Tworzenie i optymalizacja filtrów IIR

Raport 2 (12.11.2016)

Uniwersytet Jagielloński – Zakład Fotoniki

1 Wstęp

1.1 Metoda tworzenia filtrów IIR

Najpopularniejszą metodą tworzenia filtrów cyfrowych ze sprzężeniem zwrotnym (Infinite Impule Response – IIR) jest metoda polegająca na stworzeniu wpierw ich analogowych prototypów, a następnie przejściu (za pomocą transformaty biliniowej) do ich cyfrowych odpowiedników.
Wśród powszechnie znanych typów filtrów analogowych znajdują się:

- filtry Butterworth'a
- filtry Czebyszewa I/II
- filtry Eliptyczne

1.2 Badanie stabilności filtrów - porównanie filtrów analogowych i cyfrowych

Układ analogowy jet stabilny, jeżeli bieguny jego transmitancji leżą w lewej półpłaszczyźnie płaszczyzny zespolonej.

Układ dyskretny jest stabilny, jeżeli bieguny jego transmitancji leżą wewnątrz okręgu jednostkowego na płaszczyźnie zespolonej.

Dla układu analogowego osią częstotliwości jest oś urojona $s = j\omega$.
Dla układu dyskretnego częstotliwość zmiania się po okręgu jednostkowym $z = e^{j\Omega}$.

W przypadku biegunów na osi (dla analogowych) lub na okręgu (dla dyskretnych) stabilność jest warunkowa, a układ odpowiada oscylacją.

Część I

1 Projektowanie filtrów analogowych Butterwortha i Czebyszewa

Podczas projektowania korzystamy z oznaczeń przedstawionych na rysunku poniżej

- Częstotliwość końca pasma przepustowego (f_P) w dB
- Największe dopuszczalne tłumienie w paśmie przepustowym (A_p) dla filtru Czebyszewa ten parametr określa jednocześnie dopuszczalne zafalowanie charakterystyki amplitudowej w paśmie przepustowym (w kHz)
- Częstotliwość początku pasma zaporowego (f_R) w dB
- Wymagane minimalne tłumienie w paśmie zaporowym (A_R) (w kHz)

```
Ap = 0.5;
fp = 3;
Ar = 20;
fr = 7;
```

1.1 Obliczenie rzędu filtra

1.1.1 Współczynnik selektywności k

$$k = \frac{f_P}{f_R}$$

```
k[fp_, fr_] := fp / fr;
```

1.1.2 Współczynnik dyskryminacji d

$$d = \sqrt{\frac{10^{0.1\,A_P} - 1}{10^{0.1\,A_R} - 1}}$$

```
d[Ap_, Ar_] := Sqrt[(10^(0.1 Ap) - 1)/(10^(0.1 Ar) - 1)];
```

Rząd filtra Butterwortha oblicza się ze współczynników k i d

$$n = \frac{|\log d|}{|\log k|} \tag{1.1}$$

```
nbutt[d_, k_] := Abs[Log[d]] / Abs[Log[k]];
```

```
nb = nbutt[d[Ap, Ar], k[fp, fr]]
```

3.95298

Rząd filtra Czebyszewa określa się z podobnego wzoru:

$$n = \frac{\operatorname{arc\,cosh}\left(\frac{1}{d}\right)}{\operatorname{arc\,cosh}\left(\frac{1}{k}\right)} \tag{1.2}$$

```
nczeb[d_, k_] := ArcCos[1/d]/ArcCos[1/k];
```

```
nc = nczeb[d[Ap, Ar], k[fp, fr]]
```

2.71107 + 0. i

Rząd filtru nie może być ułamkowy, więc zaokrąglamy w górę.

```
nb = Ceiling[nb]
nc = Ceiling[nc]
```

4

3

Rząd filtr Czebyszewa będzie zawsze mniejszy lub równy rzędowi filtru Butterwortha.

