

به نام خدا

گزارش تمرین متلب سری ۱

آرشام لولوهری ۹۹۱۰۲۱۵۶

محمد رضا سیدنژاد ۹۹۱۰۱۷۵۱

: ۱

(1.1

کد به صورت زیر است:

```
Editor - F:\Arsham\TERM 4\Signals & Systems\MatlabFiles\HW1_Q1.m
HW1_Q1.m + [ ]
1 %& Q1.1
2 clear;clc;close all
3
4 dimension = 500; %dimension of the page
5 C = zeros(dimension,dimension,3);
6 initial = randi(dimension-1,1,2); %walker first step
7 C(initial(1),initial(2),:) = 255;
8 prev = initial; %walker previous step
9 current = initial; %walker current step
10 total = 100000; %total steps
11 dir = 0; %direction of walker
12 for i=1:total
13     prev = current;
14     dir = randi(8,1);
15     if current(1,1) == 1 %if walker reaches top
16         dir=4;
17     end
18     if current(1,1) == dimension-1 %if walker reaches down
19         dir=3;
20     end
21     if current(1,2) == 1 %if walker reaches left
22         dir=1;
23     end
24     if current(1,2) == dimension-1 %if walker reaches right
25         dir=2;
26     end
27
28     if dir==1 %right
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
```

```
Editor - F:\Arsham\TERM 4\Signals & Systems\MatlabFiles\HW1_Q1.m
HW1_Q1.m + [ ]
29     if dir==1 %right
30         current(1,2)=prev(1,2)+1;
31         C(current(1,1),current(1,2),:) = 255;
32     elseif dir==2 %left
33         current(1,2)=prev(1,2)-1;
34         C(current(1,1),current(1,2),:) = 255;
35     elseif dir==3 %up
36         current(1,1) = prev(1,1)-1;
37         C(current(1,1),current(1,2),:) = 255;
38     elseif dir==4 %down
39         current(1,1)=prev(1,1)+1;
40         C(current(1,1),current(1,2),:) = 255;
41     elseif dir==5 %up right
42         current(1,1)=prev(1,1)-1;
43         current(1,2)=prev(1,2)+1;
44         C(current(1,1),current(1,2),:) = 255;
45     elseif dir==6 %up left
46         current(1,1)=prev(1,1)-1;
47         current(1,2)=prev(1,2)-1;
48         C(current(1,1),current(1,2),:) = 255;
49     elseif dir==7 %down right
50         current(1,1)=prev(1,1)+1;
51         current(1,2)=prev(1,2)+1;
52         C(current(1,1),current(1,2),:) = 255;
53     elseif dir==8 %down left
54         current(1,1)=prev(1,1)+1;
55         current(1,2)=prev(1,2)-1;
56         C(current(1,1),current(1,2),:) = 255;
```

```

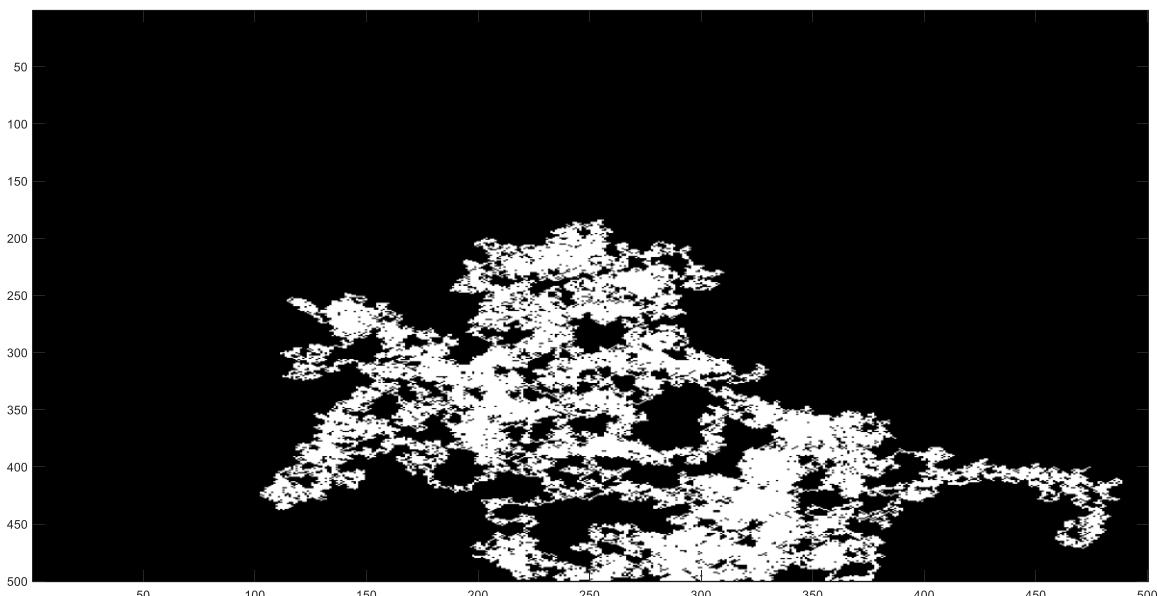
Editor - F:\arsham\TERM 4\Signals & Systems\MatlabFiles\HW1_Q1.m
1 - %HW1_Q1.m
2 - % C(current(1,1),current(1,2),:)=255;
3 - %else dir==1 %left
4 - %    current(1,2)=prev(1,1)-1;
5 - %    C(current(1,1),current(1,2),:)=255;
6 - %elseif dir==2 %up
7 - %    current(1,1) = prev(1,2)-1;
8 - %    C(current(1,1),current(1,2),:)=255;
9 - %elseif dir==3 %up
10 - %    current(1,1) = prev(1,1)-1;
11 - %    C(current(1,1),current(1,2),:)=255;
12 - %elseif dir==4 %down
13 - %    current(1,1)=prev(1,1)+1;
14 - %    C(current(1,1),current(1,2),:)=255;
15 - %elseif dir==5 %right
16 - %    current(1,1)=prev(1,1)-1;
17 - %    current(1,2)=prev(1,2)+1;
18 - %    C(current(1,1),current(1,2),:)=255;
19 - %elseif dir==6 %up left
20 - %    current(1,1)=prev(1,1)-1;
21 - %    current(1,2)=prev(1,2)-1;
22 - %    C(current(1,1),current(1,2),:)=255;
23 - %elseif dir==7 %down right
24 - %    current(1,1)=prev(1,1)+1;
25 - %    current(1,2)=prev(1,2)+1;
26 - %    C(current(1,1),current(1,2),:)=255;
27 - %elseif dir==8 %down left
28 - %    current(1,1)=prev(1,1)+1;
29 - %    current(1,2)=prev(1,2)-1;
30 - %    C(current(1,1),current(1,2),:)=255;
31 - end
32 - imagesc(C);

```

ماتریس C یک ماتریس $3 \times 500 \times 500$ است که دو بُعد اول به ترتیب شماره‌ی سطر و ستون، و بعد سوم رنگ پیکسل مشخص شده را نشان میدهد. ابتدا هر سه لایه‌ی این ماتریس سه بعدی، مقدار صفر دارد و در نتیجه کل صفحه با تصویر شدن این ماتریس، سیاه است.

حال مقدار اولیه‌ای را برای *walker* به صورت زندوم در صفحه انتخاب کرده و به صورت ماتریس یک در دو، داخل *initial* میریزیم. از دو متغیر *prev*, *current* برای مشخص کردن گام قبلی و کنونی *walker* استفاده میکنیم که درایه اول آنها، شماره سطر *walker* و درایه دوم، شماره ستون آن است. تعداد کل گام‌ها در *total* ریخته میشود و حلقه‌ی *for* به آن تعداد اجرا میشود. در هر گام، جهت حرکت به صورت زندوم مشخص شده و داخل *dir* ریخته میشود. جهت با توجه به مقدار *dir* مشخص میشود بطوریکه اعداد ۱ تا ۸ به ترتیب جهت‌های راست، چپ، بالا، پایین، بالا راست، بالا چپ، پایین راست و پایین چپ را نشان میدهد که در هر حالت اقدام لازم انجام میشود. مثلا در *dir==3* ، درایه اول *current* یا گام کنونی *walker* باید یکی کم شود تا یک پیکسل بالا برویم. برای سفید شدن پیکسل هم کافیست هر سه لایه‌ی ماتریس ۲۵۵ شود (هر لایه، یکی از رنگ‌های *rgb* است).

ضمنا در ابتدای حلقه تعیین شده است که اگر `dir` و `current` به گونه ای بودند که گام بعدی `walker` را از کادر خارج میکرد، با دادن مقدار جدید به `dir` ، جهت حرکت بر عکس جهت غیر مجاز میشود تا از کادر خارج نشود(مثلا اگر به مرز سمت چپ صفحه رسیدیم، `dir` با گرفتن مقدار جدید، گام بعدی را به سمت راست قرار میدهد). نمونه ای از نتیجه به صدهزار بار تکرار در زیر آمده است:



(1.2
کد به صورت زیر است:

```

Editor - F:\Parham\TERM 4\Signals & Systems\MatlabFiles\HW1_Q1.m
  60
  61      %% q1.2
  62
  63      clear;clc;close all;
  64
  65      dimension = 250;
  66      P = zeros(dimension,dimension,3);
  67      center = dimension/2;
  68      P(center:center,:)=255;
  69      imagesc(P);
  70      total_walkers = 2000;
  71      initial2 = [0,0];
  72      prev2=[0,0];
  73      current2=[0,0];
  74
  75      for i=1:total_walkers
  76          initial2 = (randi(dimension-6,1,2)){3,3};
  77          prev2=initial2;
  78          current2=initial2;
  79          exit = false;
  80          while exit~=true
  81              prev2 = current2;
  82              dir = randi(8,1);
  83              if current2(1,1)<=3 %if walker reaches top
  84                  dir=4;
  85              elseif current2(1,1)>=(dimension-3) %if walker reaches down
  86                  dir=3;
  87              end
  88              if current2(1,2)<=3 %if walker reaches left
  89
  90
  91
  92
  93
  94
  95
  96
  97
  98
  99
  100
  101
  102
  103
  104
  105
  106
  107
  108
  109
  110
  111
  112
  113
  114
  115
  116
  117
  118
  119
  120
  121
  122
  123
  124

```

```

Editor - F:\Parham\TERM 4\Signals & Systems\MatlabFiles\HW1_Q1.m*
  87
  88      end
  89      if current2(1,2)<=3 %if walker reaches left
  90          dir=1;
  91      elseif current2(1,2)>=(dimension-3) %if walker reaches right
  92          dir=2;
  93      end
  94      if dir==1 %right
  95          current2(1,2)=prev2(1,2)+1;
  96      elseif dir==2 %left
  97          current2(1,2)=prev2(1,2)-1;
  98      elseif dir==3 %up
  99          current2(1,1) = prev2(1,1)-1;
  100     elseif dir==4 %down
  101         current2(1,1)=prev2(1,1)+1;
  102     elseif dir==5 %up right
  103         current2(1,1)=prev2(1,1)-1;
  104         current2(1,2)=prev2(1,2)+1;
  105     elseif dir==6 %up left
  106         current2(1,1)=prev2(1,1)-1;
  107         current2(1,2)=prev2(1,2)-1;
  108     elseif dir==7 %down right
  109         current2(1,1)=prev2(1,1)+1;
  110         current2(1,2)=prev2(1,2)+1;
  111     elseif dir==8 %down left
  112         current2(1,1)=prev2(1,1)+1;
  113         current2(1,2)=prev2(1,2)-1;
  114     end
  115     if ((P(current2(1,1)-1,current2(1,2)-1,1)==255) || (P(current2(1,1)-1,current2(1,2),1)==255))
  116         exit = true;
  117
  118
  119
  120
  121
  122
  123
  124

```

```

Editor - F:\Parham\TERM 4\Signals & Systems\MatlabFiles\HW1_Q1.m*
  96
  97      current2(i,:)=prev2(i,:);
  98      elseif dir==3 %up
  99          current2(1,1) = prev2(1,1)-1;
  100     elseif dir==4 %down
  101         current2(1,1)=prev2(1,1)+1;
  102     elseif dir==5 %up right
  103         current2(1,1)=prev2(1,1)-1;
  104         current2(1,2)=prev2(1,2)+1;
  105     elseif dir==6 %up left
  106         current2(1,1)=prev2(1,1)-1;
  107         current2(1,2)=prev2(1,2)-1;
  108     elseif dir==7 %down right
  109         current2(1,1)=prev2(1,1)+1;
  110         current2(1,2)=prev2(1,2)+1;
  111     elseif dir==8 %down left
  112         current2(1,1)=prev2(1,1)+1;
  113         current2(1,2)=prev2(1,2)-1;
  114     end
  115     if ((P(current2(1,1)-1,current2(1,2)-1,1)==255) || (P(current2(1,1)-1,current2(1,2),1)==255))
  116         exit = true;
  117         P(current2(1,1),current2(1,2),:) = 255;
  118         pause(0.0001);
  119         imagesc(P);
  120
  121
  122
  123
  124

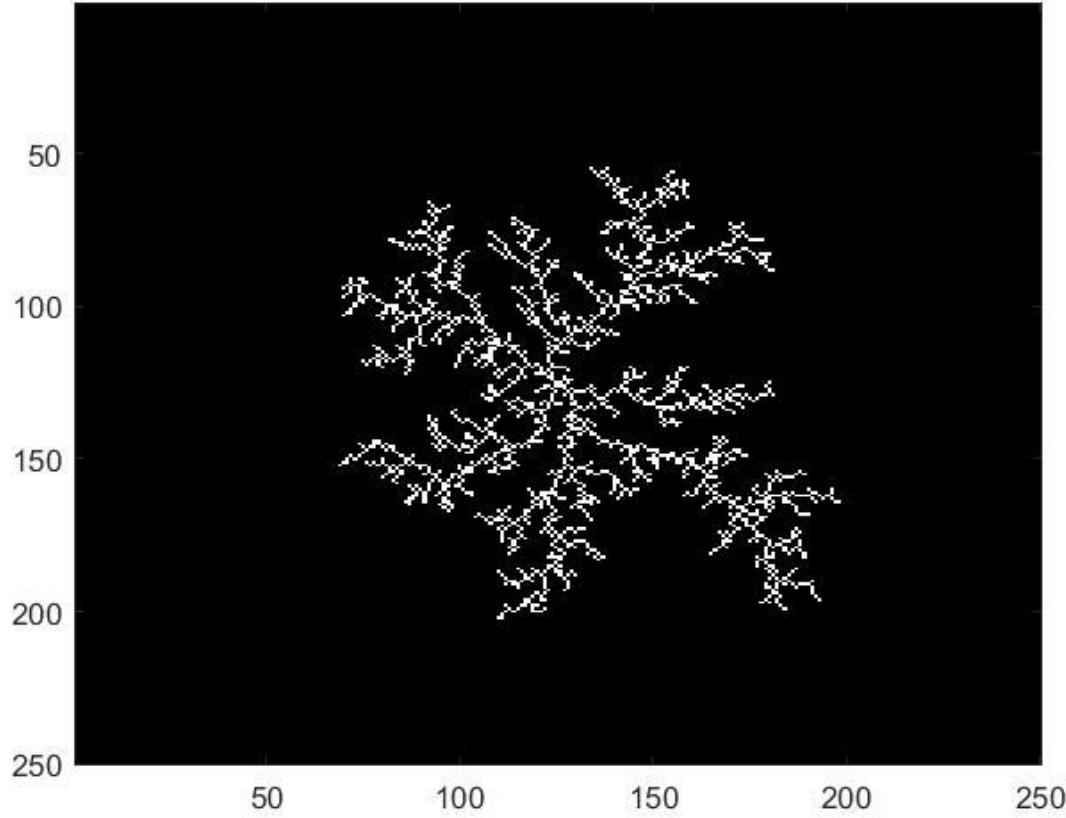
```

ماتریس P همانند ماتریس C در بخش قبل، پیکسل های صفحه و رنگ آنها را مشخص میکند. متغیر های خط ۶۵ تا ۷۳ که نام مشابه بخش قبل

دارند، نقشی کاملا مشابه بخش قبل دارند. `Total_walkers` هم تعداد کل ذراتی است که حرکت رندوم دارند و وقتی به کریستال اصلی میرسند، به آن میپیوندند.

