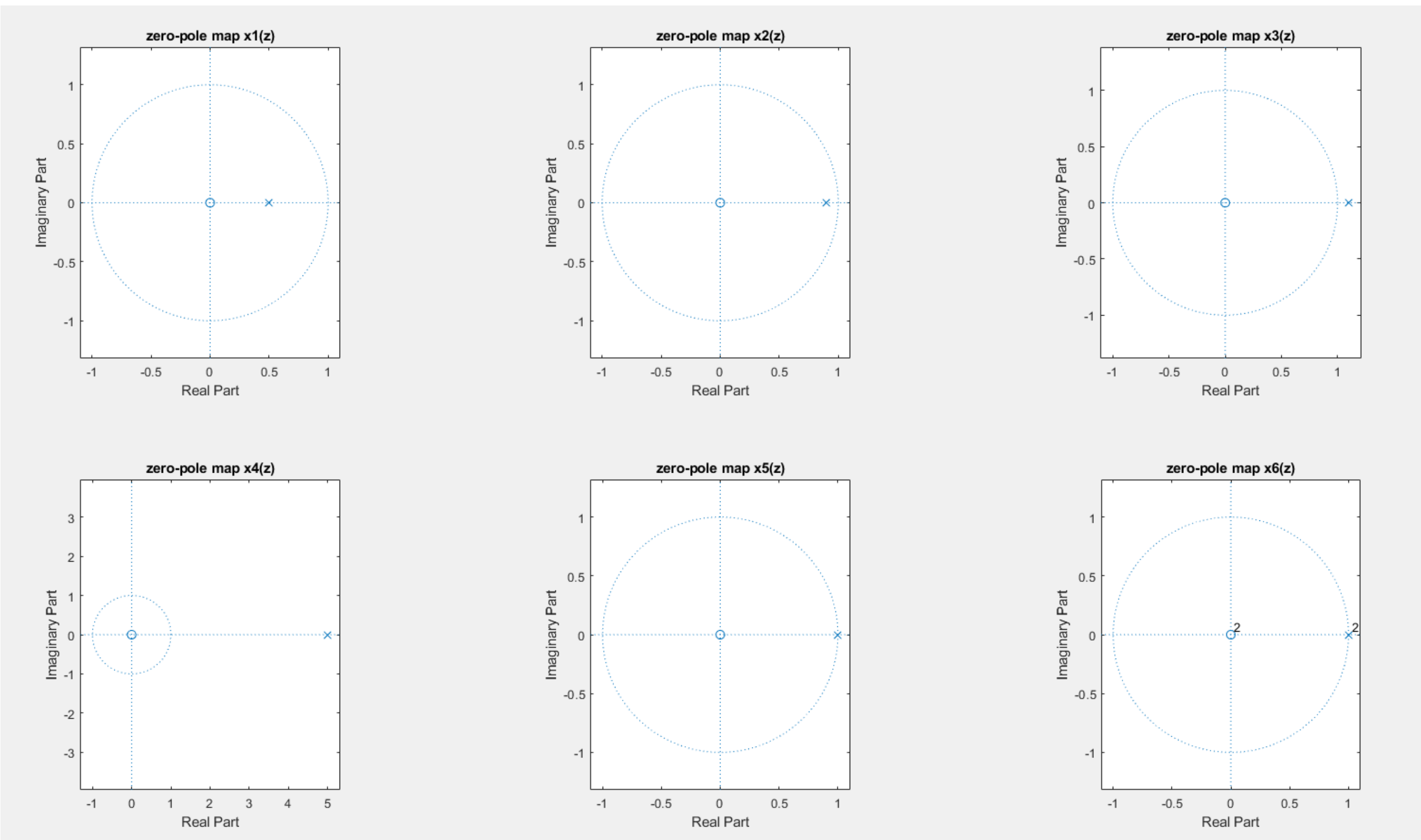
این سیگنال مطلقا جمع پذیر است (یک دنباله هندسی که جمع سری آن به یک عدد میل می‌کند) پس پایداری است. همچنین ROC آن شامل دایره واحد است.

تحلیل پایداری h4(z):

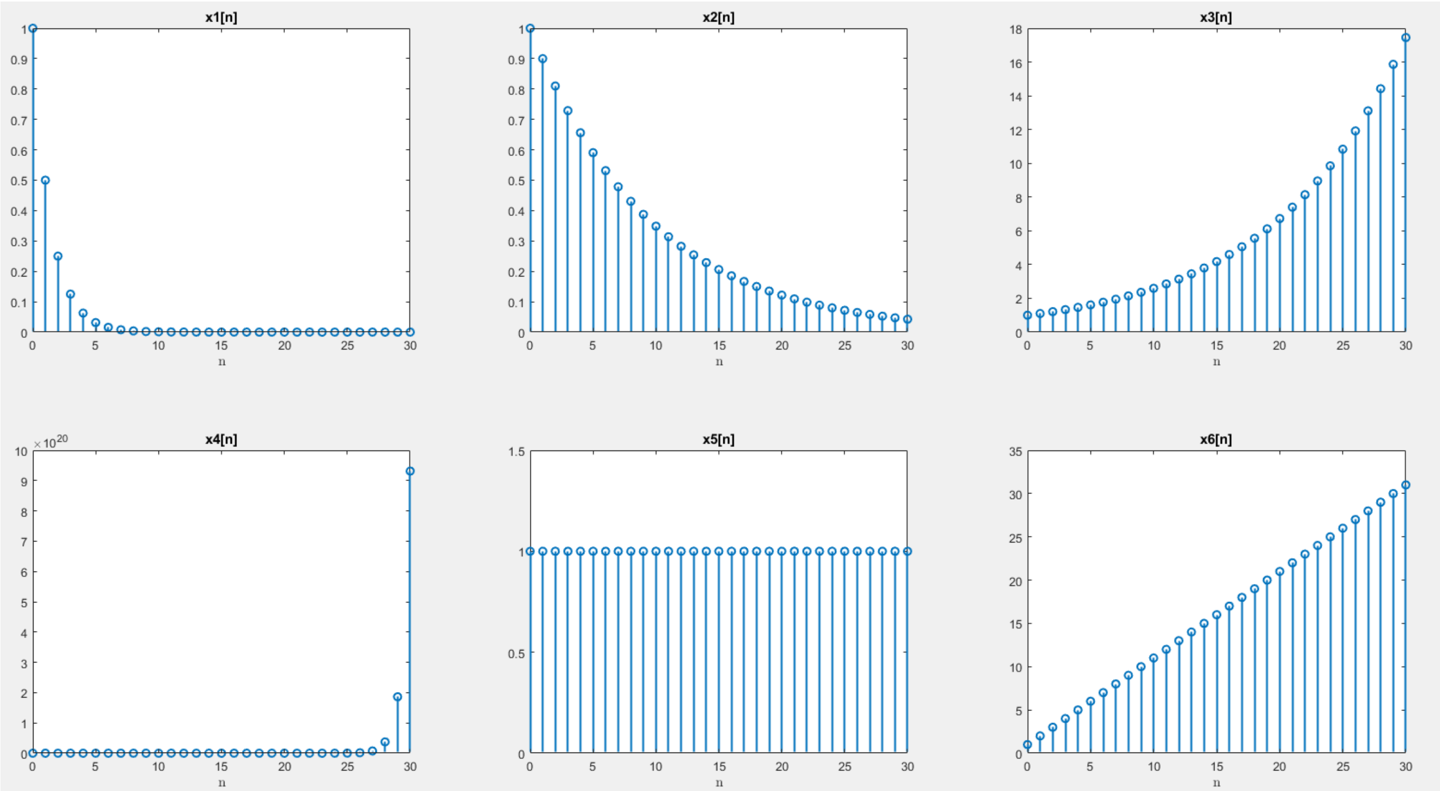
این سیگنال مطلقا جمع پذیر نیست پس ناپایدار است. همچنین ROC آن شامل دایره واحد نیست (از 2 تا بینهایت است).

بخش 2)

نمودار صفر-قطب 6 سیگنال:



نمودار پاسخ ضربه:



در یک دید کلی به نمودار های صفر–قطب و پاسخ ضربه ها متوجه می‌شویم که هر چقدر قطب داخل تر باشد نسبت به دایره واحد، سیگنال زود تر همگرا می‌شود و هر چقدر دور تر باشد نسبت به دایره واحد، سیگنال با سرعت بیشتری افزایش پیدا می‌کند و واگرا می‌شود. هر چقدر ضریب z^-1 بزرگتر از یک باشد، در بسط سیگمای تبدیل زد که در واقع ضرایب آن سیگنال ما در حوزه زمان را شکل می‌دهند، مشاهده می‌شود که افزایش ضرایب سرعت بیشتری دارد و همچنین واگرا است. هر چقدر هم اندازه ضریب از یک کوچک تر باشد برعکس این قضیه دیده می‌شود که علت ارتباط فاصله قطب ها از دایره واحد با ظاهر سیگنال است. همچنین مشاهده می‌شود که در صورتی که قطب روی دایره واحد باشد و از مرتبه یک باشد نمودار ثابت می‌شود و با افزایش مرتبه آن نمودار شکل خطی و در ادامه توانی پیدا می‌کند.