1.2 Częstotliwość graniczna

W dolnoprzepustowym filtrze Czebyszewa jest to częstotliwość, dla której tłumienie filtru nie przekracza założonych zafalowań charakterystyki. A więc poprostu f_P

$$f_g = f_P \tag{1.3}$$

```
fgc = fp;
```

Dla filtru Butterwortha, częstotliwość graniczna to taka dla której następuje spadek wzmocnienia o 3 dB względem sygnału stałego (o częstotliowści 0). Jest ona większa od f_P jeżeli A_P jest mniejsze niż 3 dB. Można ją określić ze wzoru

$$f_g = \frac{f_P}{\left(10^{0.1\,A_P} - 1\right)^{\frac{1}{2\,n}}} \tag{1.4}$$

```
fg[fp_, Ap_, n_] := fp/(10^(0.1 Ap) - 1)^(1/(2 n));
```

```
fgb = fg[fp, Ap, 4]
```

3.90228

1.3 Filtr prototypowy

Definition 1.1 Filtr prototypowy ≡ filtr anaologowy którego częstotliwość graniczna wynosi 1 Hz.

1.3.1 Butterworth

Kradrat charakterystyki amplitudowej definiuje się następująco

$$|H(j\omega)|^2 = \frac{1}{1+(j\omega\,/\,j\omega_{3\,\mathrm{dB}})^{2\,N}} \tag{1.5}$$

$$|H(j\omega)| = \sqrt{\frac{1}{1+(j\omega\,/\,j\omega_{3\,\mathrm{dB}})^{2\,N}}} \tag{1.6}$$

gdzie $\omega_{3\,\mathrm{dB}}$ jest pulsacją, dla której charakterystyka amplitudowa opada do poziomu $\frac{1}{\sqrt{2}}$ (3 dB w skali decybelowej $20\log_{10}\left(1/\sqrt{2}\right) = -3$ dB. Rozważając zera mianownika:

$$1+(s\,/\,j\omega_{3\,\mathrm{dB}})^{2\,N} = 0$$

Otrzymujemy bieguny transmitancji filtra Butterwortha:

```
sol = Solve[1 + (s / (ⅈ ω))^(2 NN) == 0 && NN > 0, s, Complexes]
```

$\left\{\left\{s \to \text{ConditionalExpression}\left[i\, e^{\frac{i(\pi+2\pi C[1])}{2\,NN}}\omega,\ (C[1] \in \text{Integers} \,\&\&\, -1+2\,NN-2\,C[1] \geq 0 \,\&\&\, C[1] \geq 0) \,||\, (C[1] \in \text{Integers} \,\&\&\, C[1] \leq -1 \,\&\&\, 1+2\,NN+2\,C[1] > 0)\right]\right\}\right\}$

```
sol[[1, 1, 2, 1]]
```

$i\, e^{\frac{i(\pi+2\pi C[1])}{2\,NN}}\omega$

$$\omega_{3\mathrm{dB}}\, e^{j\,(\pi/(2\,N))\left(2\,k+N-1\right)} \tag{1.7}$$

Do filtra wybieramy tylko te znajdujące się w lewej półpłaszczyźnie

```
roots[k_, N_] := Exp[ⅈ (π / (2 N)) (2 k + N - 1)];
NN = 2;
rootsbutt = N@Table[roots[i, NN], {i, 0, NN}];
selrootbutt = s - Select[rootsbutt, Re[#] <= 0 &];
1 / Times[selrootbutt /. {List → Sequence}]
```

$$\frac{1}{((0.707107 - 0.707107\,i) + s)\,((0.707107 + 0.707107\,i) + s)}$$

Z równania (1.6) otrzymujemy:

$$20\log_{10}\left(1\Big/\sqrt{1+(\omega_p\,/\,\omega_{3\mathrm{dB}})^{2\,N}}\right) = -A_P \tag{1.8}$$

$$20\log_{10}\left(1\Big/\sqrt{1+(\omega_s\,/\,\omega_{3\mathrm{dB}})^{2\,N}}\right) = -A_R \tag{1.9}$$