در ابیتا، کریستال فقط یک جزئ در مرکز صفحه دارد که سفید است و مطابق صورت سوال، تنها پایه‌ی خوش‌ی ماست. در هریک از مراحل حلقه‌ی `for`، حرکت یک `walker` انجام می‌شود و تا زمانی که به کریستال نپیوندد، به مرحله بعد نمی‌رویم. در هر مرحله، یک `while` داریم که حرکت `random walk`، داخل آن اتفاق می‌افتد. فرایند این حرکت کاملا مشابه بخش قبل است. جهت حرکت به صورت رندوم انتخاب می‌شود و اگر در نزدیکی مرز بودیم، جهت حرکت در خلاف جهت غیرمجاز قرار می‌گیرد. تنها تفاوت این است که ذرات، تنها زمانی پیکسل خود را سفید می‌کنند که در مجاورت خودشان، یک پیکسل سفید(یک جزئ از کریستال) را داشته باشند و به آن بپیوندند. در این صورت `exit=true` شده و از `while` خارج می‌شویم (شرط خط ۱۱۴ بررسی می‌کند که آیا هریک از ۸ پیکسل همسایه‌ی ذره، سفید هست یا نه).

نمونه‌ای از خروجی به ازای دو هزار `walker` در زیر آمده است، که با ران کردن برنامه، این شکل از صفر و گام به گام ساخته می‌شود:



(1.3

کد به صورت زیر اصلاح شده است:

```
Editor - F:\Arsham\TERM 4\Signals & Systems\Matlab Files\HW1\HW1_Q1.m
HW1_Q1.m x HW1_Q3.m x +
124
125 % Q1.3
126 clear;clc;close all;
127
128 dimension = 250;
129 P = zeros(dimension,dimension,3);
130 center = dimension/2;
131 P(center:center,:)=255;
132
133 h=figure;
134 axis tight manual;
135 filename='crystal.gif';
136
137 imagesc(P);
138 frame = getframe(h);
139 imframe=im(frame);
140 [imind,cm]=rgb2ind(im,256);
141 imwrite(imind,cm,filename,'gif','Loopcount',Inf,'DelayTime',0.001);
142
143 total_walkers = 2000;
144 initial2 = [0,0];
145 prev2=[0,0];
146 current2=[0,0];
147
148 sum = (dimension/2)*(dimension/2+1)-(50*51); %sum of weights,minus numbers 1 to 50
149 w1=[zeros(1,50), (51:dimension/2)/sum]; %weights vector1, with probability 0 for margins
150 w2=flipud(w1); %weights vector2, with probability 0 for margins
151 W=[w1,w2]; %final weights vector
152
153
```

```

Editor - F:\Jansham\TERM 4\Signals & Systems\MatlabFiles\HWI\HWI_Q1.m
151 - W=[w1,w2]; %final weights vector
152
153 - for i=1:total_walkers
154 - initial2 = randsample(dimension,2,true,W).';
155 - prev2=initial2;
156 - current2=initial2;
157 - exit = false;
158 - while exit==true
159 -     prev2 = current2;
160 -     dirW = zeros(1,8);
161 -     if current2(1,1)<dimension/2 & current2(1,2)<dimension/2 %walker in quarter up left
162 -         dirW = [2/13,1/13,1/13,2/13,1/13,1/13,4/13,1/13];
163 -     elseif current2(1,1)<dimension/2 & current2(1,2)>dimension/2 %walker in quarter up right
164 -         dirW = [1/13,2/13,1/13,2/13,1/13,4/13,1/13,1/13];
165 -     elseif current2(1,1)>dimension/2 & current2(1,2)<dimension/2 %walker in quarter down left
166 -         dirW = [2/13,1/13,2/13,1/13,4/13,1/13,1/13,1/13];
167 -     elseif current2(1,1)>dimension/2 & current2(1,2)>dimension/2 %walker in quarter down right
168 -         dirW = [1/13,2/13,1/13,2/13,1/13,4/13,1/13,1/13];
169 -     end
170 -
171 -     dir = randsample(8,1,true,dirW);
172
173 -     if current2(1,1)<=3 %if walker reaches top
174 -         dir=4;
175 -     elseif current2(1,1)>=(dimension-3) %if walker reaches down
176 -         dir=3;
177 -     end
178 -     if current2(1,2)<=3 %if walker reaches left
179 -         dir=1;
180 -     elseif current2(1,2)>=(dimension-3) %if walker reaches right
181 -         dir=2;
182 -     end
183 -     if dir==1 %right
184 -         current2(1,2)=prev2(1,2)+1;
185 -     elseif dir==2 %left
186 -         current2(1,2)=prev2(1,2)-1;
187 -     elseif dir==3 %up
188 -         current2(1,1) = prev2(1,1)-1;
189 -     elseif dir==4 %down
190 -         current2(1,1)=prev2(1,1)+1;
191 -     elseif dir==5 %up right
192 -         current2(1,1)=prev2(1,1)-1;
193 -         current2(1,2)=prev2(1,2)+1;
194 -     elseif dir==6 %up left
195 -         current2(1,1)=prev2(1,1)-1;
196 -         current2(1,2)=prev2(1,2)-1;
197 -     elseif dir==7 %down right
198 -         current2(1,1)=prev2(1,1)+1;
199 -         current2(1,2)=prev2(1,2)+1;
200 -     elseif dir==8 %down left
201 -         current2(1,1)=prev2(1,1)+1;
202 -         current2(1,2)=prev2(1,2)-1;
203 -     end
204 -     if ((P(current2(1,1)-1,current2(1,2)-1,1)==255) || (P(current2(1,1)-1,current2(1,2),1)==255) || (P(current2(1,1),current2(1,2),-1)==255))
205 -         exit = true;
206 -     end
207
208
209
210
211
212
213
214
215
216
217
218
219
220

```

```

Editor - F:\Jansham\TERM 4\Signals & Systems\MatlabFiles\HWI\HWI_Q2.m
178 - if current2(1,2)<=3 %if walker reaches left
179 -     dir=1;
180 - elseif current2(1,2)>=(dimension-3) %if walker reaches right
181 -     dir=2;
182 - end
183 - if dir==1 %right
184 -     current2(1,2)=prev2(1,2)+1;
185 - elseif dir==2 %left
186 -     current2(1,2)=prev2(1,2)-1;
187 - elseif dir==3 %up
188 -     current2(1,1) = prev2(1,1)-1;
189 - elseif dir==4 %down
190 -     current2(1,1)=prev2(1,1)+1;
191 - elseif dir==5 %up right
192 -     current2(1,1)=prev2(1,1)-1;
193 -     current2(1,2)=prev2(1,2)+1;
194 - elseif dir==6 %up left
195 -     current2(1,1)=prev2(1,1)-1;
196 -     current2(1,2)=prev2(1,2)-1;
197 - elseif dir==7 %down right
198 -     current2(1,1)=prev2(1,1)+1;
199 -     current2(1,2)=prev2(1,2)+1;
200 - elseif dir==8 %down left
201 -     current2(1,1)=prev2(1,1)+1;
202 -     current2(1,2)=prev2(1,2)-1;
203 - end
204 - if ((P(current2(1,1)-1,current2(1,2)-1,1)==255) || (P(current2(1,1)-1,current2(1,2),1)==255) || (P(current2(1,1),current2(1,2),-1)==255))
205 -     exit = true;
206 - end
207
208
209
210
211
212
213
214
215
216
217
218
219
220

```

```

Editor - F:\Jansham\TERM 4\Signals & Systems\MatlabFiles\HWI\HWI_Q3.m
193 - current2(1,2)=prev2(1,2)+1;
194 - elseif dir==1 %up left
195 -     current2(1,1)=prev2(1,1)-1;
196 -     current2(1,2)=prev2(1,2)-1;
197 - elseif dir==7 %down right
198 -     current2(1,1)=prev2(1,1)+1;
199 -     current2(1,2)=prev2(1,2)+1;
200 - elseif dir==8 %down left
201 -     current2(1,1)=prev2(1,1)+1;
202 -     current2(1,2)=prev2(1,2)-1;
203 - end
204 - if ((P(current2(1,1)-1,current2(1,2)-1,1)==255) || (P(current2(1,1)-1,current2(1,2),1)==255) || (P(current2(1,1),current2(1,2),-1)==255))
205 -     exit = true;
206 - P(current2(1,1),current2(1,2),:)=255;
207 - pause(0.0001);
208 -
209 - imagesc(P);
210 - drawnow;
211 - frame = getframe(h);
212 - im=frame2im(frame);
213 - [imind,cm]=rgb2ind(im,256);
214 - imwrite(imind,cm,filename,'gif','WriteMode','append','DelayTime',0.001);
215 - end
216 - end
217 - end
218
219
220

```

اولین کاری که برای افزایش سرعت رشد کریستال انجام شده، این است که احتمال اینکه نقطه شروع walker ها در اطراف باشد را کاهش داده ایم. در واقع بجای اینکه انتخاب نقطه شروع کاملاً رندوم باشد، با استفاده

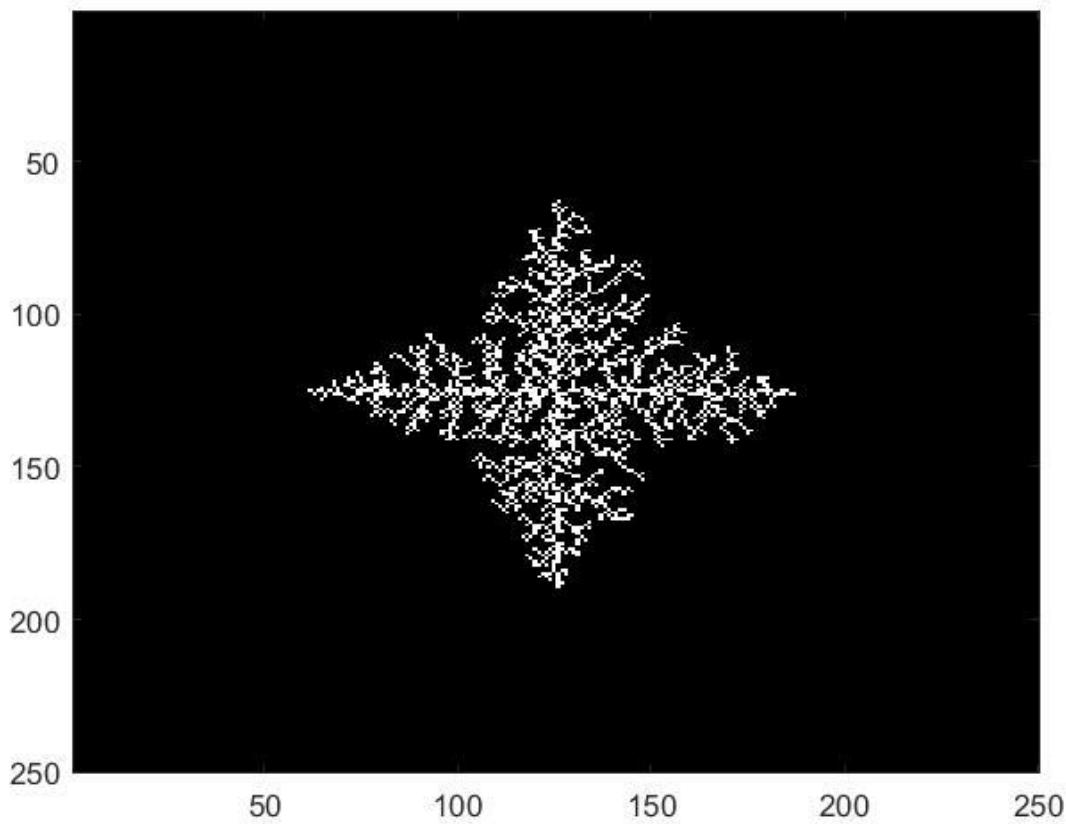
از `randsample` ، به پیکسل هایی که نزدیک وسط صفحه هستند، احتمال انتخاب بیشتری داده ایم. بردار احتمال انتخاب هر عدد که به `w1,w2` داده میشود، `W` است که خود از دو بردار `w1,w2` تشکیل شده که قرینه‌ی یکدیگرند. تعریف این دو بردار بدین شکل است که برای ۵۰ عدد اول و آخر، احتمال صفر را بدھیم. یعنی نقطه شروع هیچ ذره‌ای در فاصله‌ی کمتر از ۵۰ پیکسلی حاشیه‌ها نیست. از طرف دیگر هرچه عدد ما به ۱۲۵ یا همان وسط صفحه نزدیک میشود، وزن عدد و در نتیجه احتمال انتخاب آن بیشتر میشود. پس در ادامه‌ی `w1` ، از عدد ۵۱ تا ۱۲۵ ، هر کدام وزنی برابر با عدد خود دارند. بدین ترتیب هرچه از ۵۱ به ۱۲۵ نزدیک شویم، احتمال انتخاب پیکسل افزایش می‌یابد. حال احتمال انتخاب هر عدد، برابر با حاصل تقسیم وزن آن، به مجموع وزن تمام اعداد است. پس هر وزنی در `w1` باید بر مجموع وزن‌ها یا `sum` تقسیم شود. محاسبه `sum` نیز بر این اساس است که از ۵۱ تا ۱۲۵ ، هر عدد وزنی برابر با خود را دارد. پس باید مجموع اعداد ۱ تا ۵ را از مجموع اعداد ۱ تا ۱۲۵ کم کنیم. چون همین وزن‌ها برای اعداد ۱۲۶ تا ۲۰۰ نیز وجود دارند، کل مجموع در ۲ ضرب میشود.

