IIR Filter design

14.1 Ustalenie postaci filtrów cyfrowych

Poniżej znajdują się definicje filtrów analogowych i ich cyfrowych odpowiedników w postaci tabel dla rzędu N={1,2,...7} oraz częstości ω={ωmin, ωmax, $\Delta\omega$}

14.1.1 Definicje

```
Δω = 0.3;
ωmin = 0.1;
ωmax = 2.0;
```

14.1.2 Butterworth

```
bmodels = Table[TransferFunctionFactor@ButterworthFilterModel[{"Lowpass", n, ω}, s],
   {n, 1, 7, 1}, {ω, ωmin, ωmax, Δω}];
DGbmodels =
  Table[TransferFunctionFactor@
    ToDiscreteTimeModel[ButterworthFilterModel[{"Lowpass", n, ω}, s], 1], {n, 1, 7, 1},
   {ω, ωmin, ωmax, Δω}];
```

14.1.3 Chebyshev 1

```
c1models = Table[TransferFunctionFactor@Chebyshev1FilterModel[{"Lowpass", n, ω}, s],
   {n, 1, 7, 1}, {ω, ωmin, ωmax, Δω}];
DGc1models =
  Table[TransferFunctionFactor@ToDiscreteTimeModel[Chebyshev1FilterModel[{"Lowpass", n, ω}, s],
     1], {n, 1, 7, 1}, {ω, ωmin, ωmax, Δω}];
```

14.1.4 Chebyshev 2

```
c2models = Table[TransferFunctionFactor@Chebyshev2FilterModel[{n, ω}, s], {n, 1, 7, 1},
   {ω, ωmin, ωmax, Δω}];
DGc2models =
  Table[TransferFunctionFactor@ToDiscreteTimeModel[Chebyshev2FilterModel[{n, ω}, s], 1],
   {n, 1, 7, 1}, {ω, ωmin, ωmax, Δω}];
```

14.1.5 Eliptic

```
emodels = Table[TransferFunctionFactor@EllipticFilterModel[{n, ω}, s], {n, 1, 7, 1},
   {ω, ωmin, ωmax, Δω}];
DGemodels = Table[TransferFunctionFactor@ToDiscreteTimeModel[EllipticFilterModel[{n, ω}, s], 1],
   {n, 1, 7, 1}, {ω, ωmin, ωmax, Δω}];
```

14.2 Dyskretyzacja na poziomie zer i biegunów

14.2.1 Wstęp

Na początek przedstawiam sposób w jaki dyskretyzowałem dotychczas, tj. mając filtr cyfrowy w postaci $\frac{a_0(z-a_1)}{(z-b_1)\,(z-b_2)}$ dyskretyzowałem liczby $a_1,\ b_1,\ \ b_2$.

Okazuje się, że nie jest najlepszy sposób dyskretyzacji by sprawdzić rzeczywiste efekty tego procesu.

Lepszy sposób prezentowany w kolejnym paragrafie, polega na dyskretyzacji końcowych współczynników filtru, a więc współczynników $c_0,\ c_1,\ d_0,\ \ d_1,\ \ d_2$ filtru $\frac{c_0 + c_1\,z}{d_0 + d_1\,z + d_2\,z^2}$ będącego wymnożoną postacią wyżej przedstawionego filtru.

Przy dyskretyzacji biegunów koniecznym zabiegiem okazało się nie dyskretyzowanie współczynnika stojącego przed całością filtra, gdyż przy niskich częstotliwościach i wysokich rzędach filtrów stawał się on bardzo mały, a po dyskretyzacji = 0. Przy tych założeniach większość filtrów okazała się stabilna przy dyskretyzacji za pomocą 6 bitów.
Okazuje się jednak, że dyskretyzacja na poziomie biegunów prowadzi do obniżenia rzędu filtra w przypadku filtrów eliptycznych (tj. zera i bieguny znajdujące się blisko siebie zaczynają wzajemnie się skracać)