Skąd rząd filtru:

$$N = \frac{1}{2}\log_{10}\left(\frac{10^{A_P/10}-1}{10^{A_R/10}-1}\right)\Big/\log_{10}\left(\frac{\omega_p}{\omega_s}\right) \tag{1.10}$$

$$\omega_{3\mathrm{dB}} = \frac{\omega_{\mathrm{pass}}}{\left(10^{A_P/10}-1\right)^{1/(2N)}} \tag{1.11}$$

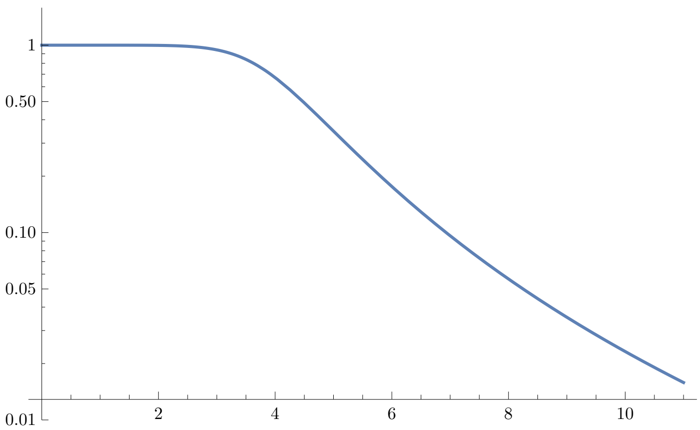
1.4 Całość dla filtru Butterworth

```
Ap = 0.5;
fp = 3;
As = 20;
fs = 7;

NButterworth[Ap_, As_, fp_, fs_] := Ceiling[1/2 Log10[(10^(Ap/10) - 1)/(10^(As/10) - 1)] / Log10[fp / fs]];

f3dB[fp_, Ap_, NN_] := fp/(10^(Ap/10) - 1)^(1/(2 NN));(**pulsacja**)

NN = NButterworth[Ap, As, fp, fs]
f3dB[fp, Ap, NN]
```

```
4

3.90228
```

```
roots[k_, N_] := f3dB[fp, Ap, NN] Exp[ⅈ (π / (2 N)) (2 k + N - 1)];
rootsbutt = N@Table[roots[i, NN], {i, 0, NN}];
selrootbutt = s - Select[rootsbutt, Re[#] <= 0 &];
ButterworthFilter[ω_] := f3dB[fp, Ap, NN]^NN / Times[selrootbutt /. {List → Sequence}] /. {s → ⅈω}

ButterworthFilter[ω]
LogPlot[Abs[ButterworthFilter[ω]], {ω, 0, 11}, PlotRange → Full]
```

```
231.885 / (((1.49334 - 3.60523 ⅈ) + ⅈ ω)
    ((1.49334 + 3.60523 ⅈ) + ⅈ ω) ((3.60523 - 1.49334 ⅈ) + ⅈ ω) ((3.60523 + 1.49334 ⅈ) + ⅈ ω))
```



1.5 Filtr Czebyszewa

$$|H(j\omega)|^2 = \frac{1}{1 + \epsilon^2\ V_N^2(i\omega / i\omega_{3\text{dB}})} \tag{1.12}$$

Gdzie

$$V_N(x) = \begin{cases} \cos(N \arccos(x)) & |x| \le 1 \\ \cosh(N \operatorname{arccosh}(x)) & |x| > 1 \end{cases} \tag{1.13}$$

Jest to filtr I typu. Charakterystyka jest równomiernie falista w paśmie przepustowym.
Amplituda zafalowania jest regulowana parametrem ϵ.
W paśmie zaporowym charakterystyka opada monotonicznie

Dla filtru typu II jest odwrotnie. Charakterystyka jest płaska w paśmie przepustowym i równomiernie falista w paśmie zaporowym.
Parametr ϵ reguluje amplitudę zafalowania charakterystyki amplitudowej w paśmie zaporowym.

$$|H(j\omega)|^2 = \frac{1}{1 + \frac{1}{\epsilon^2\ V_N^2(i\omega_{3\text{dB}}/i\omega)}} \tag{1.14}$$

1.6 Filtr eliptyczny

Do aproksymacji idealnej, prostokątnej charakterystyki częstotliwościowej w przypadku filtra eliptycznego stosuje się eliptyczną funkcję Jacobiego.
Charakterystyka amplitudowa filtra eliptycznego jest równomiernie falista w paśmie przepustowym i zaporowym. Filtr eliptyczny opisują 4 parametry: N – rząd filtra, ω - krawędź pasma przepustowego, A_P oraz A_R.