همین کار را باید به صورت متقارن برای ۱۲۶ تا ۲۵۰ انجام دهیم تا با افزایش عدد، احتمال انتخاب کمتر شود. پس `w2` را `fliplr(w1)` قرار میدهیم. با متصل کردن این دو بردار به هم، بردار احتمال `W` تشکیل میشود. چون نقاط شروع به مرکز نزدیک‌تر است، سریعتر به کریستال میپیوندند.

اقدام دوم، مرکزگرا کردن جهت حرکت ذرات است. در واقع صفحه نمایش را ۴ بخش در نظر میگیریم: بالا چپ، بالا راست، پایین چپ و پایین راست. ذره‌ما در هر بخش که باشد، `dir` را برای گام بعدی به گونه‌ای انتخاب میکنیم که احتمال حرکت به سمت مرکز بیشتر باشد. این

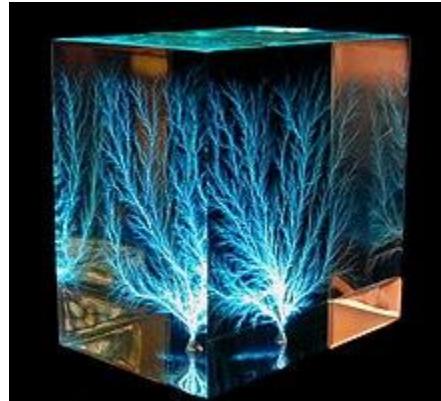
کار را نیز با `randsample` و بردار احتمال `dirW` انجام میدهیم. برای مثال در خط ۱۶۱ در تصویر بالا، شرط گذاشته ایم که اگر مکان کنونی ذره در بخش بالاچپ صفحه بود، احتمال حرکت ذره به سمت پایین راست (`dir=7`)، از همه بیشتر باشد. اولویت بعدی، حرکت به سمت پایین و یا راست (به ترتیب `dir=4` و `dir=1`) است که احتمالشان نصف `dir=7` است. بقیه جهات هم در اولویت آخر و با حداقل احتمال (نصف پایین و راست) هستند. همین کار را برای هر ۴ جهت اعمال میکنیم و پس از مشخص شدن بردار احتمال `dirW`، با استفاده از آن جهت را مشخص میکنیم. البته انجام اقدام دوم برای افزایش سرعت، بدلیل محدودیت زیادی که برای حرکت ذرات ایجاد میکند، اندکی شکل کریستال را تغییر میدهد و فرم آن کمی منظم تر میشود. اما سرعت بطور قابل ملاحظه ای زیاد میشود و به ازای همان دو هزار ذره، سرعت تشکیل کامل کریستال تقریباً دوبرابر میشود.

تصویر نهایی در زیر آمده است. توسط کد های موجود در خطوط حدود ۱۳۳ و نیز ۲۰۹، فریم به فریم این تصاویر ذخیره شده و در فایل `crystal.gif` ذخیره شده است:



(1.4

این نقش ها در اثر تخلیه الکتریکی ، و عمدتا در اثر رعد و برق میتوانند در سطوح مختلف و حتی بدن انسان ایجاد شوند. عامل ایجاد این نقش ها، حرکت الکترون ها و مسیر آنها در اثر تخلیه الکتریکی است. نمونه ای از یک درخت الکتریکی به صورت تصویر زیر است:



تصویر کد به شکل زیر است:

```

Editor : F:\arsham\TERM 4\Signals & Systems\MatlabFiles\HW1\HW1_Q1.m*
220
221
222
223 % Q1.4
224 clear;clc;close all;
225
226 dimension = 250;
227 P = zeros(dimension,dimension,3);
228 center = dimension/2;
229 P(dimension-3:center,center,:)=255;
230 imagesc(P);
231 total_walkers = 5000;
232 initial2 = [0,0];
233 prev2=[0,0];
234 current2=[0,0];
235
236 %horizontal distribution
237 sum_h = (dimension/2)*(dimension/2+1)-2*(3+2*1); %sum of weights,minus 1,2,3
238 wh1=zeros(1,50); %weights vector1, with probability 0 for margins
239 wh2=randperm(wh1); %weights vector2, with probability 0 for margins
240 W_H=[wh1,wh2]; %final weights vector, for horizontal position
241
242 %vertical distribution
243 sum_v = (dimension)*(dimension+1)/2-(1+2+3+248+249+250);
244 W_V=zeros(1,3), (4:(dimension-3))/sum_v, zeros(1,3));
245
246 for i=1:total_walkers
247     initial2(i,1) = randsample(dimension,1,true,W_V);
248     initial2(i,2) = randsample(dimension,1,true,W_H);
249     prev2=initial2;
250     current2=initial2;
251     exit = false;
252     while exit==true
253         prev2 = current2;
254         dir = randi(8,1);
255         if current2(1,1)<3 %if walker reaches top
256             dir=4;
257         elseif current2(1,1)>=(dimension-3) %if walker reaches down
258             dir=3;
259         end
260         if current2(1,2)<3 %if walker reaches left
261             dir=1;
262         elseif current2(1,2)>=(dimension-3) %if walker reaches right
263             dir=2;
264         end
265         if dir==1 %right
266             current2(1,2)=prev2(1,2)+1;
267         elseif dir==2 %left
268             current2(1,2)=prev2(1,2)-1;
269         elseif dir==3 %up
270             current2(1,1) = prev2(1,1)-1;
271         elseif dir==4 %down
272             current2(1,1)=prev2(1,1)+1;
273         elseif dir==5 %up right
274             current2(1,1)=prev2(1,1)-1;
275             current2(1,2)=prev2(1,2)+1;

```

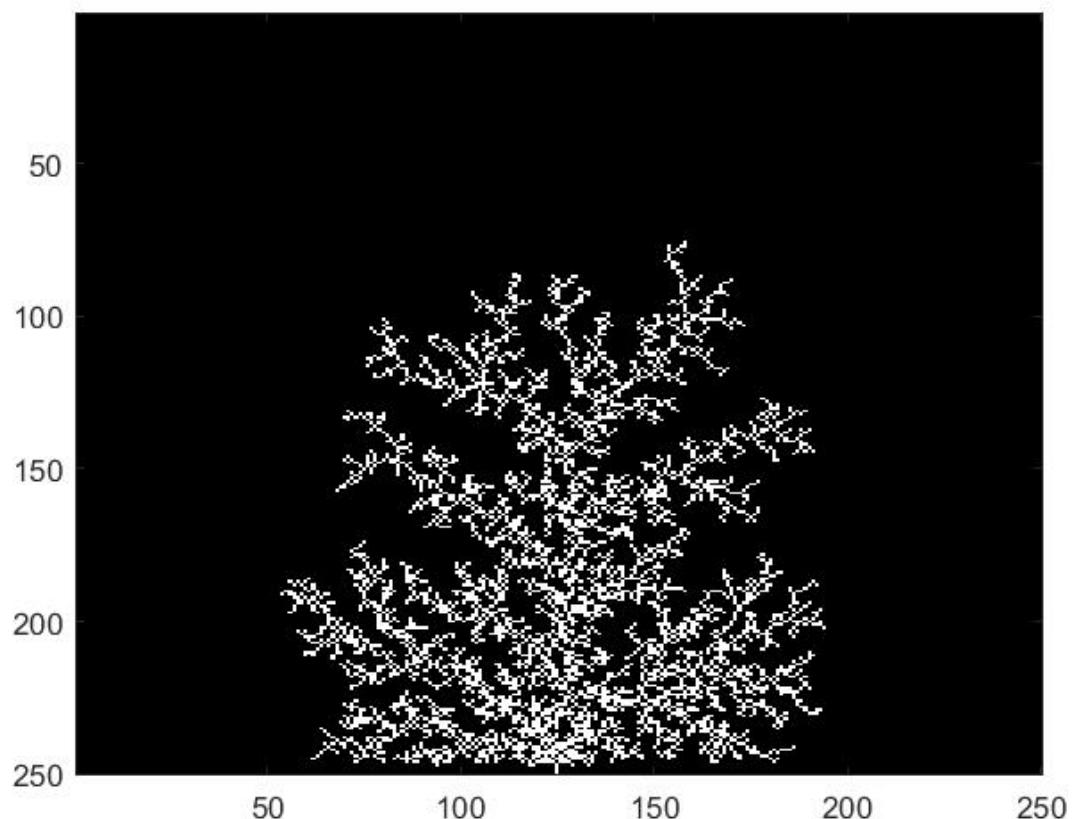
```

Editor - F:\arsham\TERM 4\Signals & Systems\Matlab files\HW1\HW1_Q1.m*
265 - if dir==1 %right
266 -     current2(1,2)=prev2(1,2)+1;
267 - elseif dir==2 %left
268 -     current2(1,2)=prev2(1,2)-1;
269 - elseif dir==3 %up
270 -     current2(1,1) = prev2(1,1)-1;
271 - elseif dir==4 %down
272 -     current2(1,1)=prev2(1,1)+1;
273 - elseif dir==5 %up right
274 -     current2(1,1)=prev2(1,1)-1;
275 -     current2(1,2)=prev2(1,2)+1;
276 - elseif dir==6 %up left
277 -     current2(1,1)=prev2(1,1)-1;
278 -     current2(1,2)=prev2(1,2)-1;
279 - elseif dir==7 %down right
280 -     current2(1,1)=prev2(1,1)+1;
281 -     current2(1,2)=prev2(1,2)+1;
282 - elseif dir==8 %down left
283 -     current2(1,1)=prev2(1,1)+1;
284 -     current2(1,2)=prev2(1,2)-1;
285 - end
286 - if ((P(current2(1,1)-1,current2(1,2)-1,1)==255) || (P(current2(1,1)-1,current2(1,2),1)==2
287 - exit = true;
288 - P(current2(1,1),current2(1,2),:)=255;
289 - pause(0.0001);
290 - imagesc(P);
291 - end
292 - end
293 - end

```

برای تولید چنین شکلی، واضح است که اول از همه، پایه‌ی خوش‌هی ما باید در پایین صفحه باشد. در گام بعدی، مشابه بخش ۱.۳، احتمال نقطه شروع ذرات را مشخص می‌کنیم. تعیین ستون (حدوده افقی) نقطه شروع، تقریباً مانند بخش قبل است. ۳ پیکسل ابتدا و انتها هرگز نقطه شروع نیستند و هرچه از دو طرف به مرکز نزدیک می‌شویم، وزن و احتمال انتخاب اعداد، بیشتر می‌شود. پس تشکیل W_{H} , W_{L} , W_{V} تا حد زیادی شبیه بخش قبل است. برای تعیین سطر (موقعیت عمودی) نقطه شروع، با توجه به اینکه پایه‌ی ما اینبار در پایین صفحه است، توزیع احتمال را بدین شکل تعیین کرده ایم که هرچه به پیکسل‌های پایینتر می‌آییم، احتمال انتخاب پیکسل بیشتر است. پس وزن هر عدد برابر با خود آن عدد قرار داده شده است. بدین ترتیب بردار احتمال W_{V} ، احتمال ذرات را از ۴ تا ۲۵۷، به صورت تقسیم عدد بر مجموع وزن‌ها قرار داده است. مجموع وزن‌ها یا \sum_{V} هم برابر با حاصل جمع اعداد ۴ تا ۲۵۷ است. این احتمالات به `randsample` داده می‌شود و نقطه شروع ذرات مشخص می‌شود.

بدین ترتیب به ازای پنج هزار ذره، تصویر درخت حاصله به شکل زیر است:



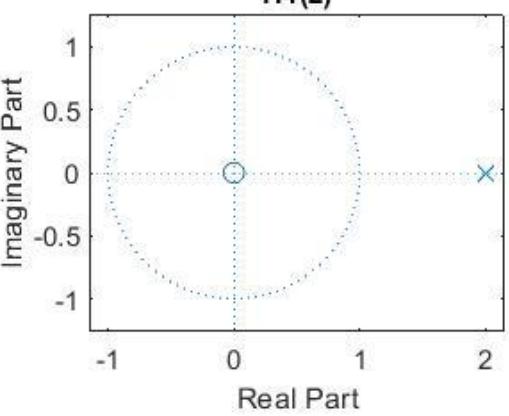
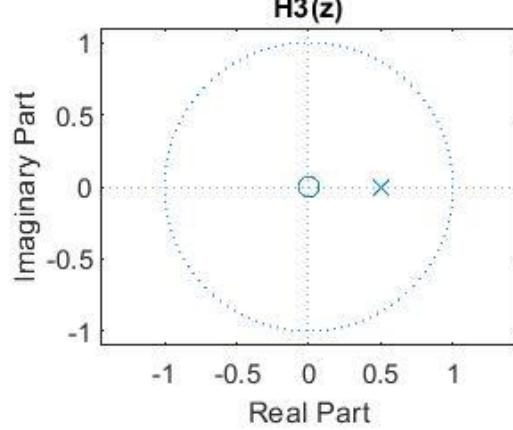
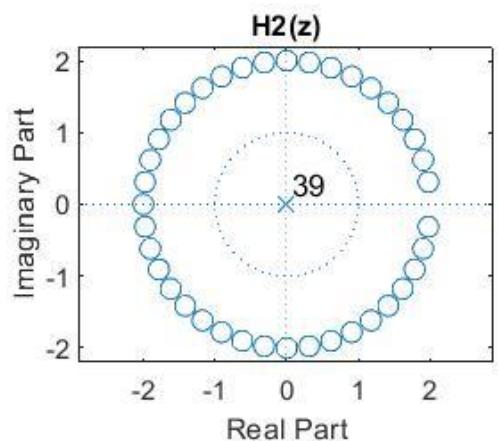
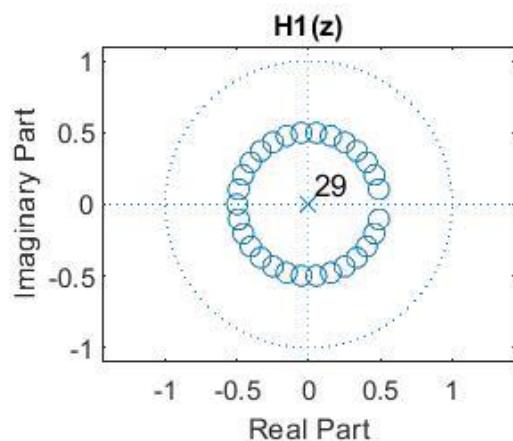
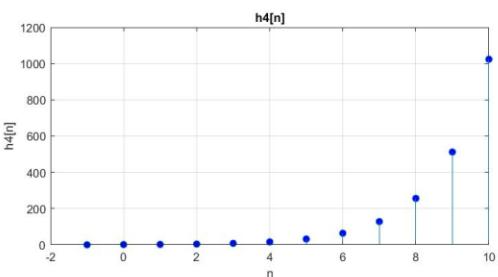
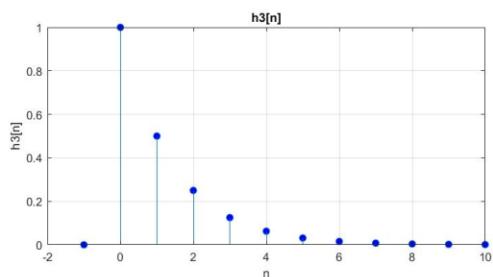
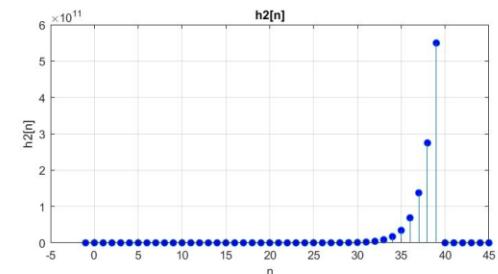
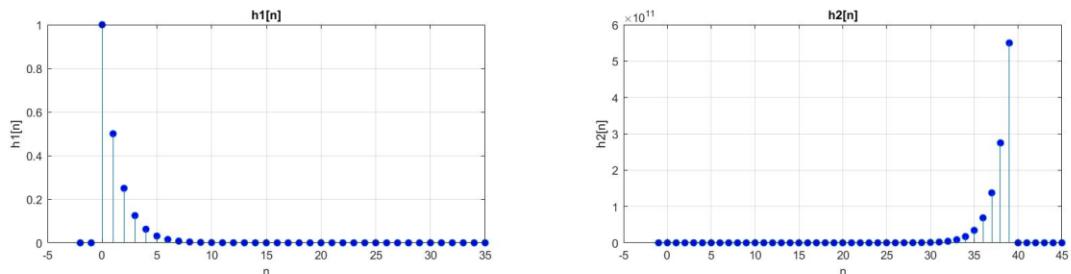
گزارش سوال ۲

در ابتدا یک ماتریس اولیه به عنوان صفحه اولیه در نظر گرفته شده که هر عضو آن یک عدد رندم بین ۰ تا ۱ است. سپس با دستور خط پنج تعیین کردیم که آن دسته از خانه هایی که مقدارشان از نیم بیشتر است مقداری برابر با یک داشته باشند و بقیه مقداری برابر با ۰ به عبارتی حال خانه ها به احتمال ۵۰ درصد ۰ و به احتمال ۵۰ درصد یک هستند. یعنی حدوداً نصف خانه ها مرده و نصفی زنده اند که این مسئله بنا به خواسته سوال کاملاً تصادفی است. سپس مقدار دلیلی بین تصاویر gif و تعداد تصاویر و نام فایلی که در آن ذخیره سازی صورت می گیرد داده شده است. همه اطلاعات به تابع designGif داده می شود. این تابع step بار فرآیند را ادامه می دهد و هر مرحله را به فایل gif اضافه می کند به این صورت که مرحله اول که مرحله تشکیل فایل است با یک if else , if شناسایی می شود در این مرحله فایل ایجاد می شود و در مراحل بعد تکه های بعدی تصویر به فایل ایجاد شده متصل می شوند. هر مرحله فرآیند به وسیله تابع OneLevelCGOL شبیه سازی می شود. روند کار این تابع به وسیله کانولوشن است. با استفاده از طراحی یک فیلتر (keh جزئیات آن در ادامه بررسی می شود) و قراردادن این فیلتر در دستور conv2 خروجی تولید می شود که هر خانه آن برابر است با جمع سلول های زنده همسایه به علاوه نیم در صورت زنده بودن سلول هم شماره. این خروجی به علت این ایجاد می شود که در عملیات conv مربع سه در سه بر روی خانه های مربع اصلی می لغزد درایه به درایه ضرب و حاصل ضرب ها جمع بسته می شود . نیم بودن ضریب خانه مرکزی به ما کمک می کند تا بین زنده بودن سلول مورد بررسی و نبودنش فرق بگذاریم . به عبارتی حالا سه همسایه زنده در حالتی که خود سلول زنده است با حالتی که خود سلول مرده است در جمع ایجاد شده نتیجه مقاوتی دارد در نتیجه می توان بنا به رابطه آورده شده سلول هایی که در مرحله بعد مرده هستند را از سلول های زنده تفکیک کرد . در نهایت خانه های مربوط به سلول های مرده را صفر و مابقی را بنا به خط های سی و هفت و سی و هشت تفکیک کرد . توجه شود که اگر ورودی سوم تابع same نبود خروجی ابعادی متفاوت با map می داشت و خانه های مرزی مقادیری نا مناسب برای کاربرد در اینجا داشتند .

(3.1

۱. در ابتدا برای تابع Heaviside یا همان پله واحد، تغییری ایجاد کرده ایم تا مقدارش در مبدا برابر با ۱ شود. چهار سیگنال مذکور را بر حسب توابعی از n و به شکل symfun تعریف کرده ایم. یک پنجره برای رسم نمودار سیگنال ها ایجاد کرده، آن را ۴ بخش میکنیم و هر سیگنال را در یک بخش رسم میکنیم (محدوده n با استفاده از $n1, n2, n3, n4$ ، و بر اساس محدوده n ناصفر هر سیگنال تعیین شده است). برای رسم سیگنال به صورت گسته از stem استفاده شده است. سپس در $H(z)$ ها، تبدیل z هر سیگنال را ذخیره کرده ایم. با تابع numden ، صورت و مخرج هریک از تبدیل ها را به صورت جفت تابع از z ذخیره میکنیم (صورت و مخرج ها، پس از تغییر فرم هر تبدیل z به شکل کسر گویا بدست آمده است). سپس در H_num/den_coeffs ، ضرایب چند جمله ای های صورت و مخرج را ذخیره کرده ایم. حال با دادن این ضرایب به تابع tf ، تابع انتقال را بدست می آوریم. با دادن این توابع انتقال به دستور های zero,pole ، بردار های صفر و قطب های هر تبدیل z بدست می آید. این بردار ها را در $H_poles/zeros$ ذخیره میکنیم. حال کافیست این بردار ها را به zplane بدهیم تا نمودار های صفر و قطب رسم شوند. همه را داخل یک figure رسم میکنیم.

نمودار پاسخ های ضربه، و نمودار صفر و قطب آنها، به ترتیب در زیر آمده است:

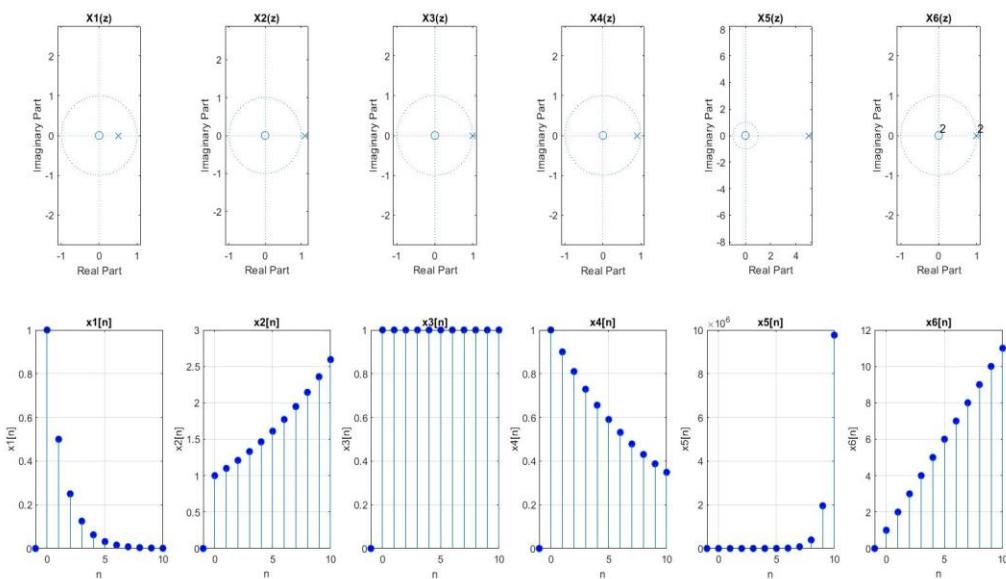


با فرض LTI بودن سیستم ها، میدانیم پایداری سیستم، معادل با این است که ناحیه همگرایی تبدیل z ، شامل دایره i واحد بشود. از طرفی تبدیل z برای هر چهار سیستم به شکل کسر گویاست. با توجه به اینکه تمام سیستم ها دست راستی و علی نیز هستند) همگی به ازای $n < 0$ صفر اند)، پس ناحیه همگرایی همه آنها، خارج بیرونی ترین قطبشان است و تا بینهایت ادامه دارد. با این حساب، ناحیه همگرایی برای $H_1(z)$ و $|z| > 0$ ، برای $H_2(z)$ ، برای $|z| > 0.5$ ، $H_3(z)$ و برای $|z| > 2$ ، $H_4(z)$ است که هرسه ناحیه i اول، شامل دایره واحد میشوند. پس سه سیستم است که هر سه ناحیه i اول، شامل دایره واحد میشوند. پس سه سیستم h_1, h_2, h_3 پایدار اند و h_4 ناپایدار است.

این موضوع را میتوان تایید کرد چرا که از طرف دیگر، سیستم LTI پایدار است، اگر و تنها اگر پاسخ ضربه آن، مطلقاً جمع پذیر (در حوزه گستته) باشد. طبق تعریف سیگنال ها و نیز نمودار های تصویر اول در بالا، همواره کران دار اند واز جایی به بعد (به ترتیب در h_1, h_2 و h_3) صفر میشوند. پس حتماً مطلقاً جمع پذیرند. مقادیر h_3 نیز تشکیل دنباله هندسی با مقدار اولیه 1 و ضریب 0.5 هستند که جمع تمام آنها 2 است. پس h_3 هم مطلقاً جمع پذیر است. اما h_4 یک دنباله هندسی با ضریب بیشتر از 1 است که طبیعتاً به بینهایت میل میکند و در نتیجه h_4 مطلقاً جمع پذیر نیست و سیستم آن پایدار نیست. پس نتایج بدست آمده با دانسته ها و محاسبات ما مطابقت دارد.

۲. کد بخش نمودار صفر و قطب کاملا مشابه بخش قبلی است. در واقع در اینجا ما مستقیماً تبدیل z را داریم و از بعد از دستور `ztrans` در بخش قبل، دستورات کاملاً مشابه است تا نمودارها رسم شوند. چون خواسته شده که تمام نمودارها در یک صفحه باشند، صفحه را توسط `subplot` به صورت 2×3 تنظیم می‌کنیم و در سطر اول، نمودارهای صفر و قطب را رسم می‌کنیم.

برای محاسبه پاسخ ضربه یا همان تبدیل وارون توابع داده شده، از دستور `iztrans` استفاده شده است. البته بدیهی است که در چنین شرایطی، علی‌بودن سیستم‌ها که در صورت سوال بعنوان یک داده‌ی اضافی به ما داده شده، لحاظ نمی‌شود. پس برای لحاظ شدن این ویژگی و صفر کردن مقادیر منفی، لازم است یک پله واحد یا $\text{Heaviside}(n)$ را نیز در `iztrans` ضرب کنیم. **فرمول پاسخ‌های ضربه در $(n) \# x$ ها ذخیره شده است.** محدوده‌ی رسم همه نمودارها از ۱۰ تا ۱ در نظر گرفته ایم و مشابه رسم $h[n]$ ‌ها در بخش قبل، نمودار هریک را رسم کرده ایم. صفحه‌ی نتیجه نهایی به صورت زیر است:



با فرض LTI بودن آنها، پایداری سیستم معادل با این است که ناحیه همگرایی تبدیل z آنها، شامل دایره واحد باشد. چون این سیستم‌ها علی هستند و تبدیل z به شکل کسر گویا دارند، ناحیه همگرایی آنها خارج بیرونی ترین قطب آنهاست. پس این ناحیه برای $(z - X_1(z)) > 0.5$ و برای $(z - X_2(z)) > 1.1$ و برای $(z - X_3(z)) > 1$ و برای $(z - X_4(z)) > 0.9$ و برای $(z - X_5(z)) > 5$ و برای $(z - X_6(z)) > 1$ است. پس سیستم‌های X_1, X_4 پایدار هستند. چون همگی دنباله هندسی هستند، با نمودار پاسخ ضربه‌ها نیز میتوان نشان داد که پاسخ ضربه‌ی x_1, x_4 مطلقاً جمع پذیر و در نتیجه این سیستم‌ها پایدارند.