Dyskretyzacja współczynników ma bardzo podobny problem widoczny przy niskich częstotliwościach filtru Butterwortha i Chebysheva 1, z powodu znikającego licznika znika cały filtr.
Ponadto widać, iż filtry eliptyczne powyżej 3 rzędu robią się bardzo niestabilne (niektóre nie są zaznaczone na czerwono bo filtr znikł z powodów wspomnianych wyżej)

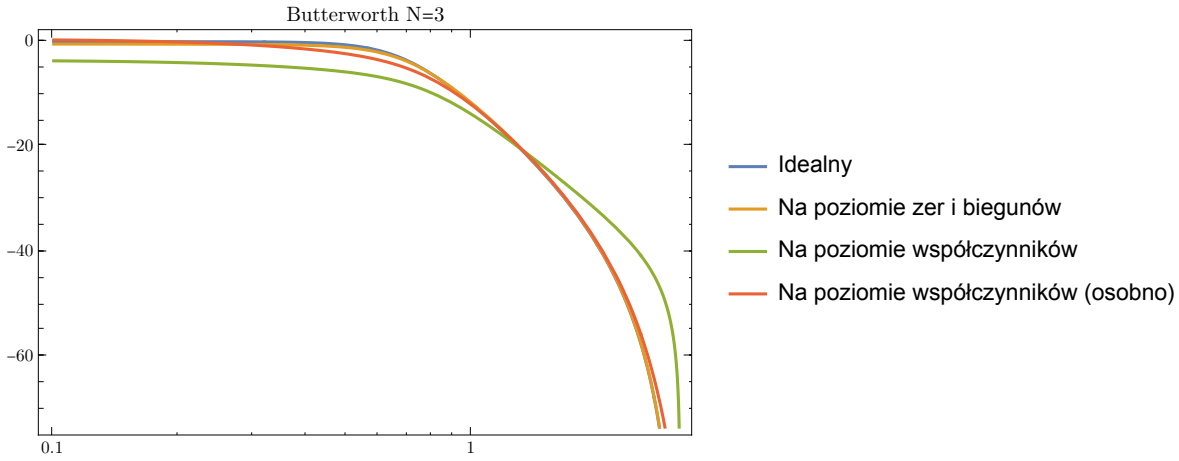
W świetle tych wyników oraz faktu, iż na urządzeniu będziemy dyskretyzować współczynniki, a nie bieguny, proponuję zastosować dyskretyzację zer i biegunów z osobna, tak by zapobiec zerowaniu się licznika.
Wykonane w celu sprawdzenia tej propozycji symulacje przedstawiam poniżej.

15 Podsumowanie

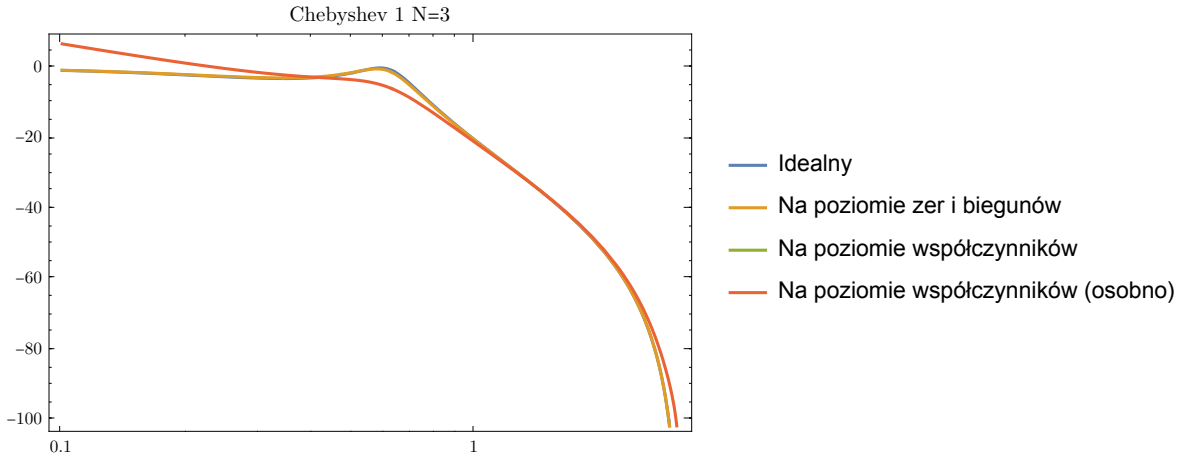
Wbrew pozorom podzielenie procesu dyskretyzacji na współczynniki licznika i mianownika nie przyniosło wielkich zmian - filtry który znikały, stały się po prostu niestabilne.
Używając Bode Plot (ilustrujący kwadrat amplitudy (dB) w zależności od ω) można zademonstrować, że lepsze odwzorowanie filtru zależy właśnie od przyjętej strategii dyskretyzacji współczynników.