1.7 Transformacja częstotliwości

Po zaprojektowaniu analogowego filtru dolnoprzepustowego możemy utworzyć inne filtry korzystając z następującej tabeli:

Low pass	$s' = s / \omega_0$
High pass	$s' = \omega_0 / s$
Band pass	$s' = \frac{\omega_0}{\Delta\omega} \frac{(s/\omega_0)^2+1}{s/\omega_0}$
Band stop	$s' = \frac{\Delta\omega}{\omega_0} \frac{s/\omega_0}{(s/\omega_0)^2+1}$

where $\Delta\omega = \omega_2 - \omega_1$ and $\omega_0 = \sqrt{\omega_1\ \omega_2}$

2 Przykładowy projekt filtru dolnoprzepustowego

Zakładamy wartości parametrów filtru:

- częstotliwość próbkowania $f_{\text{pr}} = 100\ \text{Hz}$

- krawędź pasma przepustowego $f_1 = 20\,\mathrm{Hz}$
- krawędź pasma zaporowego $f_2 = 25\,\mathrm{Hz}$
- nieliniowość charakterystyki w paśmie przepustowym $A_p = 3\,\mathrm{dB}$
- nieliniowość charakterystyki w paśmie zaporowym $A_s = 30\,\mathrm{dB}$

Przeliczenie częstotliwości filtra cyfrowego f_1 i f_2 na pulsację Ω w radianach:

$$\Omega_1 = 2\,\pi\,\frac{f_1}{f_{\mathrm{pr}}} = 2\,\pi\,\frac{20}{100} = 1.2566\,\mathrm{rad}$$

$$\Omega_2 = 2\,\pi\,\frac{f_2}{f_{\mathrm{pr}}} = 2\,\pi\,\frac{25}{100} = 1.5708\,\mathrm{rad}$$

```
A1 = 3;
f1 = 20;
A2 = 30;
f2 = 25;
fp = 100;
```

```
Ω1 = N@2 π f1/fp
Ω2 = N@2 π f2/fp
```

```
1.25664

1.5708
```

2.1 Krawędź pasma filtru analogowego

$$\omega_1 = \frac{2}{2\,\Delta\mathrm{t}}\,\tan(\Omega_1\,/\,2) = \frac{2}{1\,/\,100}\,\tan(1.2566\,/\,2) = 145.3085\,\mathrm{rad}\,/\,s$$

$$\omega_2 = 200\,\mathrm{rad}\,/\,s$$

```
ω1 = 2/(1/fp) Tan[Ω1/2]
ω2 = 2/(1/fp) Tan[Ω2/2]
fa1 = 1/(2 π) ω1;
fa2 = 1/(2 π) ω2;
```

```
145.309

200.
```

2.2 Określenie parametrów filtru analogowego na podstawie podanych wymagań

```
NButterworth[Ap_, As_, fp_, fs_] := Ceiling[1/2 Log10[(10^(Ap/10) - 1)/(10^(As/10) - 1)] / Log10[fp/fs]];
```

```
f3dB[fp_, Ap_, NN_] := fp / (10^(Ap/10) - 1)^(1/(2 NN)); (**pulsacja**)

NN = NButterworth[A1, A2, fa1, fa2]
N@f3dB[ω1, A1, NN]
```