با توجه به نمودار‌ها، میبینیم اگر قطب‌ها داخل دایره واحد باشند، طبق توضیحات بالا، سیستم پایدار است و در غیر این صورت پایدار نیست. در سیستم‌هایی که قطب‌ها داخل دایره واحدند، پاسخ ضربه در بینهایت همگرایست و به صفر می‌کند. وقتی قطبی روی دایره واحد باشد، یک مقدار ثابت در پاسخ ضربه وجود دارد. و اگر قطب یا قطب‌هایی خارج دایره واحد باشند، جزئی در پاسخ ضربه وجود دارد که در بینهایت همگرایست. علت این سه مورد این است که تبدیل وارون یک کسر به شکل

$$X(z) = \frac{1}{1 - \alpha z^{-1}}, \quad \alpha \in \mathbb{R}$$

و با فرض علی بودن، به صورت زیر است:

$$x[n] = \alpha^n \cdot u[n]$$

رفتار چنین سیگنالی به مقدار α بستگی دارد و اگر بیشتر از ۱ باشد، و اگرا به بینهایت، اگر برابر با ۱ باشد، ثابت و برابر با ۱، و اگر کمتر از ۱ باشد، همگرا به صفر است. پس اگر قطب مرتبه اول داشته باشیم و تبدیل Z به فرم نوشته شده در بالا باشد، پاسخ ضربه، ضریبی از

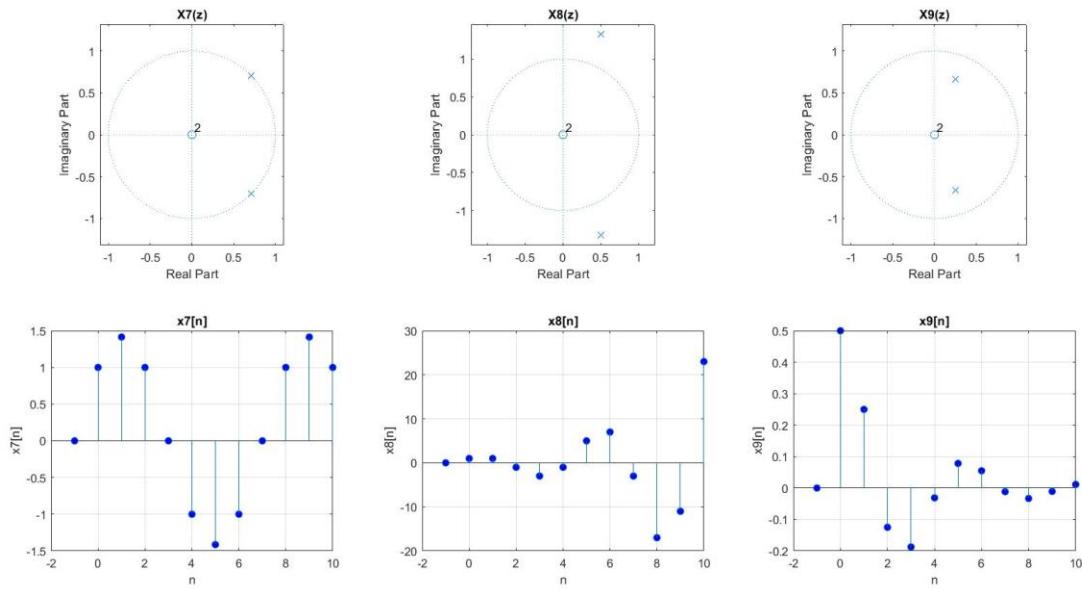
$$\alpha^n \cdot u[n]$$

خواهد بود. اما اگر مرتبه‌ی قطب بیشتر از ۱ باشد، فرم سیگنال تغییر میکند. مثلا برای سیگنال آخر، داریم:

$$X_6(z) = z \cdot \left(-z \cdot \frac{d}{dz} \left(\frac{1}{1 - z^{-1}} \right) \right)$$

عبارة داخل پرانتز، طبق خصوصیات تبدیل Z باعث میشود یک n در سیگنال $u[n]^3$ که همان $u[n]$ بود، ضرب شود و به $n \cdot u[n]$ بررسیم. سپس ضرب شدن یک Z باعث شیفت یک واحدی سیگنال شده و به $(n+1)u[n+1]$ یا همان $un+1$ میرسیم. پس میبینیم از سیگنال درجه صفر به درجه یک رسیدیم. به همین ترتیب اگر مرتبه قطب بالاتر برود، سیگنال به درجه های ۲ و بالاتر میرسد و در نتیجه سرعت رشد سیگنال بیشتر میشود. پس قطب های مکرر باعث ضرب شدن چند جمله ای هایی در سیگنال شده و سرعت رشد را افزایش، و یا سرعت کاهش مقدار سیگنال را کم میکنند (اگر قطب ها داخل دایره واحد باشند، سیگنال همگرا میشود).

۳. واضح است که کلیت کداملا مشابه قبل است و فقط توابع جدید در این بخش نوشته شده اند و مشابه قبل نمودار آنها رسم شده است:



تفاوت اصلی در این سیگنال‌ها نسبت به موارد قبل، داشتن یک بخش به فرم متناوب است. در واقع وقتی قطب مختلط داریم، حتماً مزدوج مختلط آن را نیز داریم. پس فرم‌های $\alpha^n \cdot u[n]$

و

$$(\alpha^*)^n \cdot u[n]$$

را در پاسخ‌های خود داریم که α و مزدوج مختلط آن، قطب هستند. میتوان نوشت:

$$\alpha = r \cdot e^{j\theta}$$

که در نتیجه،

$$\alpha^n = r^n \cdot e^{j\theta n}$$

است. عبارت $r^n e^{j\theta n}$ حقیقی است و دامنه را تعیین میکند و عبارت n نیز فرم متناوب دارد و بخش متناوب را میسازد. توجه داریم که چون دو قطب مزدوج مختلط داریم که ضریب متناظر با این دو نیز مزدوج مختلط یکدیگرند، در این بخش متناوب، جزئی های موهومی ساده میشوند و در نهایت، فقط بخش حقیقی متناوب باقی میماند. با رسیدن به توان n ، واضح است که اگر قطب مختلط، شعاع کمتر از ۱ داشته باشد، در n های بزرگ، دامنه را به صفر میل میدهد. در نتیجه مقدار سیگنال همواره کران دار شده و مطلقاً جمع پذیر نیز هست. **پس سیستم ۹X که قطب هایش داخل دایره واحدند، پایدار است.** اگر قطب ها روی دایره واحد باشند، ضریب ثابت ۱ داریم و در نتیجه دامنه تغییر نمیکند. پس یک سیگنال متناوبی و سینوسی داریم که با دامنه ثابت تا بینهایت ادامه دارد. چنین سیگنالی مطلقاً جمع پذیر نیست چون اگر از مقادیر سیگنال قدر مطلق گرفته و با هم جمع کنیم، بینهایت تناوب داریم که در هر تناوب یک مقدار مثبتی وجود دارد و در نتیجه حاصل بینهایت خواهد شد. **پس سیگنال ۷X که قطب هایش روی دایره واحدند، پایدار نیست.** در نهایت اگر قطب ها خارج دایره واحد باشند، با وجود $|z| > 1$ ، دامنه سیگنال متناوب پیوسته زیاد میشود و در n های بزرگ به مقدار بینهایت میل میکند. پس برخلاف حالت اول که متناوب میرا بود، اینجا متناوب افزایشی داریم. طبیعتاً با قدر مطلق گرفتن و جمع زدن مقادیر این سیگنال به بینهایت میرسیم. در نتیجه **سیگنال ۸X که قطب هایش خارج دایره واحدند، پایدار نیست.**

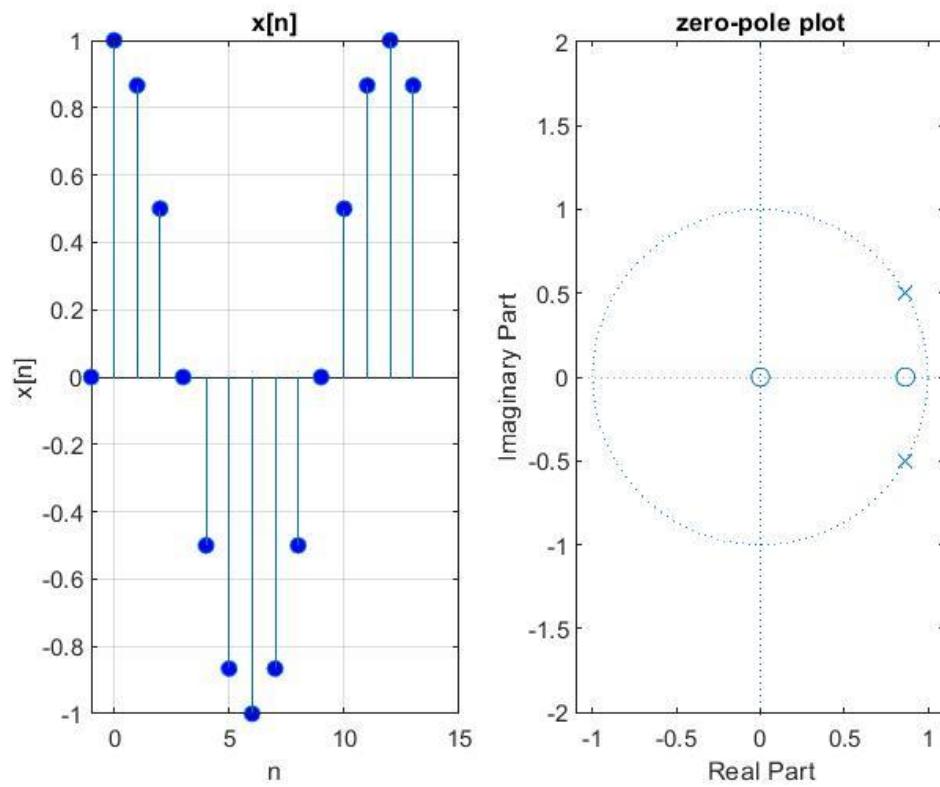
روش دیگر بررسی پایداری از روی نمودار صفر و قطب، این است که مطابق توضیحات بخش های قبل، باید ناحیه همگرایی شامل دایره واحد بشود. با فرض $z = r e^{j\theta}$ و $|z| < 1$ بودن سیستم ها، ناحیه همگرایی خارج بیرونی ترین قطب است. با این حساب تنها سیگنالی که ناحیه همگرایی اش شامل

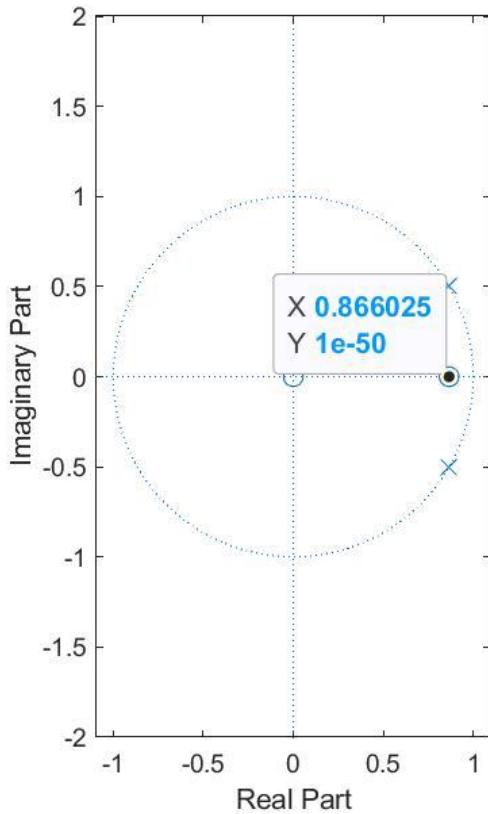
دایره واحد است، $9X$ است و در نتیجه تنها سیستم پایدار بین این سه سیستم است.

(3.2)

۱. ابتدا مشابه بخش های قبل، در تعریفتابع پله، تغییری ایجاد کرده و آن را در مبدا برابر با ۱ قرار میدهیم. سپس $[n]x$ را همانطور که در صورت سوال خواسته شده تعریف میکنیم (تابع $\cos pi$ به آرگومان کسینوس، عدد پی را هم ضرب میکند و حاصل دقیقتری نسبت به ضرب کردن مستقیم عدد پی دارد). توجه داریم که چون تبدیل z مطابق آنچه در مطلب تعریف شده، به صورت یکطرفه (از صفر تا بینهایت) محاسبه میشود، بنابراین برای محاسبه تبدیل z با دستور $ztrans$ ، میتوان سیگنالی بدون ضرب در پله واحد را به $ztrans$ داد. بنابراین $(n)x_z$ برای محاسبه تبدیل z و بدون $Heaviside(n)$ تعریف شده است. حاصل تبدیل z در $(z)X$ ریخته شده و چاپ شده است. سپس با $numden$ ، صورت و مخرج کسر گویای این تبدیل را مشخص کرده و داخل متغیر های جدید ریخته ایم. با دستور $coeffs$ و وارد کردن 'All' هم ضرایب چندجمله ای های صورت و مخرج بدست آمده اند. با استفاده از این ضرایب و دستور tf ، تابع انتقال را تشکیل داده ایم. دستور های $zero,pole$ ، صفر ها و قطب های این تابع انتقال را بدست می آورند و در نهایت با دادن این صفر و قطب ها به $zplane$ ، نمودار صفر و قطب را رسم میکنیم. نمودار خود سیگنال $(n)x$ هم با دستور $stem$ به صورت گسته رسم میشود که چون دوره تناوب آن ۱۲ است، محدوده ورودی یا

را از ۱- تا ۱۳ گرفته ایم. نمودار ها به صورت زیر هستند:





و مطابق محاسبات متلب، تبدیل z به صورت زیر است:

$$z^2/(z^2 - 3^{(1/2)*z} + 1) - (3^{(1/2)*z})/(2*(z^2 - 3^{(1/2)*z} + 1))$$

یا همان:

$$X(z) = \frac{z^2 - \sqrt{3}/2z}{z^2 - \sqrt{3}z + 1}$$

پس صفر ها در

$$z = 0, z = \sqrt{3}/2$$

هستند و قطب ها هم ریشه های مخرج یا $z^2 - 3^{(1/2)*z} + 1$ هستند که برابرند با:

$$z = \frac{\sqrt{3}}{2} + \frac{1}{2}j, z = \frac{\sqrt{3}}{2} - \frac{1}{2}j$$

و در نمودار مشخص شده اند.

با توجه به اینکه تبدیل به شکل کسر گویاست و خود سیگنال نیز دست راستی و علی است، ناحیه همگرایی خارج بیرونی ترین قطب است یعنی: $|z| > 1$.