Poniżej znajdują się przykładowe Bode Ploty dla filtrów Butterwortha trzeciego rzędu i częstości równej 1.

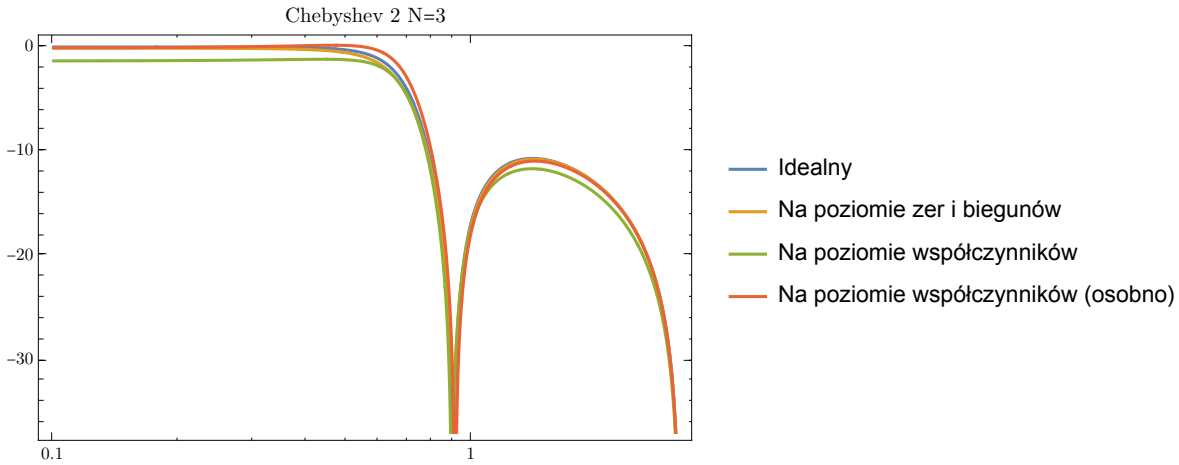
```
BodePlot[{DGbmodels[[3, 3]][e^(i ω)], DGbmodelsD[[3, 3]] /. {z -> e^(i ω)},
  DGbmodelsDc[[3, 3]] /. {z -> e^(i ω)}, DGbmodelsDc2[[3, 3]] /. {z -> e^(i ω)}}, {ω, 0.1, π},
 PlotLayout → "Magnitude",
 PlotLegends → {"Idealny", "Na poziomie zer i biegunów", "Na poziomie współczynników",
   "Na poziomie współczynników (osobno)"}, PlotLabel → "Butterworth N=3"]
```