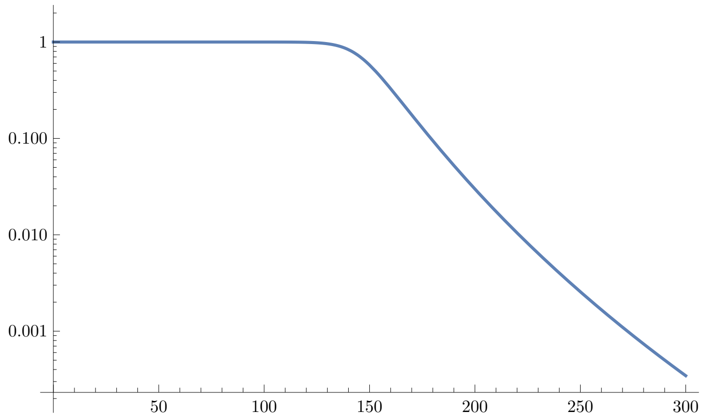
```
11
```

```
145.34
```

```
roots[k_, N_] := f3dB[ω1, A1, NN] Exp[ⅈ (π / (2 N)) (2 k + N - 1)];
rootsbutt = N@Table[roots[i, NN], {i, 0, NN}];
selrootbutt = s - Select[rootsbutt, Re[#] <= 0 &];
ButterworthFilter[ω_] := f3dB[ω1, A1, NN]^NN / Times[selrootbutt /. {List → Sequence}]
```

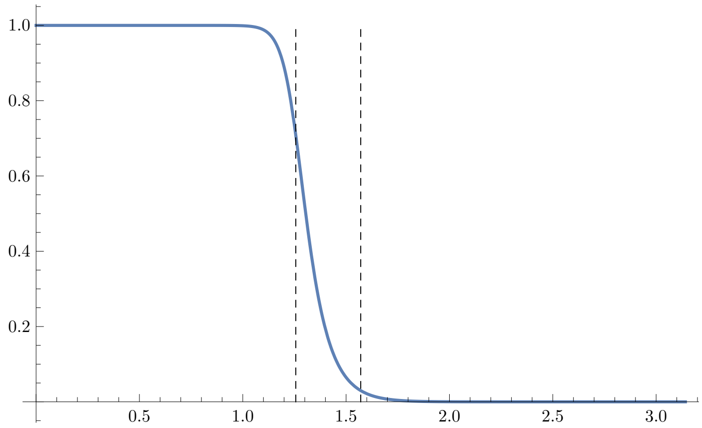
2.3 Filtr analogowy

```
LogPlot[{Abs[ButterworthFilter[s] /. {s → ⅈ ω}]}, {ω, 0, 300}, PlotRange → Full]
```



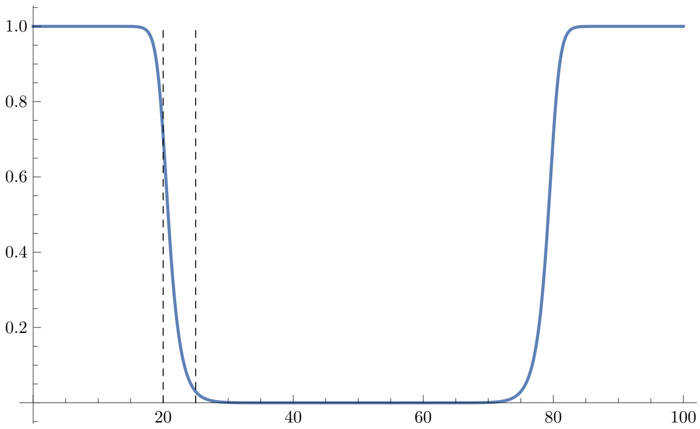
2.4 Przejście na filtr cyfrowy (w dziedzienie pulsacji/częstości)

```
Plot[Abs[ButterworthFilter[s] /. {s → 2 fp (1 - z^-1)/(1 + z^-1)} /. {z → ⅇ^(ⅈ Ω)}], {Ω, 0, π}, PlotRange → Full,
 Epilog → {Dashing[0.01], Line[{{Ω1, 0}, {Ω1, 1}}], Line[{{Ω2, 0}, {Ω2, 1}}]}]
```