در تمام سیگنال های بالا ROC به صورت |z|>|a| می‌باشد که a قطب است.

تحلیل پایداری x1[n]:

سیگنال مطلقا جمع پذیر است (به سادگی از قواعد سری هندسی به دست می‌آید که به یک عدد همگراست) پس پایدار است. همچنین علتی دیگر برای پایداری حضور دایره واحد در ROC است. که مشخصا ROC برابر |z|>0.5 است.

تحلیل پایداری x2[n]:

سیگنال مطلقا جمع پذیر است پس پایدار است. همچنین دایره واحد در ROC این سیگنال وجود دارد. ROC : |z|>0.9

تحلیل پایداری x3[n]:

این سیگنال مطلقا جمع پذیر نمی‌باشد پس ناپایدار می‌باشد. همچنین دایره واحد در ROC آن قرار ندارد. ROC : |z|>1.1

تحلیل پایداری x4[n]:

سیگنال مطلقا جمع پذیر نمی‌باشد پس ناپایدار است. همچنین دایره واحد در ROC آن قرار ندارد. ROC : |z|>5

تحلیل پایداری x5[n]:

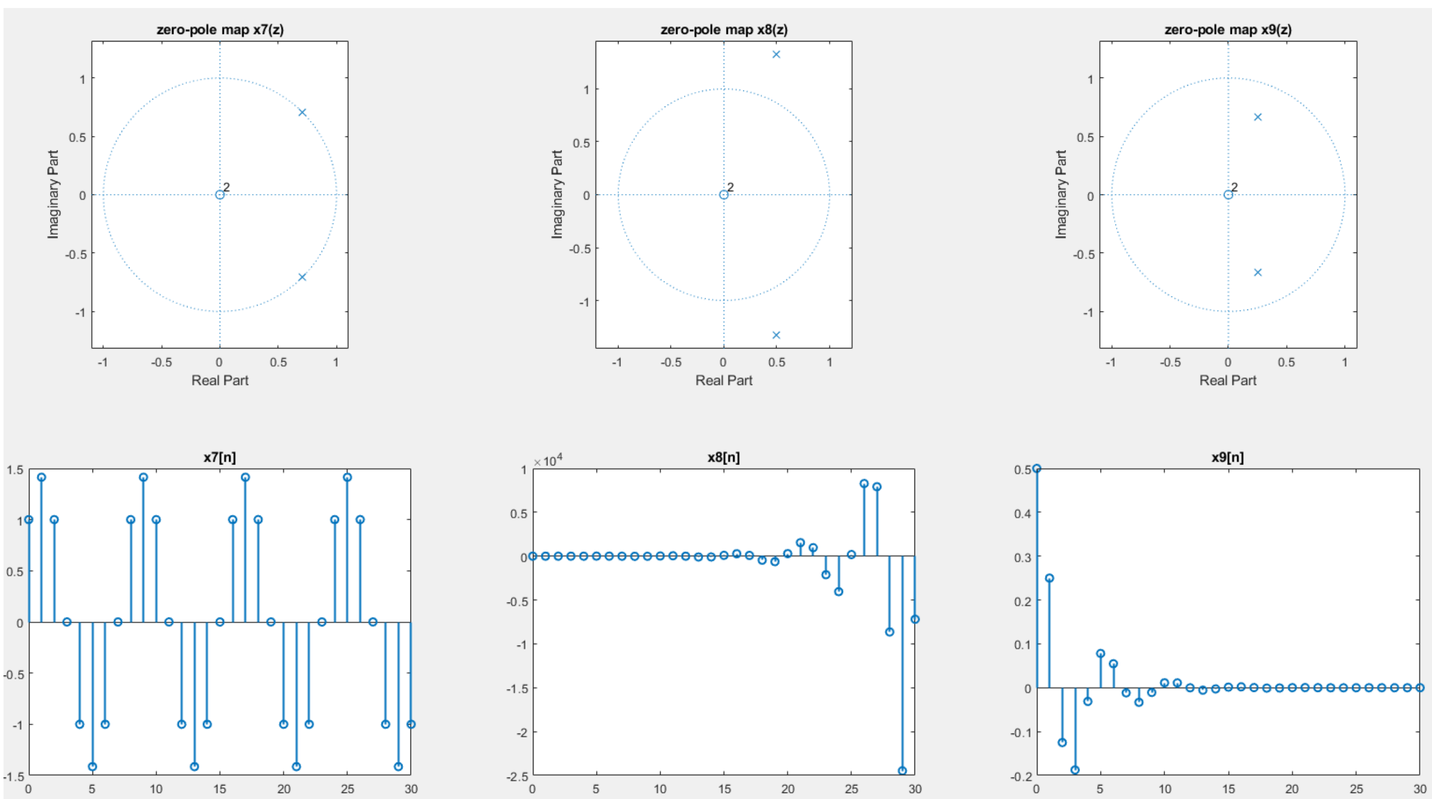
سیگنال مطلقا جمع پذیر نمی‌باشد پس ناپایدار است. همچنین دایره واحد در ROC آن قرار ندارد. ROC : |z|>1 ناحیه همگرایی دقیقا از بعد دایره واحد تا بینهایت است.

تحلیل پایداری x6[n]:

این سیگنال نیز پایداری نیست چون مطلقا جمع پذیر نیست. همچنین دایره واحد مانند سیگنال x5 در ناحیه همگرایی وجود ندارد.

بخش 3)

نمودار صفر-قطب و پاسخ ضربه این 3 سیگنال:



در این سیستم ها به علت وجود قطب های مختلط شکل نوسانی داریم. این قطب ها به صورت ae^(jb) هستند که اندازه a تا حدی شکل افزایشی و کاهشی سیگنال و پایدار بودن آن را مشخص می‌کند و مقدار e^(jb) نیز به توان n می‌رسد و باعث ایجاد نوسان می‌شود.

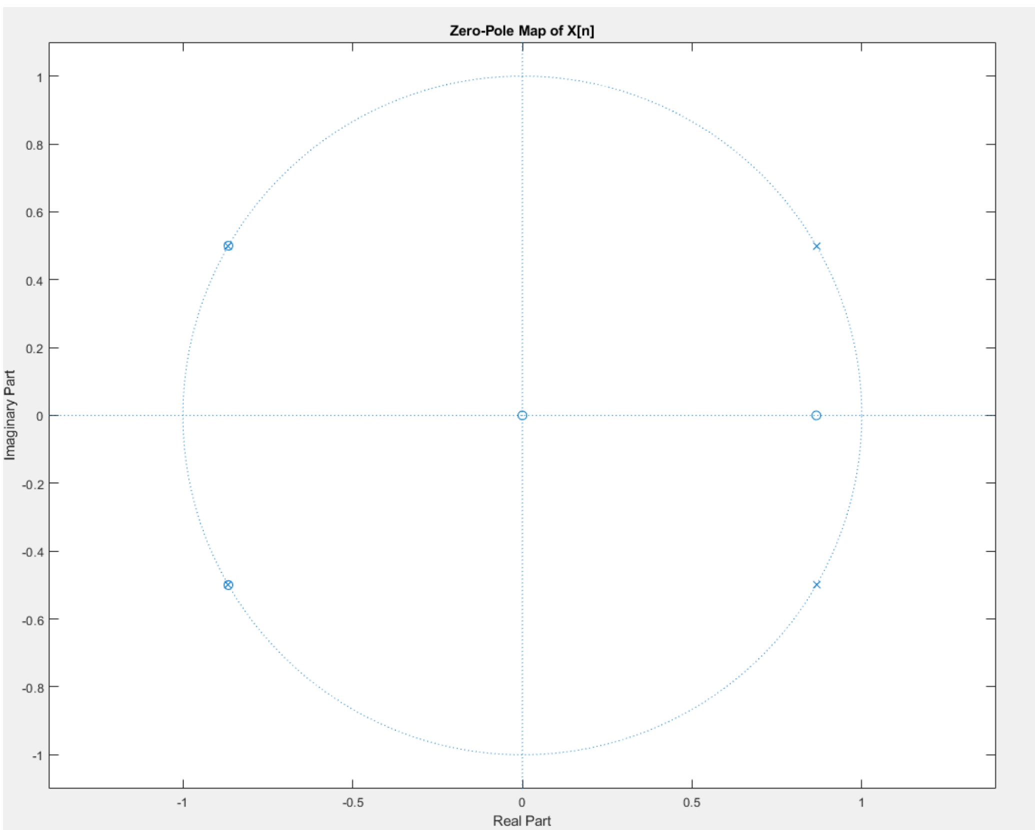
این نکته حائز اهمیت است که متلب نمودار های پاسخ ضربه را با فرض علی بودن سیگنال ها کشیده است و دیتای قبل صفر معنی خاصی ندارد.