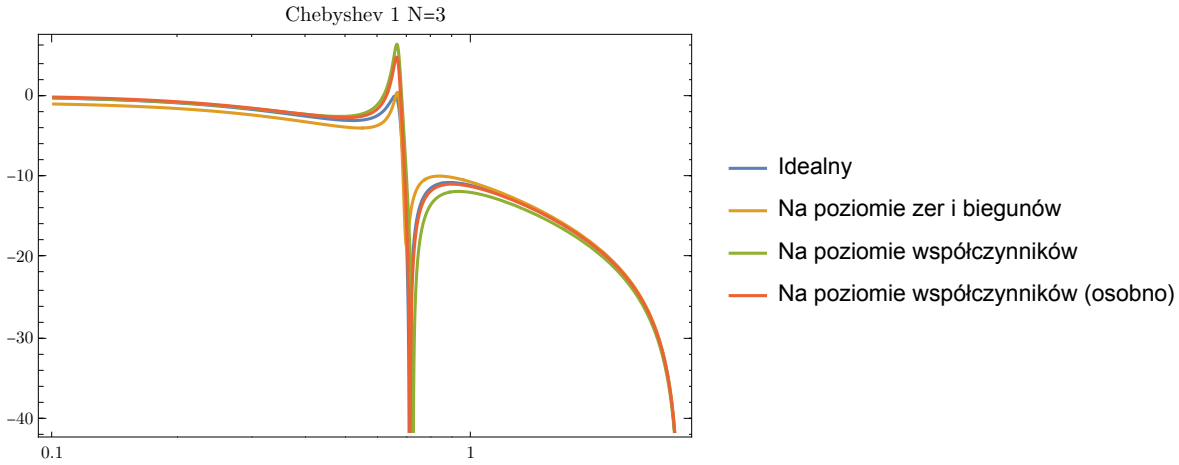

```
BodePlot[{DGc1models[[3, 3]][e^(i ω)], DGc1modelsD[[3, 3]] /. {z -> e^(i ω)},
  DGc1modelsDc[[3, 3]] /. {z -> e^(i ω)}, DGc1modelsDc2[[3, 3]] /. {z -> e^(i ω)}}, {ω, 0.1, π},
 PlotLayout → "Magnitude",
 PlotLegends → {"Idealny", "Na poziomie zer i biegunów", "Na poziomie współczynników",
   "Na poziomie współczynników (osobno)"}, PlotLabel → "Chebyshev 1 N=3"]
```



```
BodePlot[{DGc2models[[3, 3]][e^(i ω)], DGc2modelsD[[3, 3]] /. {z -> e^(i ω)},
  DGc2modelsDc[[3, 3]] /. {z -> e^(i ω)}, DGc2modelsDc2[[3, 3]] /. {z -> e^(i ω)}}, {ω, 0.1, π},
 PlotLayout → "Magnitude",
 PlotLegends → {"Idealny", "Na poziomie zer i biegunów", "Na poziomie współczynników",
   "Na poziomie współczynników (osobno)"}, PlotLabel → "Chebyshev 2 N=3"]
```




```
BodePlot[{DGemodels[[3, 3]][e^(i ω)], DGemodelsD[[3, 3]] /. {z -> e^(i ω)},
  DGemodelsDc[[3, 3]] /. {z -> e^(i ω)}, DGemodelsDc2[[3, 3]] /. {z -> e^(i ω)}}, {ω, 0.1, π},
 PlotLayout → "Magnitude",
 PlotLegends → {"Idealny", "Na poziomie zer i biegunów", "Na poziomie współczynników",
   "Na poziomie współczynników (osobno)"}, PlotLabel → "Chebyshev 1 N=3"]
```



Jeśli trzeba można wyrysować wszystkie tym poleceniem.

```
Grid@
 MapThread[BodePlot[{#1[e^(i ω)], #2 /. {z -> e^(i ω)}, #3 /. {z -> e^(i ω)}, #4 /. {z -> e^(i ω)}},
    {ω, 0.01, π}, PlotLayout → "Magnitude"] &, {DGbmodels, DGbmodelsD, DGbmodelsDc, DGbmodelsDc2},
  2]
```

```
$Aborted
```

Wybrana przeze mnie wcześniej procedura dyskretyzacji na poziomie zer i biegunów (dodatkowo wcześniej robiłem to na poziomie filtru analogowego), pomimo swoich pozornych zalet obarczona jest

jednak dodatkowym błędem, wynikającym z wymnożenia zdyskretyzowanych zer/biegunów którego należałoby by dokonać nie na liczbach rzeczywistych, lecz na liczbach n-bitowych.
Prawdopodobnie to dyskwalifikuje tę metodę.

Zatem mając dwóch kandydatów (na poziomie współczynników, i na poziomie współczynników - osobno zera i bieguny) do metody dyskretyzacji przeprowadzam symulacje dla wyższej liczby bitów.