۲. این بخش با کامنت 2 part در کد مشخص شده است. مطابق آنچه سوال خواسته، $(z), X_0(z), X_1(z)$ را تعریف میکنیم. مراحل رسم نمودار صفر و قطب برای X_1 کاملا مشابه بخش قبل است. برای سیگنال حوزه زمان کافیست تبدیل وارون را با iztrans حساب کنیم (چون مبنای محاسبات متلب، تبدیل z یک طرفه است، حاصل iztrans ، بدون ضرب در پله واحد برگردانده میشود که ما این ضرب را جداگانه انجام داده ایم و حاصل را در $x(n)$ ذخیره کرده ایم). حال مشابه بخش قبل و با stem نمودار حوزه زمان را هم رسم میکنیم. از شکل نمودار و پاسخ ذخیره شده در $x(n)$ ، میبینیم که حاصل زیر بدست آمده است:

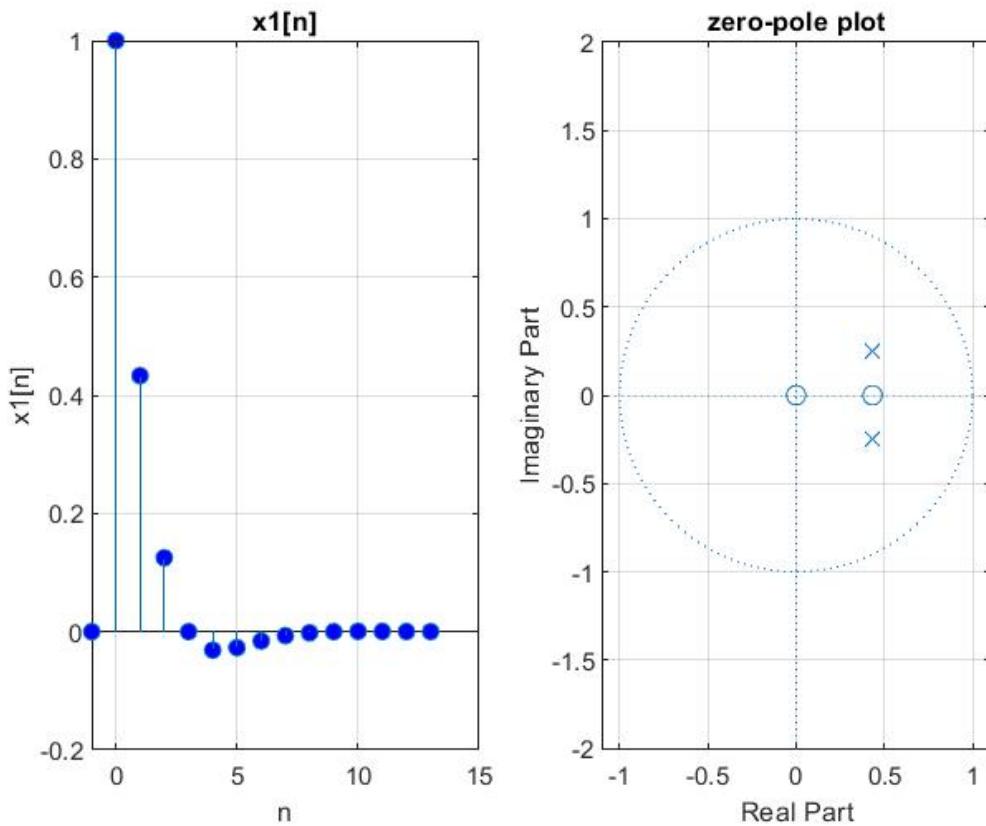
$$x_1(n) = 0.5^n * (-1)^n \cos\left(n * \frac{5\pi}{6}\right) u[n]$$

که اگر از آرگومان کسینوس، یک $n\pi$ کم کنیم، به صورت ساده تر داریم:

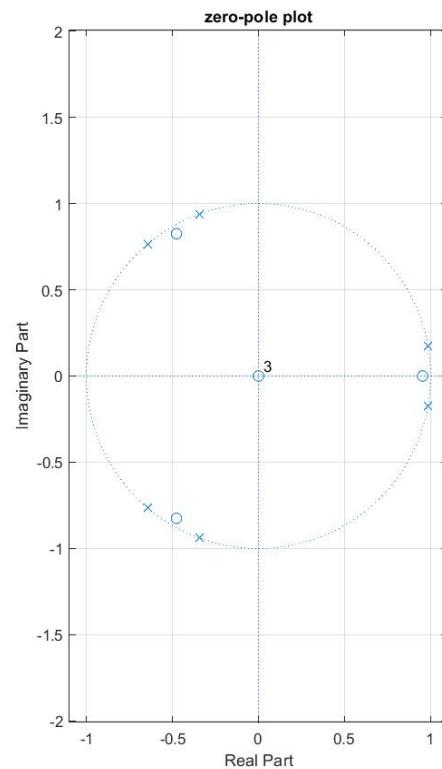
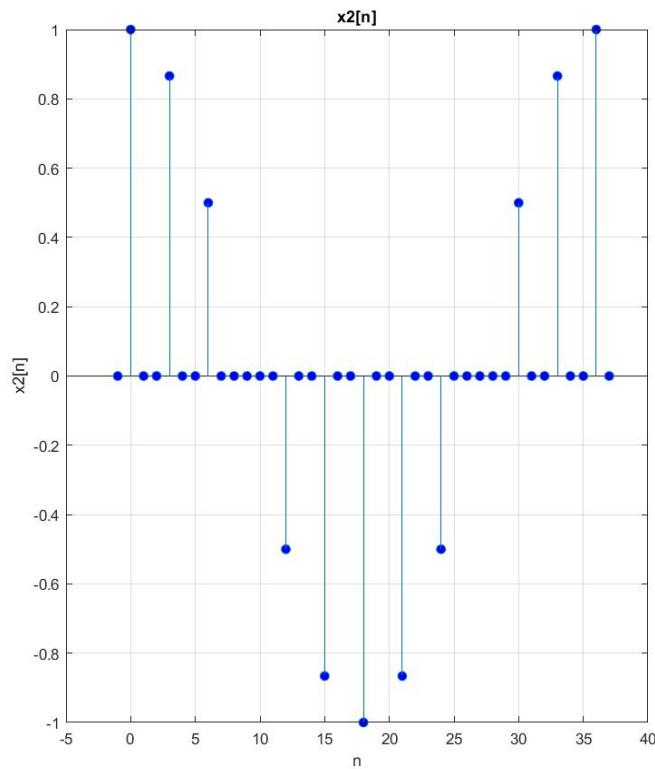
$$x_1(n) = 0.5^n * \cos\left(n * \frac{\pi}{6}\right) u[n]$$

که نشان میدهد تنها تفاوت حاصله نسبت به سیگنال قبل، ضرب شدن یک 0.5^n میباشد. از قبل هم میدانیم اگر $X(z)$ تبدیل z سیگنال $x[n]$ با ناحیه

همگرایی R باشد، $X(z) = z^0 * x[n]$ متناظر با سیگنال $x[n]$ خواهد بود و واضح است که ناحیه همگرایی و تمام نقاط در حوزه z در $|z| > 0$ ضرب میشوند. در این سوال، $z=0.5$ است و چنین سیگنال و نمودار صفر و قطبی را به وجود آورده است. طبق توضیحات ذکر شده، صفر و قطب ها در این بخش، نصف سیگنال قبلی هستند. تصویر نمودارها به صورت زیر است:



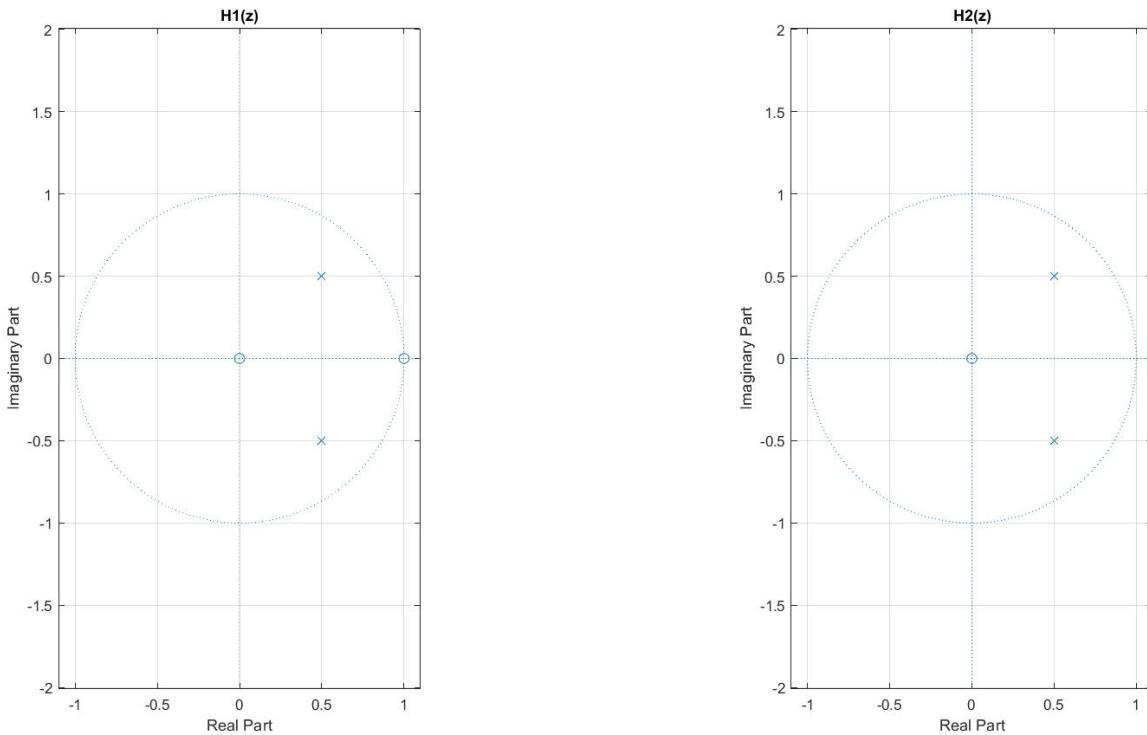
۳. این بخش با کامنت 3 part در کد مشخص شده است. ساختار کلی کد کاملا مشابه بخش قبل است و فقط X باید مطابق خواسته سوال تعریف شود. تصویر نمودارها به صورت زیر است:



تفاوتی که سیگنال زمان با سیگنال اولیه دارد این است که با ضریب ۳ منبسط شده است. به بیان دیگر تمام n ها در ۳ ضرب شده اند. چیزی شبیه به $x[n/3]$ است با این توضیح که برای n هایی که مضرب ۳ نیستند مقدار صفر داریم. از قبل هم میدانستیم اگر $X(z)$ تبدیل z سیگنال $x[n]$ با ناحیه همگرایی R باشد، $X(z^k)$ متناظر با سیگنال $x[n/k]$ ای است که به ازای n هایی که مضرب k نیستند، صفر تعریف شده است (آن را با $x_{(k)}[n]$ نشان میدهند). واضح است که در چنین شرایطی، دوره تناب سیگنال هم k برابر میشود. در اینجا $k=3$ است.

(3.3

۱. کد و نحوه به دست آوردن نمودار صفر و قطب کاملا مشابه بخش قبلی (3.2.3) است. H_1, H_2 را مطابق خواسته سوال تعریف کرده و باقی کد مشابه قبل است. نمودار به شکل زیر است:



چون تبدیل ها به صورت کسر گویا اند و سیستم ها نیز علی اند، ناحیه همگرایی خارج بیرونی ترین قطب است. با توجه به نمودار و قطب ها که $z = 0.5 + 0.5j$ هستند، در هردو تابع، شعاع قطب ها $0.5\sqrt{2}$ است. پس ناحیه همگرایی هردو تابع، $|z| > 0.5\sqrt{2}$ است. با فرض LTI بودن سیستم ها، پایداری آنها معادل با این است که ناحیه همگرایی آنها شامل دایره واحد بشود. با توجه به ROC که مشخص شد، به وضوح شامل دایره واحد است و بنابراین هردو سیستم پایدارند.

۳ و ۲. در a_1 و b_1 ، به ترتیب ضرایب چند جمله‌ای های صورت و مخرج $H_1(z)$ (از درجه زیاد به کم) قرار داده شده است. این دو را به ورودی های residuez میدهیم تا به ترتیب، r و p و k در فرمول مذکور در سوال را بگیریم. دقیقاً همین کار برای $H_2(z)$ نیز انجام شده است. پس از ران کردن برنامه، برای H_1 به مقادیر زیر میرسیم:

$$r_1 = [0.5(1 + j); 0.5(1 - j)]$$

$$p_1 = [0.5(1 + j); 0.5(1 - j)]$$

$$k_1 = \emptyset$$

پس فرم تجزیه شده زیر را داریم:

$$H_1(z) = \frac{0.5(1 + j)}{1 - 0.5(1 + j)z^{-1}} + \frac{0.5(1 - j)}{1 - 0.5(1 - j)z^{-1}}$$

با توجه به اینکه در سیستم علی، وارون فرم $\frac{ri}{1 - pi * z^{-1}}$ به صورت

$$ri * pi^n * u[n]$$

است، داریم:

$$h_1[n] = ((0.5(1 + j))^{n+1} + (0.5(1 - j))^{n+1})u[n]$$

$$h_1[n] = \left(\left(\frac{\sqrt{2}}{2} * e^{\frac{j\pi}{4}} \right)^{n+1} + \left(\frac{\sqrt{2}}{2} * e^{\frac{-j\pi}{4}} \right)^{n+1} \right) u[n]$$

$$h_1[n] = 2 * \left(\frac{\sqrt{2}}{2} \right)^{n+1} \cos \left(\frac{(n+1)\pi}{4} \right) u[n]$$

در کد هم با دستور `iztrans` تبدیل وارون گرفته شده و در $h1(n)$ ریخته شده است. در زیر خواهیم دید که حاصل در مطلب چه بدست می آید و چرا با حاصل بالا برابر است:

$$\begin{aligned}
 h_1(n) &= \text{حاصل مطلب} : h_1(n) = \left((-1)^n (1-j)^n + (1+j)^n \right) u[n] \\
 \Rightarrow h_1(n) &= \frac{(1+j)^{n+1} + (1-j)^{n+1}}{\gamma^{n+1}} = \frac{(\sqrt{2})^{n+1} \left(e^{\frac{j\pi}{4}(n+1)} + e^{-\frac{j\pi}{4}(n+1)} \right)}{\gamma^{n+1}} u[n] \\
 &= \frac{\gamma}{(\sqrt{2})^{n+1}} \cos\left(\frac{(n+1)\pi}{4}\right) u[n]
 \end{aligned}$$

حال دقیقا همین فرایند را برای $H2(z)$ تکرار میکنیم. مقادیر $r2, p2, k2$ به صورت زیر است:

$$r1 = [-j; +j]$$

$$p2 = [0.5(1+j); 0.5(1-j)]$$

$$k2 = \emptyset$$

پس مشابه قبل:

$$H2(z) = \frac{-j}{1 - 0.5(1+j)z^{-1}} + \frac{j}{1 - 0.5(1-j)z^{-1}}$$

$$h2[n] = (-j * (0.5(1+j))^n + j * (0.5(1-j))^n)u[n]$$

$$h2[n] = j * \left(-\left(\frac{\sqrt{2}}{2} * e^{\frac{j\pi}{4}} \right)^n + \left(\frac{\sqrt{2}}{2} * e^{\frac{-j\pi}{4}} \right)^n \right) u[n]$$

$$h2[n] = 2 * \left(\frac{\sqrt{2}}{2} \right)^n \sin \left(\frac{n\pi}{4} \right) u[n]$$