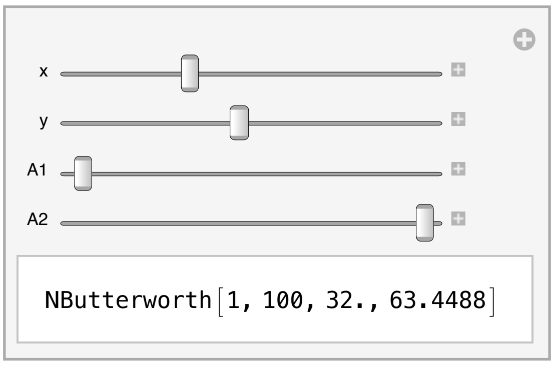
2.5 Filtr cyfrowy w dziedzinie częstotliwości

```
Plot[Abs[ButterworthFilter[s] /. {s → 2 fp (1 - z^-1)/(1 + z^-1)} /. {z → ⅇ^(ⅈ Ω)} /. {Ω → f (2 π) / fp}],
 {f, 0, 100}, PlotRange → Full,
 Epilog → {Dashing[0.01], Line[{{f1, 0}, {f1, 1}}], Line[{{f2, 0}, {f2, 1}}]}]
```



3 Badanie reakcji filtru Butterworth na zmianę częstotliwości odcięcia

```
Manipulate[NButterworth[A1, A2, x, y], {x, 0, 100}, {y, x, 100}, {A1, 0, 100, 1},
 {A2, 0, 100, 1}]
```



Obserwacja: im częstotliwości bliższe sobie, a za tem im bardziej stromy filtr tym wyższy rząd filtru Butterworth'a jest wymagany.

Oprócz tego dla:
$(2.2,\ 2.3) \to N = 78$
$(22.2,\ 22.3) \to N = 769$
$(82.2,\ 82.3) \to N = 2843$
a więc rząd filtru rośnie także z częstotliwością przy stałej różnicy $f_2 - f_1$.

Cześć II

1 Badanie błędów dyskretyzacji filtrów cyfrowych

1.1 Definicje

In[9]:=
```
Δω = 0.3;
ωmin = 0.1;
ωmax = 1.5;
bity = 6;
```

1.1.1 Funkcje dyskretyzujące

In[13]:=
```
DiscretizeList[x_, max_, bits_] :=
  If[ # ≥ 0 , 1, -1] Round[ (Abs[#] / max) (Power[2, bits - 1] - 1)] max / (Power[2, bits - 1] - 1)  & /@ x;
DiscretizeComplexList[x_, y_, bits_] := Module[{xD, yD, coeffD, max},
    If[y == {}, {},
     max = Max[Abs[Join[Re@x, Im@x, Re@y, Im@y]]];
     xD = DiscretizeList[Re@x, max, bits] + ⅈ DiscretizeList[Im@x, max, bits];
     yD = DiscretizeList[Re@y, max, bits] + ⅈ DiscretizeList[Im@y, max, bits];
     {xD, yD}]
   ];
DiscretizeModelCoeffs[tf_, bits_] := Module[{zeros, poles, zerosD, polesD},
      zeros = CoefficientList[Numerator[TransferFunctionExpand[tf][z]][[1, 1]], z];
    poles = CoefficientList[Denominator[TransferFunctionExpand[tf][z]][[1, 1]], z];
    {zerosD, polesD} = DiscretizeComplexList[zeros, poles, bits];
      Chop[FromDigits[Reverse[zerosD], z] / (FromDigits[Reverse[polesD], z])]
   ];
```

1.1.2 Funkcja błędu

```
CalcError[filtr_, filtrD_] :=
  Sqrt[NIntegrate[Abs[filtr - filtrD]^2, {Ω, 0, π}, AccuracyGoal → 5, MaxRecursion → 20] /
    NIntegrate[Abs[filtr]^2, {Ω, 0, π}, AccuracyGoal → 5, MaxRecursion → 20]];
```

Szybsza wersja:

In[16]:=
```
CalcError[filtr_, filtrD_] :=
  Sqrt[Total[Abs[Table[filtr, {Ω, 0, π, 0.001}] - Table[filtrD, {Ω, 0, π, 0.001}]]^2] /
    Total[Abs[Table[filtr, {Ω, 0, π, 0.001}]]^2]];
```