16 Badanie stabilności filtrów cyfrowych (7 bit)

17 Badanie stabilności filtrów cyfrowych (8 bit)

18 Badanie stabilności filtrów cyfrowych (9 bit)

19 Badanie stabilności filtrów cyfrowych (10 bit)

20 Badanie stabilności filtrów cyfrowych (11 bit)

21 Badanie stabilności filtrów cyfrowych (12 bit)

22 Badanie stabilności filtrów cyfrowych (13 bit)

23 Badanie stabilności filtrów cyfrowych (14 bit)

24 Badanie stabilności filtrów cyfrowych (15 bit)

25 Badanie stabilności filtrów cyfrowych (16 bit)

26 Skompresowana wersja

Zdaje sobie sprawę z tego jak fatalnie się to czyta, zwłaszcza bez spisu treści. Dlatego postanowiłem skompresować najważniejsze wyniki dla dyskretyzacji w której wpółczynniki zer i biegunów są dyskretyzowane oddzielnie.

Poniżej znajduje się wizualizacja stabilności filtrów, zasada jest ta sama - w dół rośnie rząd, w prawo rośnie częstość graniczna.
A teraz mała incepcja - promień pierścienia jest związany z liczbą bitów. Środkowy okrąg jest 5 bitowy, każdy kolejny 3 bity więcej, okręgów jest 9, zatem ostatni okrąg reprezentuje $5 + 9 * 3 = 5 + 27 = 32$ bity. Kolor niebieski oznacza filtr stabilny, kolor pomarańczowy - niestabilny.