در میان این سه سیگنال تنها سیگنال x9 پایدار می‌باشد که دایره واحد در ناحیه همگرایی آن قرار دارد. همچنین با نوشتن رابطه های مطلقا جمع پذیری برای این سه سیگنال این موضوع قابل تایید است.

2.2. Propeties of Z-Transform

بخش 1)

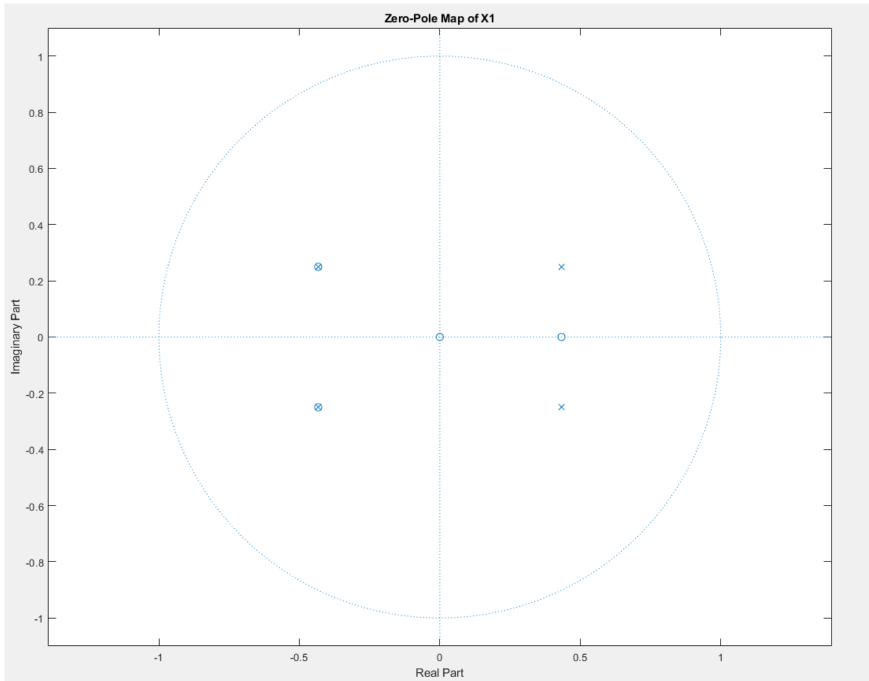
نمودار صفر-قطب سیگنال:



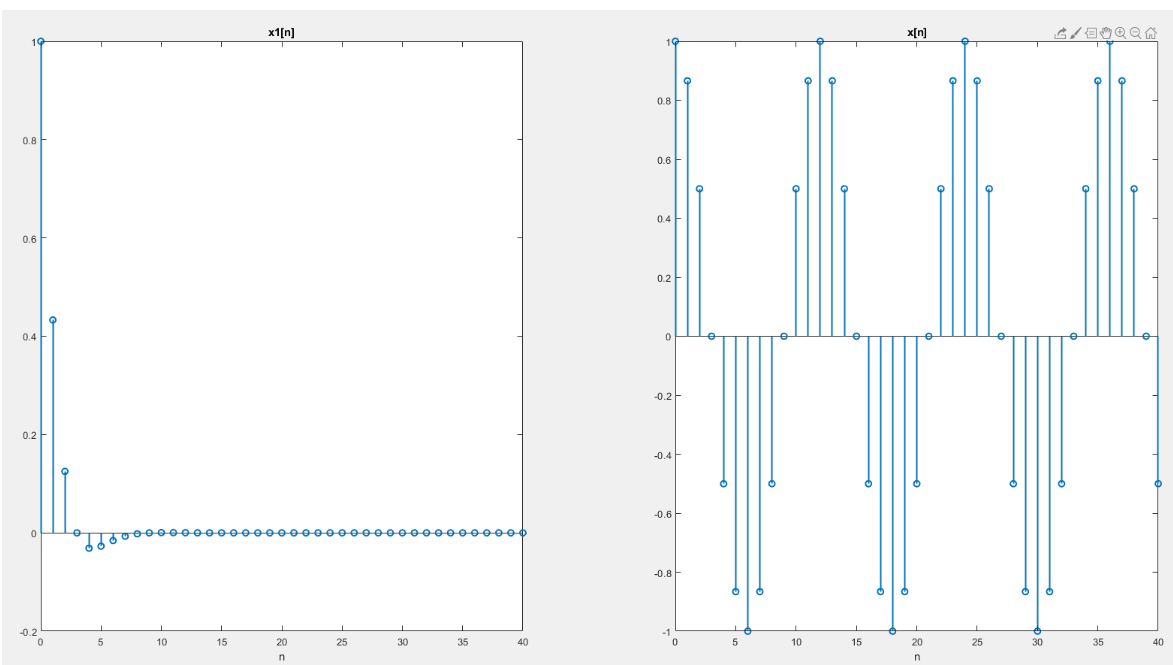
با توجه به اینکه این سیگنال right handed می‌باشد، ROC آن از آخرین قطب تا بینهایت است. با توجه به قطب ها، ROC برابر است با |z|>1

بخش 2)

نمودار صفر-قطب سیگنال:



نمودار زمانی x1 در کنار x:



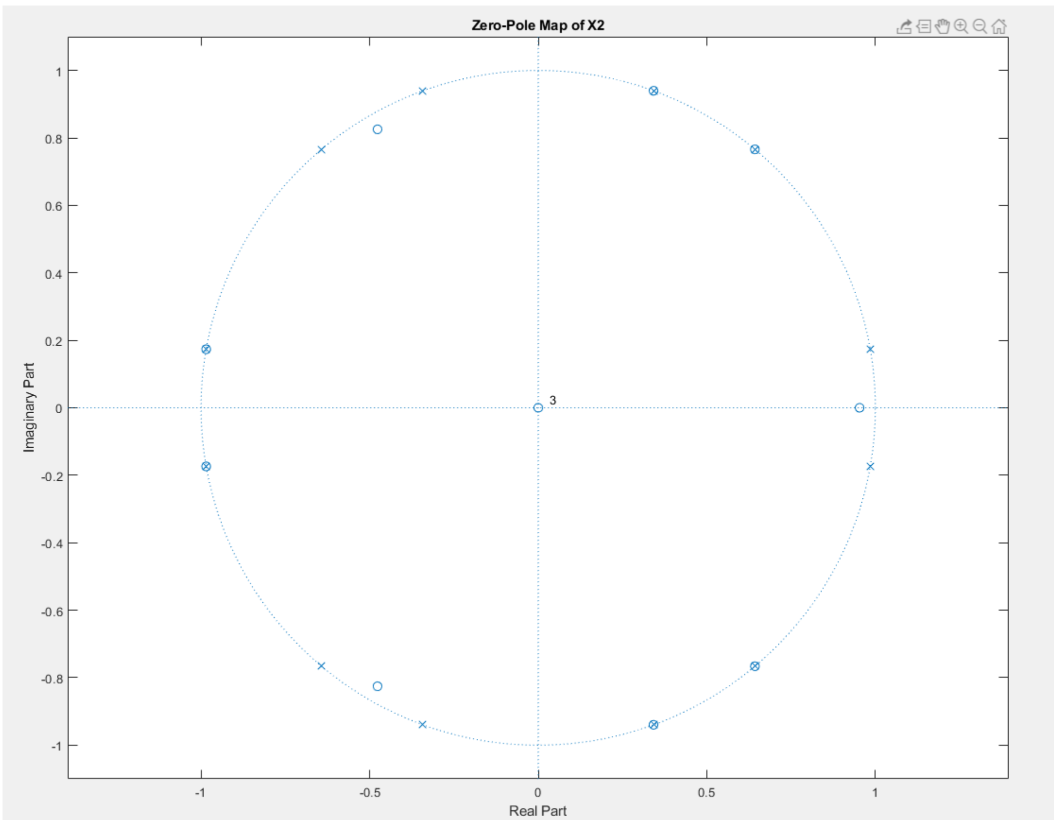
خاصیت Scaling in Z Domain مطابق زیر می‌باشد:

$$x[n] \leftrightarrow x(z) => z_0^n x[n] \leftrightarrow x(\frac{z}{z_0})$$

طبق این خاصیت سیگنال x1[n] باید ضرب شده سیگنال x[n] در $\left(\frac{1}{2}\right)^n$ باشد. که همانطور که در نمودار سمت چپی می‌بینیم، دقیقا همین اتفاق افتاده است. به صورت جزیی در n=0 که اتفاق خاصی نمی‌افتد. در n=1 دقیقا نصف سیگنال سمت راستی است و همینگونه ادامه پیدا می‌کند تا با بزرگ شدن n به صفر میل می‌کند.