به صورت دستی و با توجه به حاصل مطلب نیز داریم:

$$\begin{aligned}
 h_r(n) &= \frac{(-1)^n \left(-2 \times 2^{-\frac{n}{4}} \cos \left(\frac{3\pi}{4} n \right) + (-1-j)(1+j) + (-1+j)(1-j) \right)}{r^n} \\
 u[n] \Rightarrow h_r(n) &= 2 \times 2^{-\frac{n}{4}} \times (-1)^n \cos \left(\frac{3\pi}{4} n \right) + \frac{(1+j)^{n+1} (1-j)^{n+1}}{r^n} \\
 &= 2 \times 2^{-\frac{n}{4}} \times (-1)^n \cos \left(\frac{3\pi}{4} n \right) - \frac{\sqrt{2}}{(jr)^n} \cos \left(\frac{\pi}{4} (n+1) \right) \\
 &= \frac{\sqrt{2}}{(jr)^n} \left[\cos \left(\frac{3\pi}{4} n - n\pi \right) - \sqrt{2} \cos \left(\frac{\pi}{4} (n+1) \right) \right] \\
 &\text{و } \sqrt{2} \cos \left(\frac{n\pi}{4} + \frac{\pi}{4} \right) = \sqrt{2} \left(\frac{\sqrt{2}}{2} \cos \left(\frac{n\pi}{4} \right) - \frac{\sqrt{2}}{2} \sin \left(\frac{n\pi}{4} \right) \right) = \cos \left(\frac{n\pi}{4} \right) - \sin \left(\frac{n\pi}{4} \right) \\
 \Rightarrow h_r(n) &= \frac{1}{(jr)^n} \left[\cos \left(\frac{n\pi}{4} \right) - \cos \left(\frac{n\pi}{4} \right) + \sin \left(\frac{n\pi}{4} \right) \right] = \frac{1}{(jr)^n} \sin \left(\frac{n\pi}{4} \right) u[n]
 \end{aligned}$$

که مشخصا با هم برابرند. توجه داریم همانطور که در بخش های قبل گفته شد، حاصل بدست آمده در مطلب بدون $u[n]$ نوشته میشود چون

مبانی محاسبات متلب، تبدیل z یک طرفه است و فرض بر این است که تمام این سیگنال‌ها $u[n]$ را دارند.

۴. اگر سیستم ضد علی باشد، سیگنال متناظر با کسر کلی $\frac{ri}{1-pi*z^{-1}}$ به صورت زیر خواهد بود:

$$-ri * pi^n * u[-n - 1]$$

در بخش قبل r, p, k ها را بدست آورده بودیم و حالا فقط باید فرم جدید را بنویسیم:

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

$$h1[n] = -\left(\left(\frac{\sqrt{2}}{2} * e^{\frac{j\pi}{4}}\right)^{n+1} + \left(\frac{\sqrt{2}}{2} * e^{\frac{-j\pi}{4}}\right)^{n+1}\right)u[-n - 1]$$

$$h1[n] = -2 * \left(\frac{\sqrt{2}}{2}\right)^{n+1} \cos\left(\frac{(n+1)\pi}{4}\right)u[-n - 1]$$

$$h2[n] = \left(+j * (0.5(1+j))^n - j * (0.5(1-j))^n\right)u[-n - 1]$$

$$h2[n] = j * \left(\left(\frac{\sqrt{2}}{2} * e^{\frac{j\pi}{4}}\right)^n - \left(\frac{\sqrt{2}}{2} * e^{\frac{-j\pi}{4}}\right)^n\right)u[-n - 1]$$

$$h2[n] = -2 * \left(\frac{\sqrt{2}}{2}\right)^n \sin\left(\frac{n\pi}{4}\right)u[-n - 1]$$

با استفاده از $iztrans$ نمیتوان به این جواب ها رسید چون مبنای محاسبه $iztrans$ (و نیز $ztrans$) ، تبدیل z یکطرفه است. یعنی در تعریف تبدیل z ، زیگما از صفر به بعد گرفته میشود. در نتیجه فرض بر این است که تمام سیگنال ها علی و دست راستی اند و سیگنال های ضدعلی از این روش بدست نمی آیند.

(3.4

۱. در محاسبات زیر، تابع تبدیل بدست آمده است:

$$1. \quad Y(z) - 0,7z^{-1}Y(z) + 0,49z^{-2}Y(z) = X(z) - z^{-1}X(z)$$

$$\Rightarrow H(z) = \frac{Y(z)}{X(z)} = \frac{1 - z^{-1}}{1 - 0,7z^{-1} + 0,49z^{-2}}$$

اینتابع تبدیل را در مطلب تعریف کرده و مشابه بخش قبل، با استفاده از residuez، فرم تجزیه شده را بدست می آوریم و نمایش میدهیم. مقادیر r و p به ترتیب به صورت زیر هستند و k هم نداریم:

$$r = [1 + 0.2474j; 1 - 0.2474j]$$

$$p = [0.35 + 0.6062j; 0.35 - 0.6062j]$$

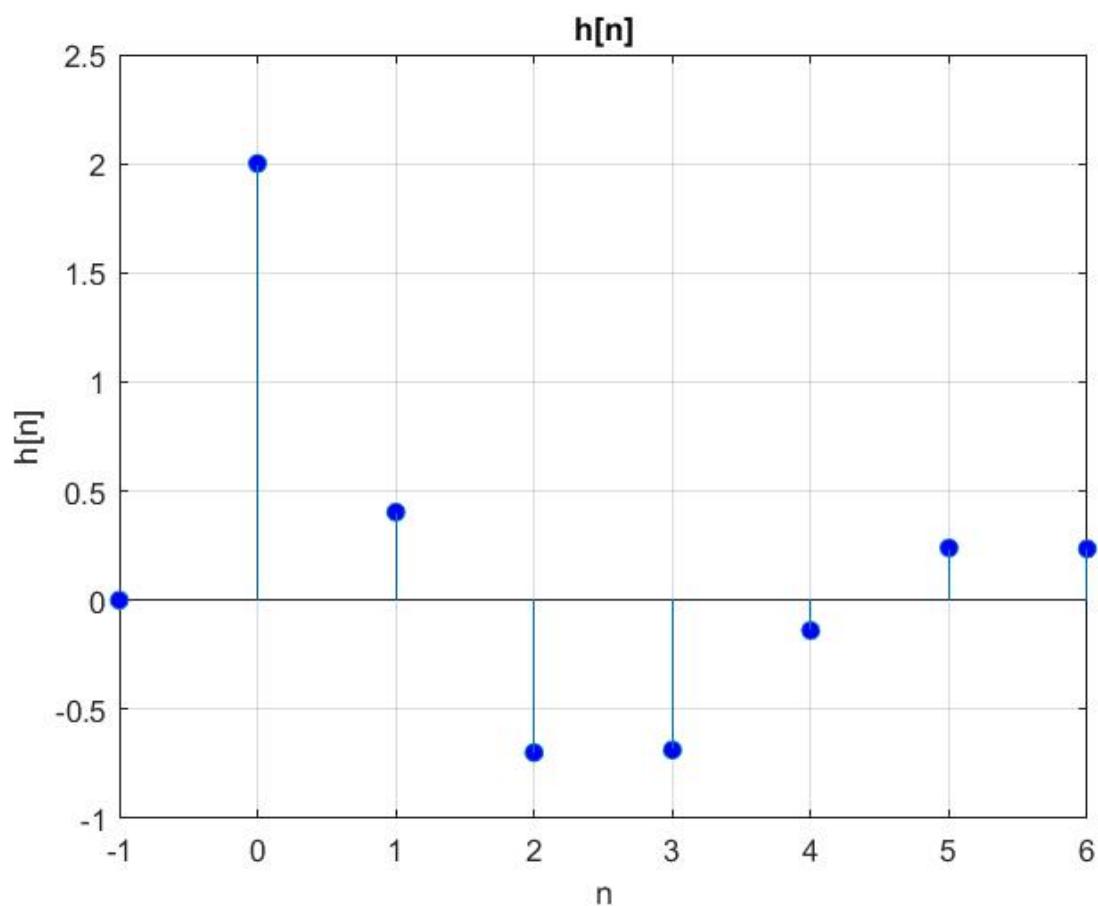
همانطور که در بخش قبل گفته شد، در سیستم علی، وارون فرم $\frac{ri}{1 - pi * z^{-1}}$ به صورت

$$ri * pi^n * u[n]$$

است. پس محاسبات را در مطلب با این مبنای انجام میدهیم و به عبارت زیر $h(n)$ میرسیم. این عبارت را به شکل زیر ساده میکنیم:

$$\begin{aligned}
 h[n] &= -\left(\frac{\sqrt{3}j}{V} - 1\right) \left(\frac{V}{V} - V \frac{\sqrt{3}j}{V}\right)^n + \left(\frac{\sqrt{3}j}{V} + 1\right) \left(\frac{V}{V} + V \frac{\sqrt{3}j}{V}\right)^n \\
 &\approx \left(1,0^3 e^{-0,24j}\right) \left(0, V e^{-j\frac{\pi}{3}}\right)^n + \left(1,0^3 e^{0,24j}\right) \left(0, V e^{j\frac{\pi}{3}}\right)^n \\
 &= 1,0^3 \times (0, V)^n \left[\frac{-j(1+0,24+\frac{n\pi}{3})}{e^{-j(0,24+\frac{n\pi}{3})}} + \frac{j(1,24+\frac{n\pi}{3})}{e^{j(0,24+\frac{n\pi}{3})}} \right] \\
 h[n] &\approx 1,0^3 \times (0, V)^n \underbrace{c.s(0,24+\frac{n\pi}{3})}_{h[n]} \quad \boxed{h[n]}
 \end{aligned}$$

حال همین $h(n)$ که ساده شد را در متلب در $h_simp(n)$ ریخته و آن را رسم میکنیم:



۲. از فرم پاسخ بدست آمده توسط متلب برای $(h(n))$ ، واضح است که قطب های ما به صورت زیر هستند:

$$p1 = \frac{7}{20} + \frac{7\sqrt{3}}{20}j, p2 = \frac{7}{20} - \frac{7\sqrt{3}}{20}j$$

حال باید ضرایب این دو را بیابیم. برای این کار $n=0,1$ را در معادله تفاضلی جایگذاری میکنیم تا شروط اولیه پیدا شوند. توجه داریم چون پاسخ ضربه را میخواهیم، بجای $x[n]$ ، $\delta[n]$ و بجای $y[n]$ ، $h[n]$ قرار میگیرد. ضمناً بدلیل علی بودن، پاسخ ضربه به ازای $n < 0$ ، صفر است:

$$n = 0: h[0] - 0.7 * 0 + 0.49 * 0 = 2 * 1 - 0$$

$$h[0] = 2$$

$$n = 1: h[1] - 0.7 * h[0] + 0.49 * 0 = 2 * 0 - 1$$

$$h[1] = 0.4$$

حال همین n ها را در فرم جدید پاسخ ضربه جایگذاری میکنیم:

$$n = 0: h[0] = \alpha_1 + \alpha_2 = 2$$

$$n = 1: h[1] = \left(\frac{7}{20} + \frac{7\sqrt{3}}{20}j \right) * \alpha_1 + \left(\frac{7}{20} - \frac{7\sqrt{3}}{20}j \right) * \alpha_2$$

$$0.4 = \frac{7}{20} * 2 + \frac{7\sqrt{3}}{20}j(\alpha_1 - \alpha_2)$$

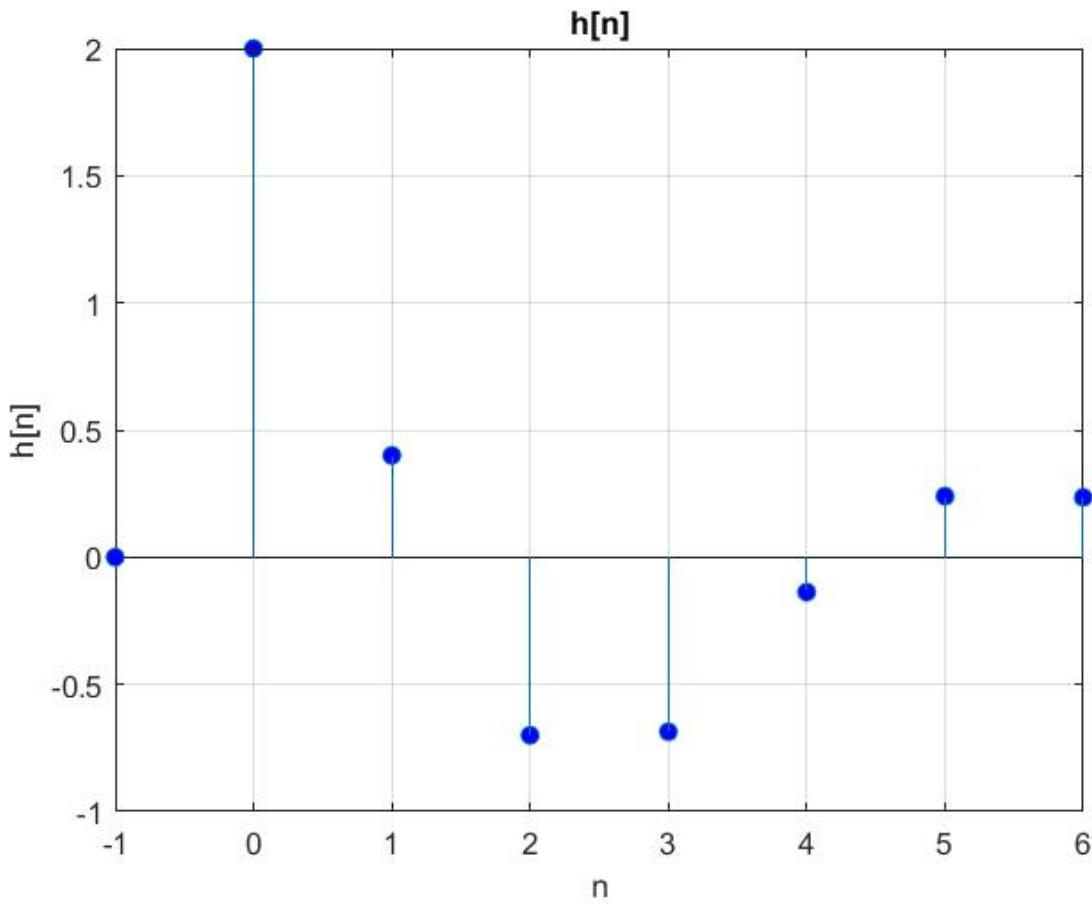
$$\alpha_1 - \alpha_2 = \frac{6}{7\sqrt{3}}j$$

$$\begin{cases} \alpha_1 = 1 + \frac{\sqrt{3}}{7}j \\ \alpha_2 = 1 - \frac{\sqrt{3}}{7}j \end{cases}$$