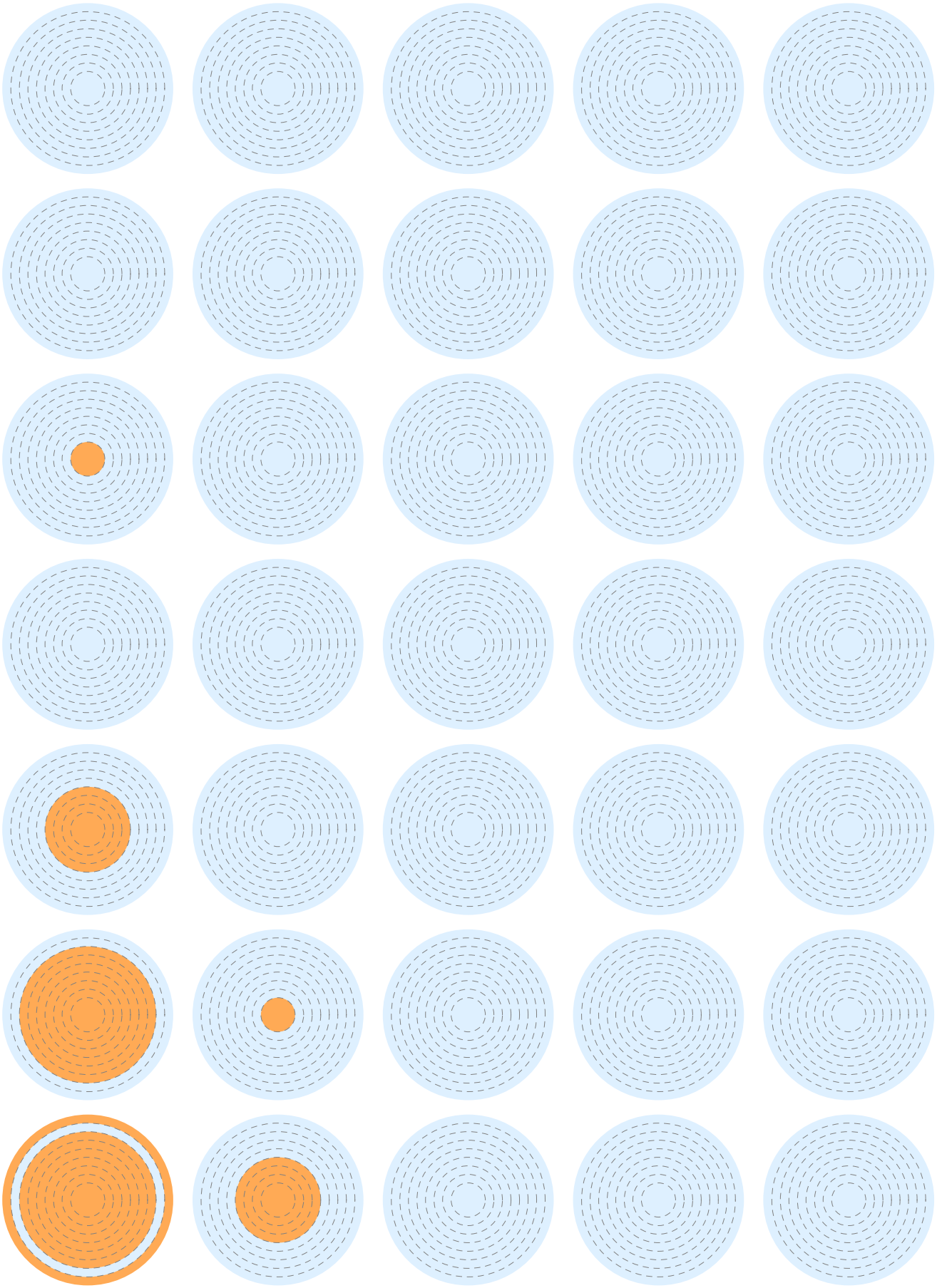
```
SP[p_] := If[Count[p, _?(Abs[#] ≥ 1 &), {1}] > 0, Lighter[Orange], LightBlue]
```

```
StableOrNot[p1_, p2_, p3_, p4_, p5_, p6_, p7_, p8_, p9_] := Graphics[{SP[p1], Disk[{0, 0}],
   EdgeForm[Directive[Dashed, Gray]], SP[p2], Disk[{0, 0}, {0.9, 0.9}],
   EdgeForm[Directive[Dashed, Gray]], SP[p3], Disk[{0, 0}, {0.8, 0.8}],
   EdgeForm[Directive[Dashed, Gray]], SP[p4], Disk[{0, 0}, {0.7, 0.7}],
   EdgeForm[Directive[Dashed, Gray]], SP[p5], Disk[{0, 0}, {0.6, 0.6}],
   EdgeForm[Directive[Dashed, Gray]], SP[p6], Disk[{0, 0}, {0.5, 0.5}],
   EdgeForm[Directive[Dashed, Gray]], SP[p6], Disk[{0, 0}, {0.4, 0.4}],
   EdgeForm[Directive[Dashed, Gray]], SP[p6], Disk[{0, 0}, {0.3, 0.3}],
   EdgeForm[Directive[Dashed, Gray]], SP[p7], Disk[{0, 0}, {0.2, 0.2}]}]
```

26.1 Butterworth

```
Allb = Table[Map[ExtractPoles[DiscretizeModelCoeffs2[#, bits]] &, DGbmodels, {2}],
   {bits, 5, 32, 3}];
```