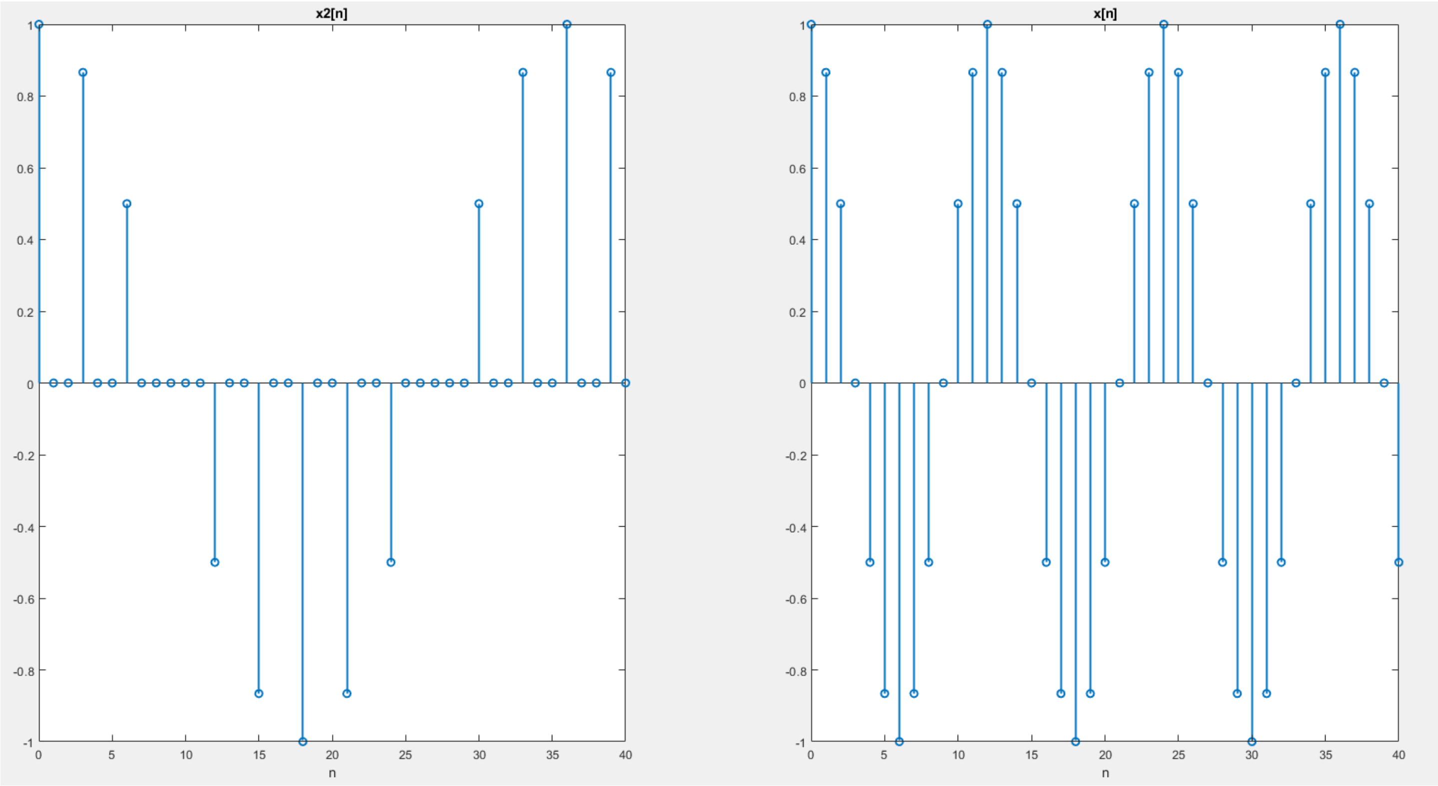
بخش 3)

نمودار صفر-قطب سیگنال:



در مقایسه این نمودار با نمودار صفر-قطب x اولا مشاهده می‌شود انگار هر صفر یا قطب تبدیل به 3 ریشه شده است که این به دلیل قرار گرفتن z^3 به جای z است. که در حقیقت z^3 = a دارای 3 جواب خواهد بود به همین دلیل است که تعداد هر صفر یا قطب 3 برابر می‌شود و در صفحه می‌چرخد. همچنین مشاهده می‌شود که قطب موجود در مبدا نیز مکرر از مرتبه 3 می‌شود. از نظر اندازه هم می‌بینیم که اندازه قطب ها و صفر های روی دایره و مبدا تغییر نمی‌کند اما اندازه صفری که کم از 1 کوچیک تر بوده است کمی بزرگتر شده که این هم تاثیر z^3 را نشان می‌دهد.

نمودار حوزه زمان x2 در کنار x:

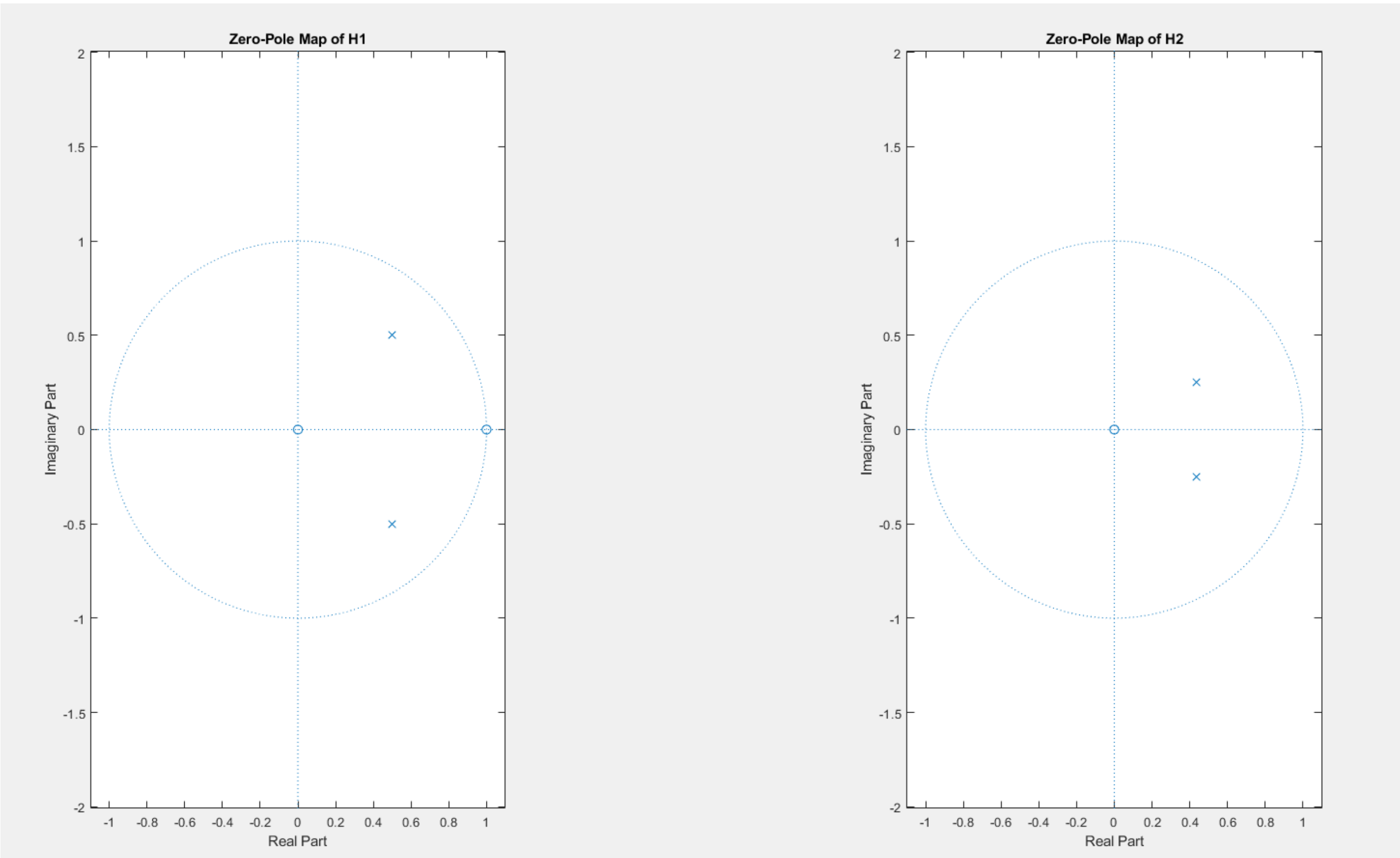


در ویژگی Time Expansion از تبدیل زد، در واقع کاری که می‌کنیم این است که بین نمونه ها تعداد صفر می‌گذاریم. در صورتی که بخواهیم x(z^k) یا در حوزه زمان x[n/k] را به دست بیاوریم باید بین نمونه ها به تعداد k-1 صفر بگذاریم. در نمودار های بالا نیز مشاهده می‌شود که دقیقا همین اتفاق افتاده است و بین نمونه ها سیگنال x[n] به تعداد 3-1=2 تا صفر قرار گرفته و نمودار x2[n] را تشکیل داده است که در واقع برابر است با x[n/3].