پس پاسخ ضربه به صورت زیر است:

$$h[n] = \left(1 + \frac{\sqrt{3}}{7}j\right) \left(\frac{7}{20} + \frac{7\sqrt{3}}{20}j\right)^n + \left(1 - \frac{\sqrt{3}}{7}j\right) \left(\frac{7}{20} - \frac{7\sqrt{3}}{20}j\right)^n$$

این پاسخ دقیقاً برابر با چیزی است که در متلب و با استفاده از residuez بدست آمد. در بخش قبل نمودار فرم ساده شده‌ی پاسخ را رسم کردیم و اینجا هم نمودار پاسخ دقیق و اصلی را رسم می‌کنیم و میبینیم مشابه هم هستند:



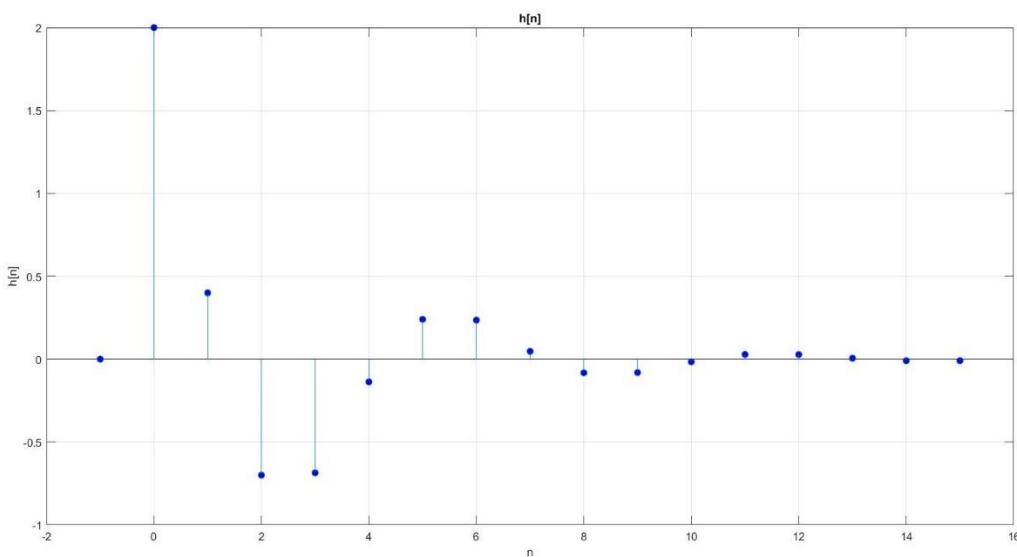
۳. در بالاتر تابع تبدیل را بدست آورده بودیم:

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

عملکرد این تابع برای پاسخ به این سوال بدین شکل است که دو بردار ورودی اول، به ترتیب باید بردار های ضرایب صورت و مخرج تابع تبدیل سیستم باشند، و بردار سوم همان بردار ورودی به این سیستم است. بدین ترتیب در خرجی میتوان پاسخ این سیستم به این ورودی را گرفت. در کد b و a به ترتیب ضرایب صورت و مخرج تابع تبدیل هستند.

بردار t یک بردار برای تعیین محدوده ای ورودی ماست که آن را از ۱ تا ۱۵ گرفته ایم (همانطور که از رابطه ای پاسخ ضربه در بخش های قبل

دیده شد، دامنه دائما در حال کاهش است و در n های بزرگتر، حاصل بسیار نزدیک به صفر است و نمودار واضحی را تشکی نمیدهد). حال باید ورودی ضربه باشد، یعنی فقط در $t=0$ برابر با ۱ ، و سایر جاها صفر باشد. بنابراین قرار میدهیم: $x(t==0)$. حال h را توسط filter با ورودی های مذکور بدست می آوریم و همراه با t نمودار را رسم میکنیم. نمودار به شکل زیر است که کاملا با بخش های قبل تطابق دارد:



در واقع در استفاده از این تابع، سیستم ما به چشم یک فیلتر دیده میشود که تابع توصیف کننده‌ی آن با ضرایب a, b مشخص میشود. سپس با دادن یک ورودی، خروجی حاصل از عبور این ورودی از فیلتر به ما داده میشود.

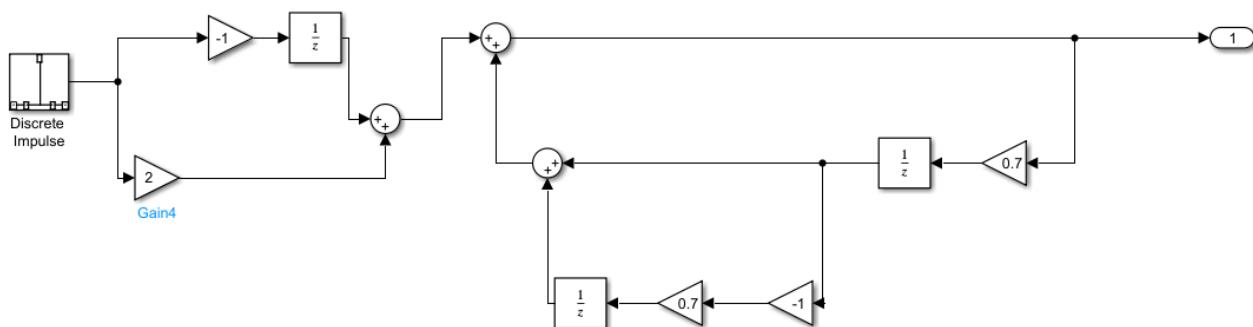
۴. اگر رابطه‌ی تابع تبدیل را طرفین وسطین کنیم، به رابطه زیر میرسیم:

$$(1 - 0.7z^{-1} + 0.49z^{-2})Y(z) = (2 - z^{-1})X(z)$$

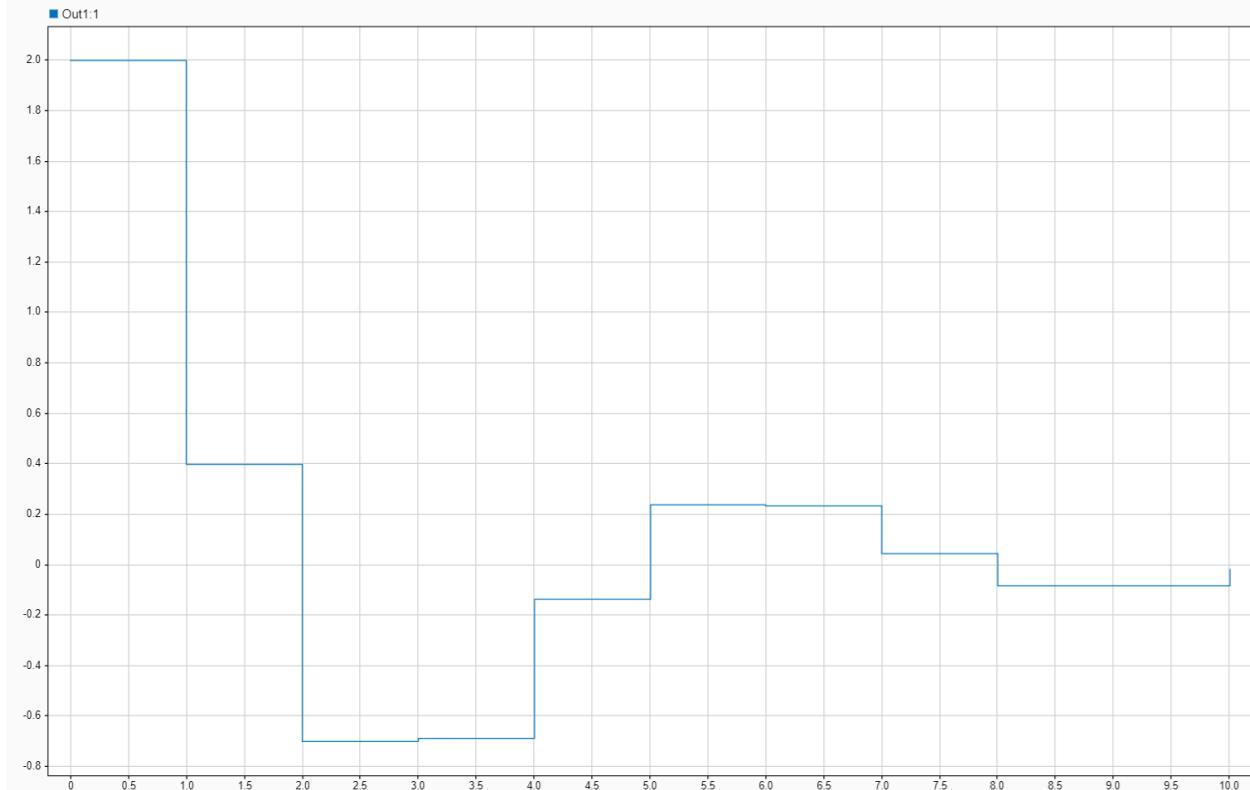
: پس

$$Y(z) = (0.7z^{-1} - 0.49z^{-2})Y(z) + (2 - z^{-1})X(z)$$

برای ساخت بخش $(z)X$ از این معادله، یک گین ۱- را به $(-1)z^8$ یا همان unit delay وصل کرده و با sum با یک گین ۲ و متصل به ورودی، جمع می‌کنیم. بخش بعدی نیز یک sum دارد که ورودی هایش $0.7z^{-1}Y(z)$ و $-0.49z^{-2}Y(z)$ هستند. ورودی اول با عبور دادن خروجی از یک گین 0.7 و یک delay ساخته می‌شود. اگر همین حاصل را از یک گین 0.7 و delay دیگر به همراه گین ۱- عبور دهیم، ی دوم نیز ساخته می‌شود. بنابراین شکل نهایی به صورت زیر است:



نمودار نهایی خروجی به صورت زیر است، که اگر به صورت گستته به آن نگاه کنیم، کاملاً با نمودار های بخش های قبل برابر است:



گزارش سوال ۴

Karplus-Strong متدی است که از آن برای شبیه سازی صدای ناشی از یک سیم (رشته) به ارتعاش در آمده استفاده می شود . در این متد از حلقه فیدبک استفاده شده است . در طی آن ابتدا ورژن N بار شیفت خورده سیگنال خروجی با یک سیگنال ورودی که اغلب نویز سفید است جمع می شود (البته سیگنال x می تواند به ازای مقادیر دیگر مثل تابع سینوسی کاربرد هایی داشته باشد) . سپس مقدار به دست آمده از یک فیلتر پایین گذر عبور داده می شود (در اینجا از میانگیر گیری با مقدار قبلی به عنوان فیلتر پایین گذر استفاده شده) سیگنال ایجاد شده در نهایت مانند سیگنال صوت یک سیم مرتعش عمل می کند . با تعیین مناسب طول y,x می توان انتظار داشت که بتوانیم از آن به عنوان یک گیتار خیالی استفاده کنیم (توضیحات مربوطه در کد آمده) . جالب است بدانید که اتفاقا این متد در موسیقی کاربرد هایی داشته . اولین کاربرد این الگوریتم به سال ۱۹۸۱ بر می گردد جایی که در تهیه اثر Acrobats May All Your Children Be از آن استفاده شد . هرچند دارندگان لایسنس این ابداع در نهایت خوش شناس نبودند و ورشکست شدند ، اما شکست آن ها پایان کار این الگوریتم نمود و هنوز بسیاری از قطعات سخت افزاری فعل در این حوزه از اصول اولیه این روش برای فعالیت های خود استفاده می کنند

در ابتدا فرکانس نمونه برداری و اسم فایلی که قرار است در آن ذخیره سازی فایل صوتی صورت پذیرد مشخص شده است . سپس نت ها و طول هر نت آمده . همه اطلاعات لازم برای ایجاد آهنگ (یعنی نت ها و مدت زمانشان) به تابع songmaker داده می شود . این تابع در ابتدا تعداد نت هارا مشخص می کند از آنجا که نت ها به شکل یک بردار یک در ۰ داده می شوند (۰ سایز بردار است) برای همین لازم است پس از استخراج مقدار ۰ یک حلقه for به تعداد ۰ بار اجرا می شود و هر ۰ نت را به شکل یک بردار استخراج می کند . برای این کار در ابتدا با استفاده از تابع notefreq فرکانس نت استخراج می شود این تابع الگوریتم بسیار ساده ای دارد . کاراکتر نت را می گیرد (یعنی j(note)) و با مجموعه ای از if else ، مقدار فرکانس نت را استخراج می کند . در نهایت از کاربرد تابع generateNote استفاده می کنیم . این تابع فرکانس نت (که در خط بالا مشخص شد) و طول نت (j(noteDurations)) را به همراه ضریب الfa (این ضریب نقش جالبی در ایجاد صدای نت دارد . مقدار نزدیک به یک آن صدای نزدیک تری به واقعیت ایجاد می کند در حالی که کم کردن این ضریب معادل این است که با دست سیم را بگیرید و صدای محدودی ایجاد کنید) که برای ایجاد کیفیت مناسب برای همه نت ها ۰,۹۹۵ در نظر گرفته شده دریافت می کند تا بنا به الگوریتم نت را ایجاد کند . طول x بنا به دستور کار ایجاد می شود و طول y از رابطه $F_s * 1/1000$ duration(ms) به دست می آید چرا که هر نمونه عددی در آن معادل یک نمونه است و تعداد نمونه ها در آن از ضرب فرکانس نمونه گیری در زمان به دست می آید . x به این صورت ایجاد می شود که در ابتدا یک بردار رندم با بازه بین صفر تا یک ایجاد می شود و دو برابر شده از آن یک کم می شود تا بازه اش بین یک تا منفی یک تنظیم شود . در ادامه N عضو اول y برابر با ایکس داده شده چرا که y های منفی مقدار صفر دارد و به همین علت بنا به رابطه د عضو اول به شکل مستقیم در N خانه اول y قرار می گیرد . خانه N+1 به همین علت و بنا به رابطه به صورت نوشته شده در آمده (x ها دیگر مقدار ندارند و در رابطه تنها $y(n-N=1)$ است که مقدار دارد) . مابقی خانه ها با یک حلقه for و بنا به رابطه شکل می گیرد . در نهایت چون کار ما با بردار های غیر ستونی است بردار را ترسیپ می کنیم . این مقدار ایجاد شده به آخر آرایه songmaker در اضافه می شود .