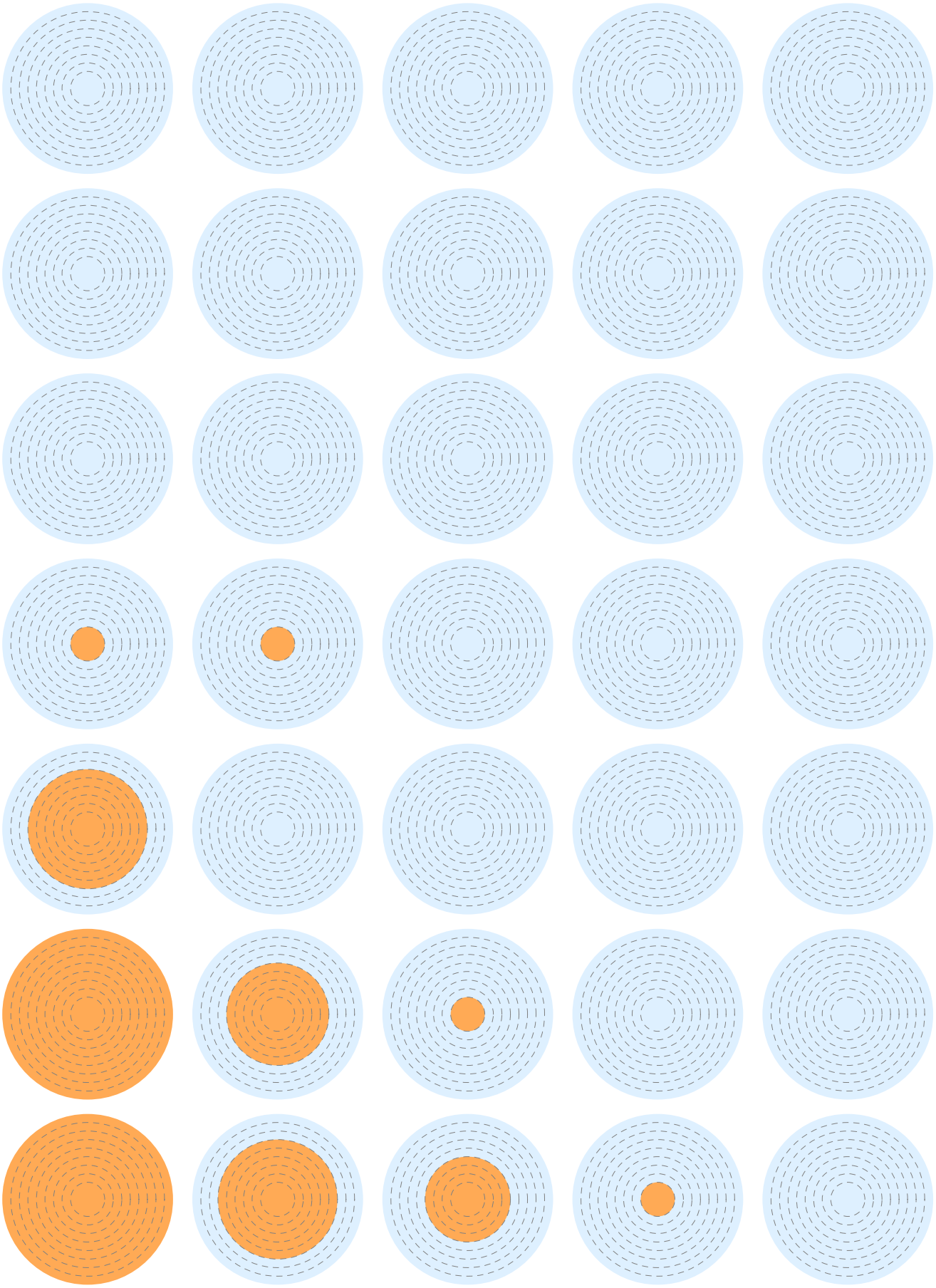
```
Grid[MapThread[StableOrNot[#9, #8, #7, #6, #5, #4, #3, #2, #1] &, Allb, 2], ItemSize → 10]
```