2.3. Inverse Z-Transform

بخش 1)

نمودار صفر-قطب دو سیگنال:



در صورت فرض علی بودن سیستم ها، ناحیه همگرایی برابر است با ناحیه ای دایره ای شکل که از بزرگترین قطب تا بینهایت ادامه دارد.

ROC of H1 : $|z| > \frac{\sqrt{2}}{2}$ ROC of H2: $|z| > 0.5$

در رابطه با پایداری سیگنال ها نیز، جفت آنها پایدارند زیرا دایره واحد در ناحیه همگرایی آنان وجود دارد.

بخش 2)

محاسبه دستی سیگنال h1[n]:

$$coefficients \rightarrow r1 = 0.5 + 0.5i \quad , \quad r2 = 0.5r - 0.5i \quad , \quad p1 = 0.5 + 0.5i \quad , \quad p2 = 0.5 - 0.5i$$

$$=> H1(z) = \frac{0.5 + 0.5i}{1 - (0.5 + 0.5i)z^{-1}} + \frac{0.5 - 0.5i}{1 - (0.5 - 0.5i)z^{-1}}$$

$$=> h1[n] = (0.5 + 0.5i)^{n+1}u[n] + (0.5 - 0.5i)^{n+1}u[n]$$

محاسبه دستی سیگنال h2[n]:

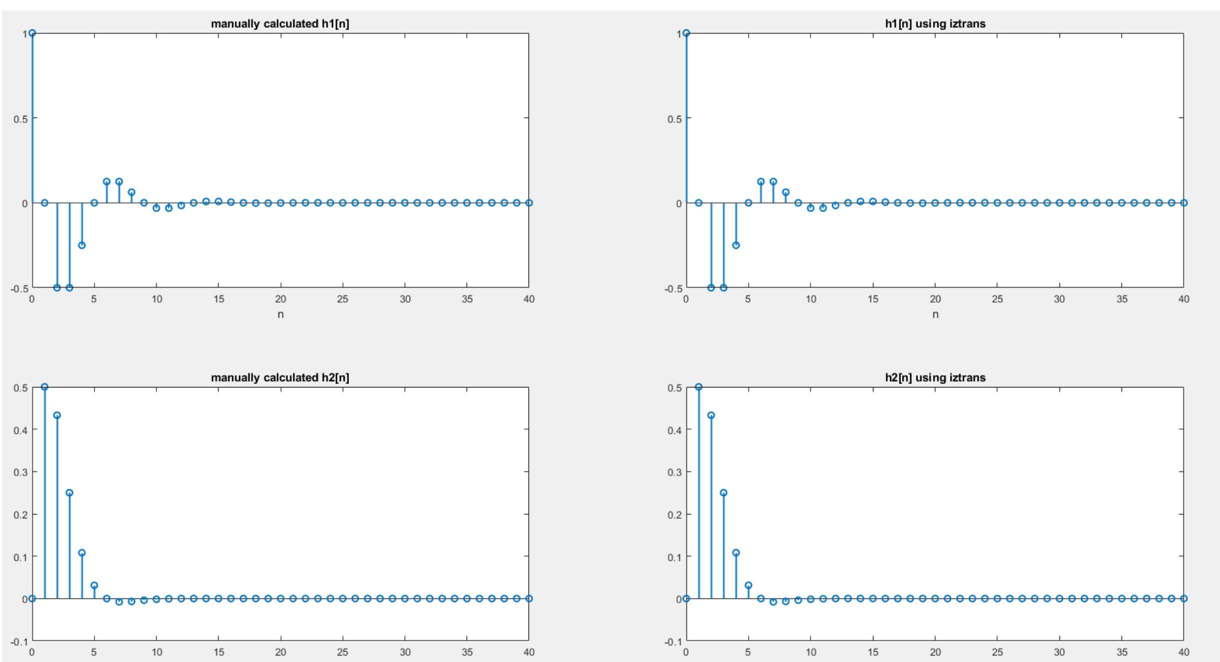
$$coefficients \rightarrow r1 = -i \quad , \quad r2 = i \quad , \quad p1 = \frac{\sqrt{3}}{4} + 0.25i \quad , \quad p2 = \frac{\sqrt{3}}{4} - 0.25i$$

$$=> H2(z) = -\frac{i}{1 - \left(\frac{\sqrt{3}}{4} + 0.25i\right)z^{-1}} + \frac{i}{1 - \left(\frac{\sqrt{3}}{4} - 0.25i\right)z^{-1}}$$

$$=> h2[n] = -i\left(\frac{\sqrt{3}}{4} + 0.25i\right)^{n} u[n] + i\left(\frac{\sqrt{3}}{4} - 0.25i\right)^{n} u[n]$$

بخش 3)

چون فرم خروجی سیمبولیک ()iztrans خیلی پیچیده است از نمودار برای مقایسه استفاده می‌کنیم.



مشاهده می‌شود که سیگنال حساب شده به صورت دستی و سیگنال به دست آمده از iztrans() برابرند.

بخش 4)

وارون تبدیل زد با فرض ضد علی بودن سیستم. (ضراب کسر های جزئی را در قسمت 2 به دست آوردیم.)

محاسبه سیگنال h1[n]:

$$h1[n] = -(0.5+0.5i)^{n+1}u[-n-1] - (0.5-0.5i)^{n+1}u[-n-1]$$

محاسبه سیگنال h2[n]:

$$h2[n] = i\left(\frac{\sqrt{3}}{4}+0.25i\right)^{n}u[-n-1] - i\left(\frac{\sqrt{3}}{4}-0.25i\right)^{n}u[-n-1]$$

با استفاده از تابع iztrans() نمی‌توان تبدیل زد برای سیستم غیر علی را به دست آورد زیرا این تابع به صورت پیش فرض سیستم را علی در نظر می‌گیرد. در واقع چیزی که برای این تابع تعریف شده، سیگنال در n=0,1,2,... می‌باشد که در doc این تابع نیز این مورد را می‌بینیم و برای n<0 دیتای معنی داری ندارد.

2.4. Differential Equation of Systems

بخش 1)

$$y[n] - 0.7y[n-1] + 0.49y[n-2] = 2x[n] - x[n-1]$$

$$=> y(z) - 0.7z^{-1}y(z) + 0.49z^{-2}y(z) = 2x(z) - z^{-1}x(z)$$

$$=> H(z) = \frac{2-z^{-1}}{1-0.7z^{-1}+0.49z^{-2}}$$

با استفاده از تابع residuez() ضرایب را به دست می‌آوریم و تبدیل معکوس را پیدا می‌کنیم.

خروجی تابع residuez():

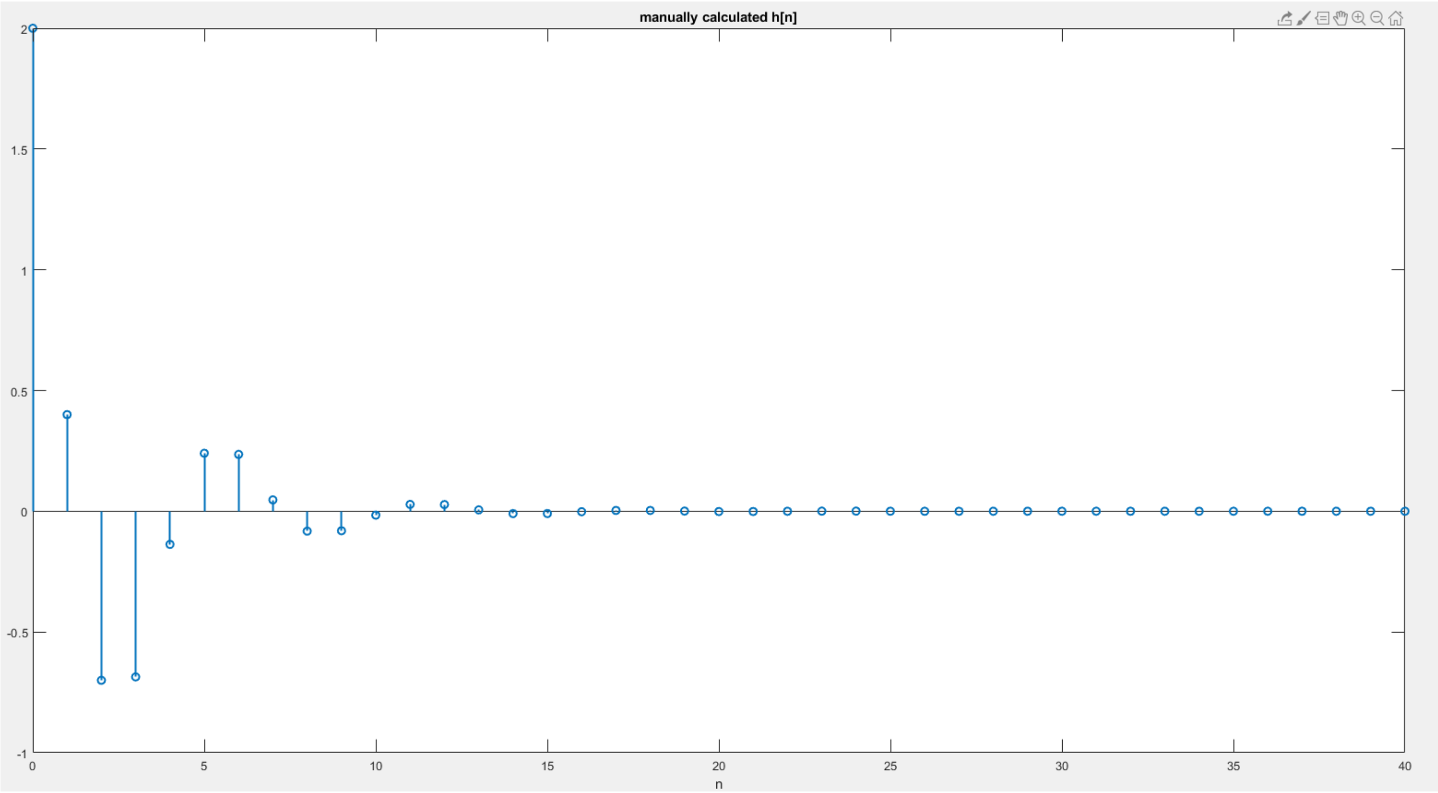
```
r =

   1.0000 + 0.2474i
   1.0000 - 0.2474i


p =

   0.3500 + 0.6062i
   0.3500 - 0.6062i


k =

     []
```