26.2 Chebyshev 1

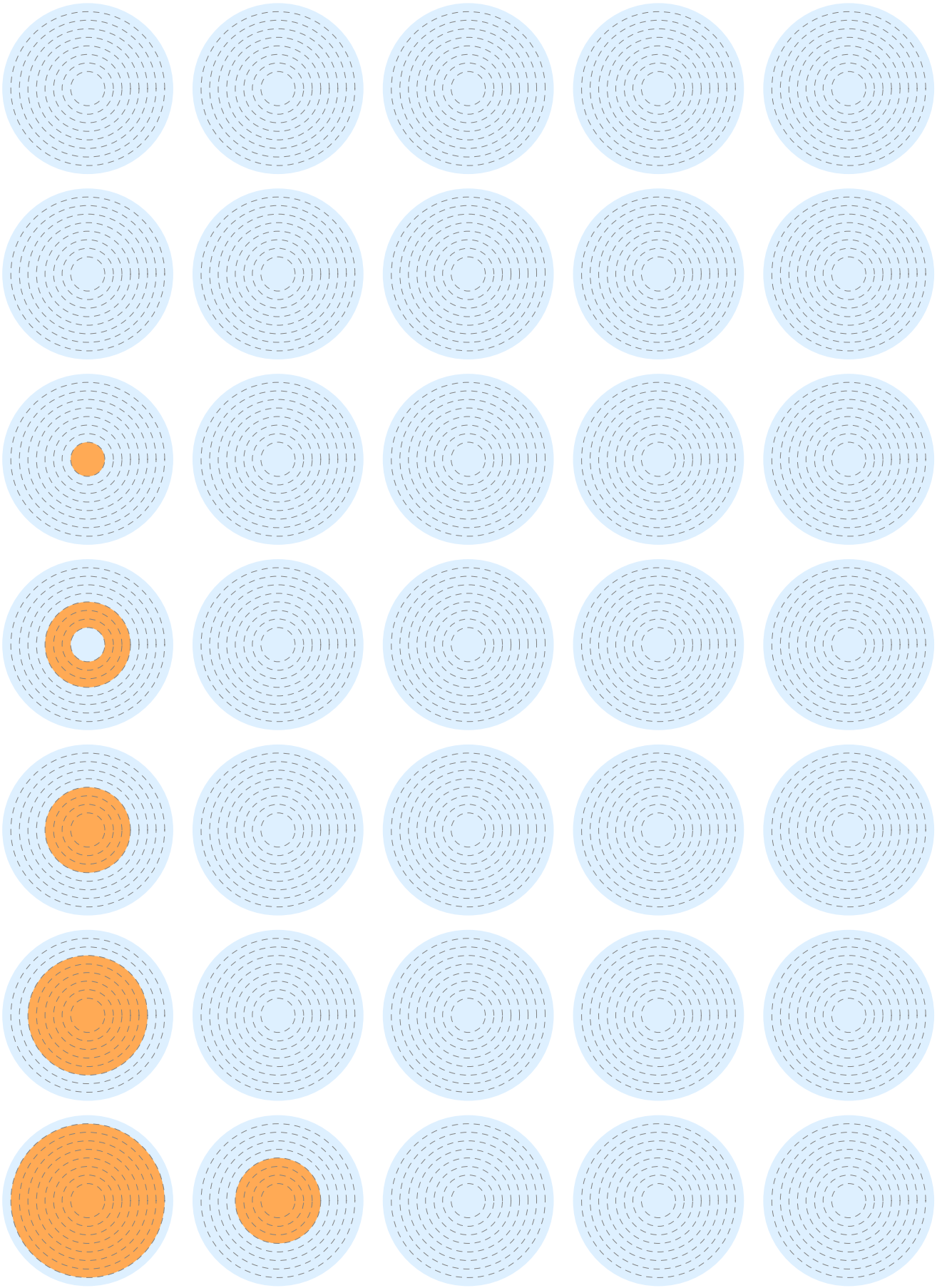
```
Allc1 = Table[Map[ExtractPoles[DiscretizeModelCoeffs2[#, bits]] &, DGc1models, {2}],
   {bits, 5, 32, 3}];

Grid[MapThread[StableOrNot[#9, #8, #7, #6, #5, #4, #3, #2, #1] &, Allc1, 2], ItemSize → 10]
```



26.3 Chebyshev 2

```
Allc2 = Table[Map[ExtractPoles[DiscretizeModelCoeffs2[#, bits]] &, DGc2models, {2}],
   {bits, 5, 32, 3}];
Grid[MapThread[StableOrNot[#9, #8, #7, #6, #5, #4, #3, #2, #1] &, Allc2, 2], ItemSize → 10]
```



26.4 Eliptyczny

```
Alle = Table[Map[ExtractPoles[DiscretizeModelCoeffs2[#, bits]] &, DGemodels, {2}],
   {bits, 5, 32, 3}];
Grid[MapThread[StableOrNot[#9, #8, #7, #6, #5, #4, #3, #2, #1] &, Alle, 2], ItemSize → 10]
```