$$r1 = 1 + 0.2474i \quad , \quad r2 = 1 - 0.2474i \quad , \quad p1 = 0.35 + 0.6062i \quad , \quad p2 = 0.35 - 0.6062i$$

$$=> H(z) = \frac{1 + 0.2474i}{1 - (0.35 + 0.6062i)z^{-1}} + \frac{1 - 0.2474i}{1 - (0.35 - 0.6062i)z^{-1}}$$

$$=> h[n] = (1 + 0.2474i)(0.35 + 0.6062i)^n u[n] + (1 - 0.2474i)(0.35 - 0.6062i)^n u[n]$$

رسم سیگنال به دست آمده:



بخش 2)

ابتدا با متلب قطب ها را به دست می‌آوریم:

البته قطب ها از داده های reiduez() بخش قبل در دسترس بودن اما برای تمایز قائل شدن بین بخش ها دوباره با استفاده از pole(tf) محاسبه شد.

```
ans =

   0.3500 + 0.6062i
   0.3500 - 0.6062i
```

برای به دست آوردن مقادیر اولیه می‌توان از قضیه مقدار اولیه یا نمودار به دست آمده از قسمت اول استفاده کرد. اینجا از قضیه استفاده می‌کنیم.

$$h[0] = \lim_{z\to\infty} H(z) = 2$$

$$h[1] = \lim_{z\to\infty} z(H(z) - h[0]) = 0.4$$

حال معادله را شکل می‌دهیم:

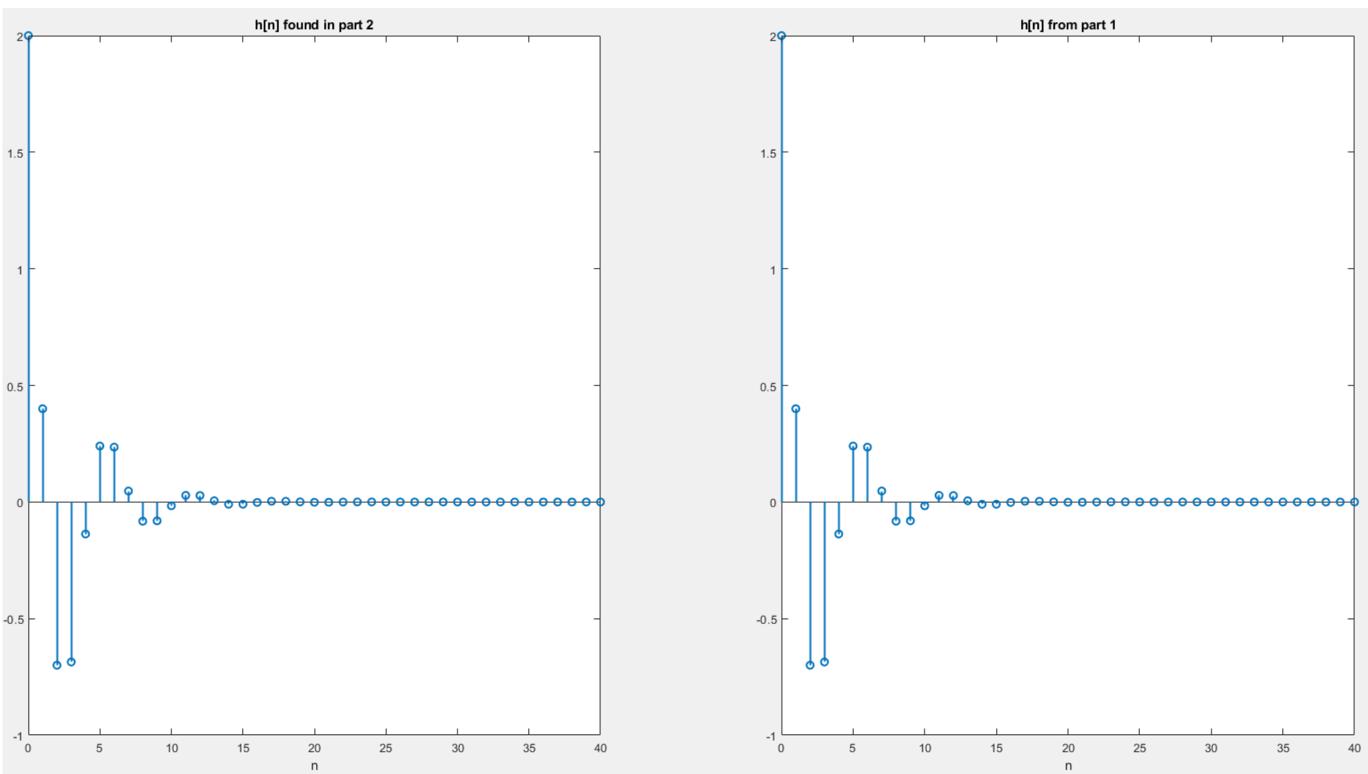
$$(1): h[0] = 2 = a_1 + a_2$$
$$(2): h[1] = 0.4 = a_1(0.35 + 0.6062i) + a_2(0.35 - 0.6062i)$$

معادله را با متلب و به روش ماتریس معکوس حل می‌کنیم:

```
a =

   1.0000 + 0.2474i
   1.0000 - 0.2474i
```

مشاهده می‌شود که این مقادیر دقیقا مطابق مقادیر به دست آمده در بخش 1 هستند.

فرم زمانی سیگنال را با استفاده از iztrans() به دست می‌آوریم و در کنار سیگنال به دست آمده در بخش قبل رسم می‌کنیم و مقایسه می‌کنیم:



مشاهده می‌شود که این دو نمودار با هم برابرند و صحت عملیات این بخش تایید می‌شود.

بخش 3)

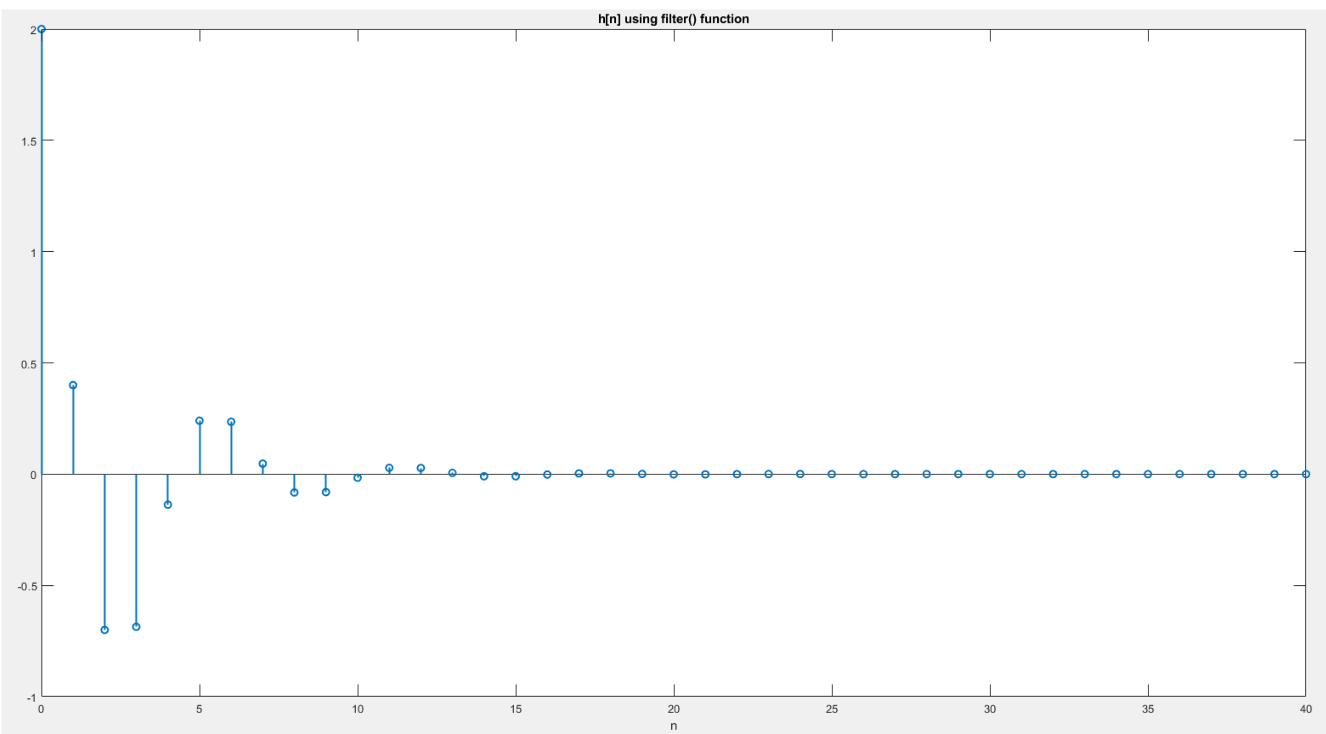
تابع filter(b,a,x) با یک ترنسفر فانکشن، ورودی x را فیلتر می‌کند و به خروجی می‌دهد. این ترنسفر فانکشن به صورت زیر ساخته می‌شود:

$$Y(z) = \frac{b(1) + b(2)z^{-1} + \ldots + b(n_b + 1)z^{-n_b}}{1 + a(2)z^{-1} + \ldots + a(n_a + 1)z^{-n_a}} X(z),$$

در اینجا ترنسفر فانکشن ما همان H(z) می‌باشد که در قسمت های قبلی به دست آمد. نکته مورد توجه این است که این تابع مقادیر a را نسبت به a(1) نرمالایز می‌کند پس اولا a(1) نباید صفر باشد و دوما اگر عددی غیر از یک بود باید در ضرایب b تغییراتی ایجاد کنیم. یعنی b را نیز المان به المان تقسیم بر a(1) کنیم که انگار صورت و مخرج را در یک عدد ضرب کرده ایم که در نتیجه‌ی کار اثری ندارد. ولی اگر این مورد را دقت نکنیم، با ضریبی نتیجه‌ی فیلتر عوض می‌شود.

تا اینجا تکلیف a و b معلوم شد. برای سیگنال ورودی x باید طوری مقادیر را انتخاب کنیم که در صورت عبور از این فیلتر، پاسخ ضربه سیستم را به ما بدهد. برای این کار دقیقا از تعریف ضربه در فضای گسسته استفاده می‌کنیم. یعنی اگر المان اول آرایه x را 1 قرار دهیم و باقی المان ها را صفر قرار دهیم، ضربه در فضای گسسته شبیه سازی می‌شود. همچنین برای دریافت پاسخ به میزان دلخواه باید به همان میزان در x صفر بگذاریم. یعنی اگر انتظار دریافت 40 مقدار از پاسخ ضربه را داشته باشیم باید x یک آرایه چهل تایی باشید که المان اول آن 1 است و بقیه صفر هستند.

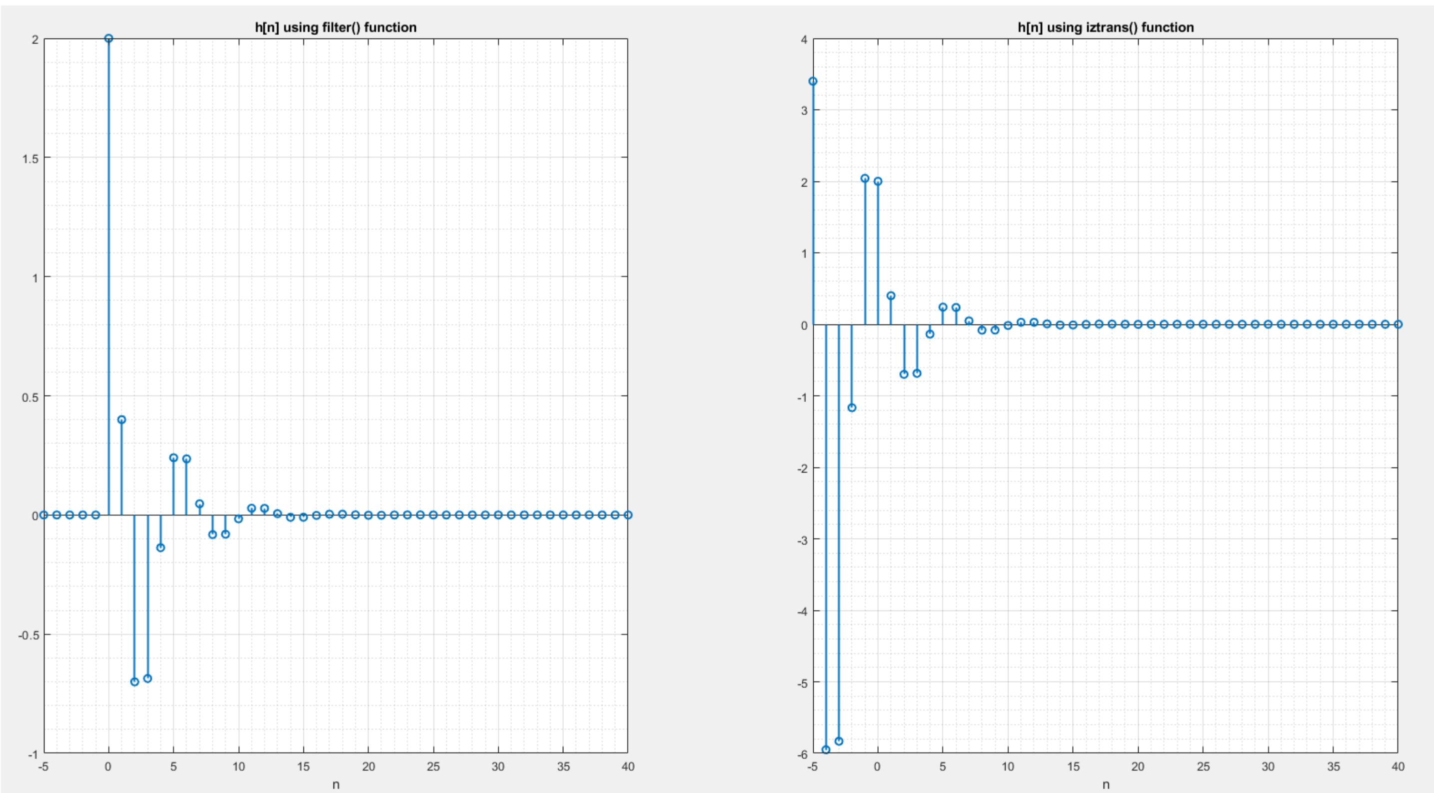
پاسخ ضربه به دست آمده با این روش:



مشاهده می‌شود که نمودار خروجی دقیقا مشابه نمودار های پاسخ ضربه در قسمت های قبل است.

البته تابع فیلتر با تابع iztrans() یک تفاوت مهمی که دارد این است که iztrans() به هر حال برای n<0 نیز مقادیری در نظر می‌گیرد. یعنی با فرض علی بودن سیستم، یکسری ضرایب به دست می‌آورد اما n<0 را برابر با صفر قرار نمی‌دهد و مطابق چیزی که محاسبه شده، مقادیر n<0 را نیز محاسبه می‌کند. اما برای تابع filter() مطابق تعریف x این اتفاق نمی‌افتد. یعنی اگر x اینگونه تعریف شود که در یک نقطه 1 باشد و قبل و بعد آن صفر باشد، خروجی فیلتر اعمال شده بر x برای مقادیر قبل از 1 موجود در x صفر است. یعنی می‌توان گفت این تابع، در صورت تنظیم مناسب x، واقعا پاسخ ضربه یک سیستم علی را محاسبه می‌کند.

تفاوت iztrans و filter برای محاسبه پاسخ ضربه:



همانطور که از نمودار مشخص است اتفاقی که بالاتر به آن اشاره شد در این شکل دیده می‌شود. در این قسمت کد ممکن است یک دوباره کاری دیده شود اما هدف نموداری دیگر برای نمایش تفاوت دو تابع iztrans() و filter بوده است.

بخش 4 (امتیازی*)

مراحل 2 و 3 را مانند بالا را برای این سیگنال انجام می‌دهیم.

بخش 2*)

$$y[n]-y[n-1]-\frac{1}{2}y[n-2]+\frac{1}{2}y[n-3]=2x[n]-x[n-1]$$

$$=> y(z)-z^{-1}y(z)-\frac{1}{2}z^{-2}y(z)+\frac{1}{2}z^{-3}y(z)=2x(z)-z^{-1}x(z)$$

$$=> H(z)=\frac{2-z^{-1}}{1-z^{-1}-\frac{1}{2}z^{-2}+\frac{1}{2}z^{-3}}$$

محاسبه قطب ها با متلب:

```
poles =

   -0.7071
    1.0000
    0.7071
```

محاسبه مقادیر اولیه با قضیه مقدار اولیه:

$$h[0]=\lim_{z\to\infty}H(z)=2$$

$$h[1]=\lim_{z\to\infty}z(H(z)-h[0])=1$$

$$h[2]=\lim_{z\to\infty}z^2(H(z)-h[0]-z^{-1}h[1])=2$$

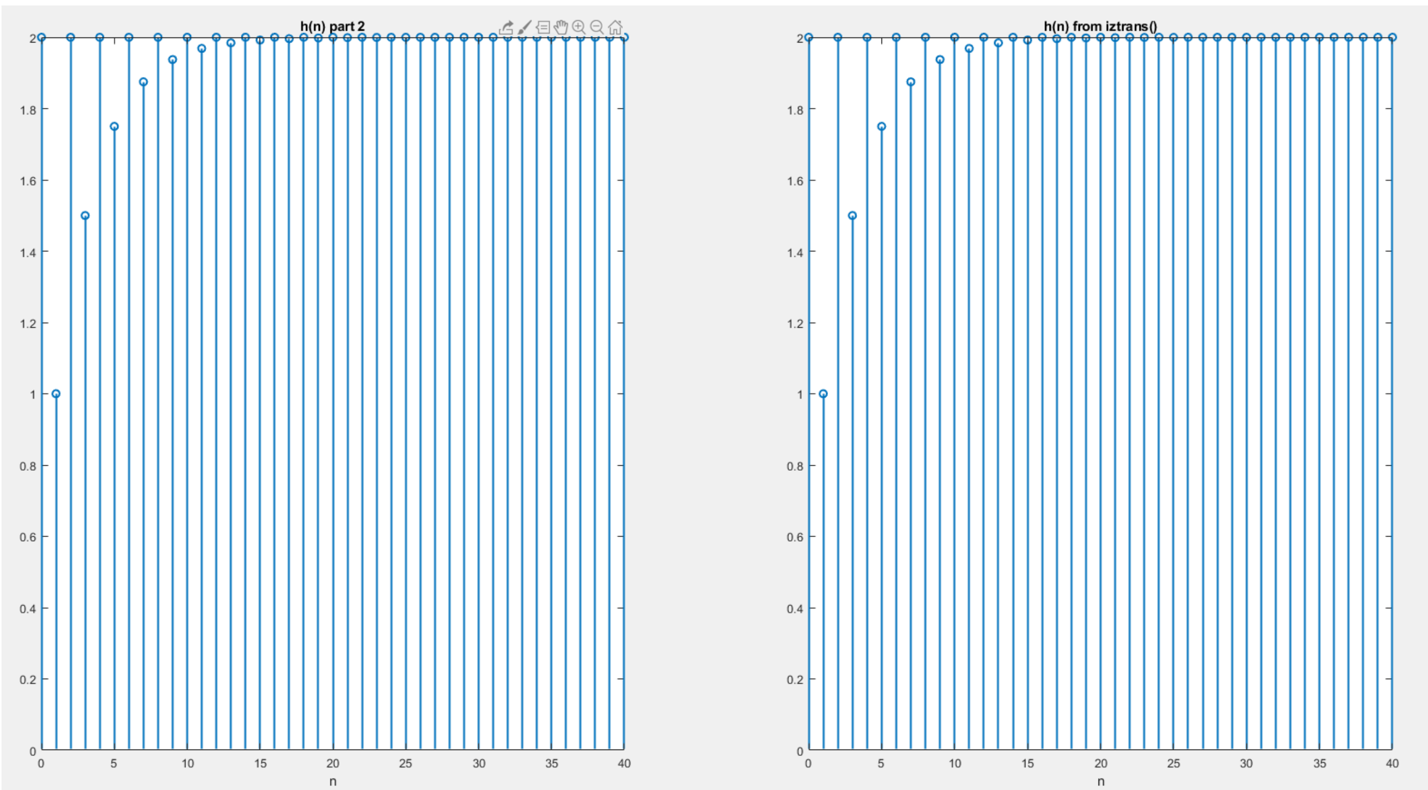
حال معادله را شکل می‌دهیم:

$$(1): h[0]=2=a_1+a_2+a_3$$
$$(2): h[1]=1=a_1(-0.7071)+a_2+a_3(0.7071)$$
$$(3): h[3]=2=a_1(-0.7071)^2+a_2+a_3(0.7071)^2$$

حل معادله با متلب:

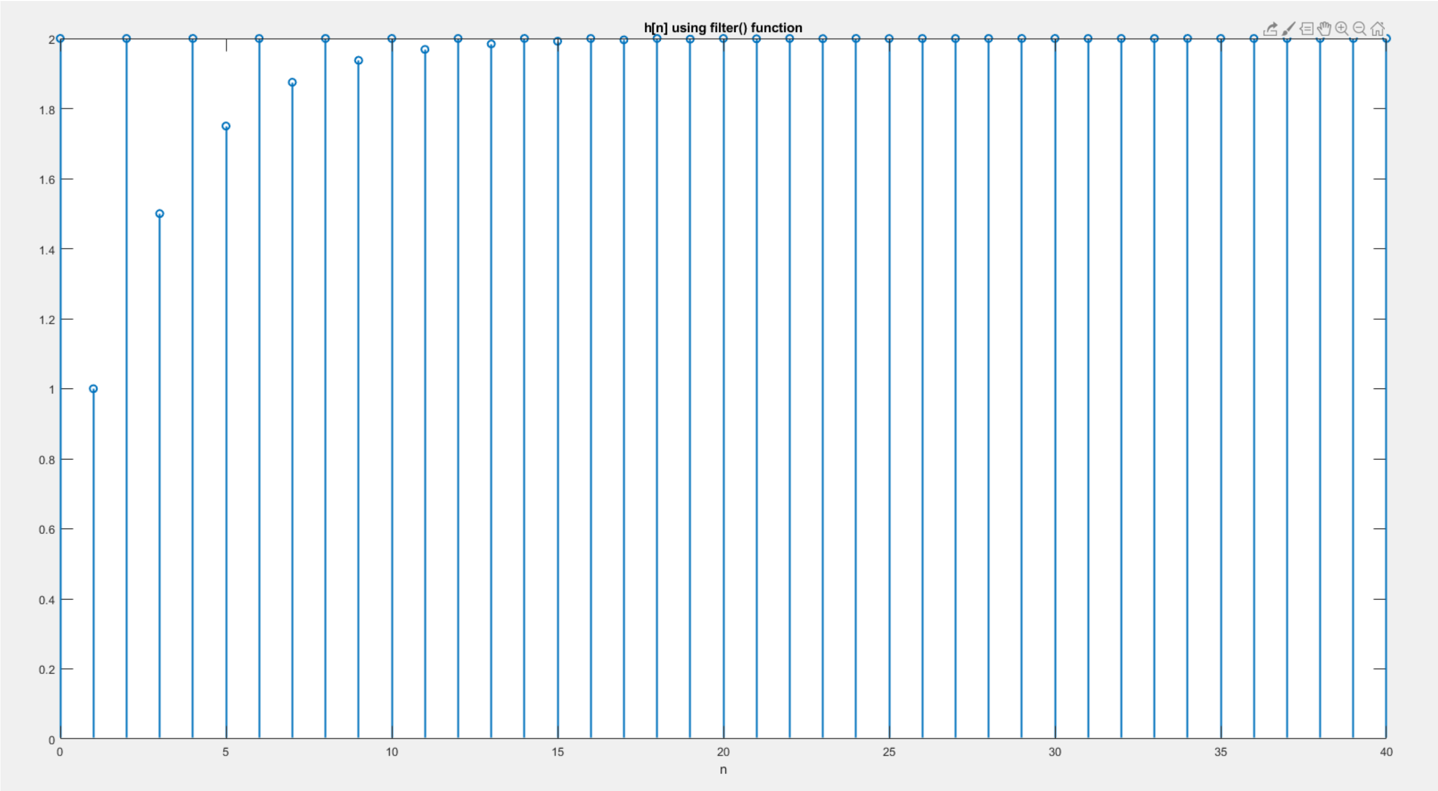
```
a =

    0.7071
    2.0000
   -0.7071
```

حال نمودار حوزه زمان را می‌کشیم و با خروجی iztrans() مقایسه می‌کنیم:



مشاهده می‌شود که نمودار ها عینا برابرند پس صحت عملیات این بخش تایید می‌شود.

بخش 3*)



مشاهده می‌شود که پاسخ ضربه مشابه قسمت های قبل می‌باشد.

Connaître, découvrir, communiquer -- telle est, au fond, notre honorable destinée.

To know, to discover, to publish -- that is, fundamentally, our honorable destiny.

Francois Arago

The End.