1.2 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, $\Delta\omega$}

1.2.1 Butterworth

In[17]:=
```
bmodels = Table[{n, ω, TransferFunctionFactor@ButterworthFilterModel[{"Lowpass" , n, ω}, s]},
    {n, 1, 7, 1}, {ω, ωmin, ωmax, Δω}];
DGbmodels =
  Table[
   {n, ω, TransferFunctionFactor@
     ToDiscreteTimeModel[ButterworthFilterModel[{"Lowpass" , n, ω}, s], 1]}, {n, 1, 7, 1},
   {ω, ωmin, ωmax, Δω}];
```

1.2.2 Chebyshev 1

In[52]:=
```
c1models = Table[{n, ω, TransferFunctionFactor@Chebyshev1FilterModel[{"Lowpass" , n, ω}, s]},
    {n, 1, 7, 1}, {ω, ωmin, ωmax, Δω}];
DGc1models =
  Table[
   {n, ω, TransferFunctionFactor@
     ToDiscreteTimeModel[Chebyshev1FilterModel[{"Lowpass" , n, ω}, s], 1]}, {n, 1, 7, 1},
   {ω, ωmin, ωmax, Δω}];
```

1.2.3 Chebyshev 2

In[54]:=
```
c2models = Table[{n, ω, TransferFunctionFactor@Chebyshev2FilterModel[{n, ω}, s]},
    {n, 1, 7, 1}, {ω, ωmin, ωmax, Δω}];
DGc2models =
  Table[{n, ω, TransferFunctionFactor@ToDiscreteTimeModel[Chebyshev2FilterModel[{n, ω}, s], 1]},
   {n, 1, 7, 1}, {ω, ωmin, ωmax, Δω}];
```

1.2.4 Eliptic

In[56]:=
```
emodels = Table[{n, ω, TransferFunctionFactor@EllipticFilterModel[{n, ω}, s]},
    {n, 1, 7, 1}, {ω, ωmin, ωmax, Δω}];
DGemodels =
  Table[{n, ω, TransferFunctionFactor@ToDiscreteTimeModel[EllipticFilterModel[{n, ω}, s], 1]},
   {n, 1, 7, 1}, {ω, ωmin, ωmax, Δω}];
```

1.3 Dyskretyzacja

In[58]:=
```
DGbmodelsDc2 = Map[{#[[1]], #[[2]], DiscretizeModelCoeffs[#[[3]], bity]} &, DGbmodels, {2}];
```

In[59]:=
```
DGc1modelsDc2 = Map[{#[[1]], #[[2]], DiscretizeModelCoeffs[#[[3]], bity]} &, DGc1models, {2}];
```

In[60]:=
```
DGc2modelsDc2 = Map[{#[[1]], #[[2]], DiscretizeModelCoeffs[#[[3]], bity]} &, DGc2models, {2}];
```

In[61]:=
```
DGemodelsDc2 = Map[{#[[1]], #[[2]], DiscretizeModelCoeffs[#[[3]], bity]} &, DGemodels, {2}];
```


1.4 Wyniki

In[29]:=
```
NicePlot[modelA_, modelB_, Nind_, ωind_] :=
 Plot[{Abs[modelA[[Nind, ωind]][[3]][ⅇ^(ⅈ ω)]]^2, Abs[(modelB[[Nind, ωind]][[3]])]^2 /. {z -> ⅇ^(ⅈ ω)}},
  {ω, 0, π}, AxesLabel → {ω, None}, GridLines → Automatic, AspectRatio → 1/3,
  PlotRange → Full, PlotLegends → {"Przed dyskretyzacją", "Po dyskretyzacji"},
  PlotLabel → "Przykładowy efekt dyskretyzacji dla N=" <> ToString[modelA[[Nind, ωind]][[1]]] <>
    ", ω=" <> ToString[modelA[[Nind, ωind]][[2]]] <> " i " <> ToString[bity] <> " bitów" ]
```

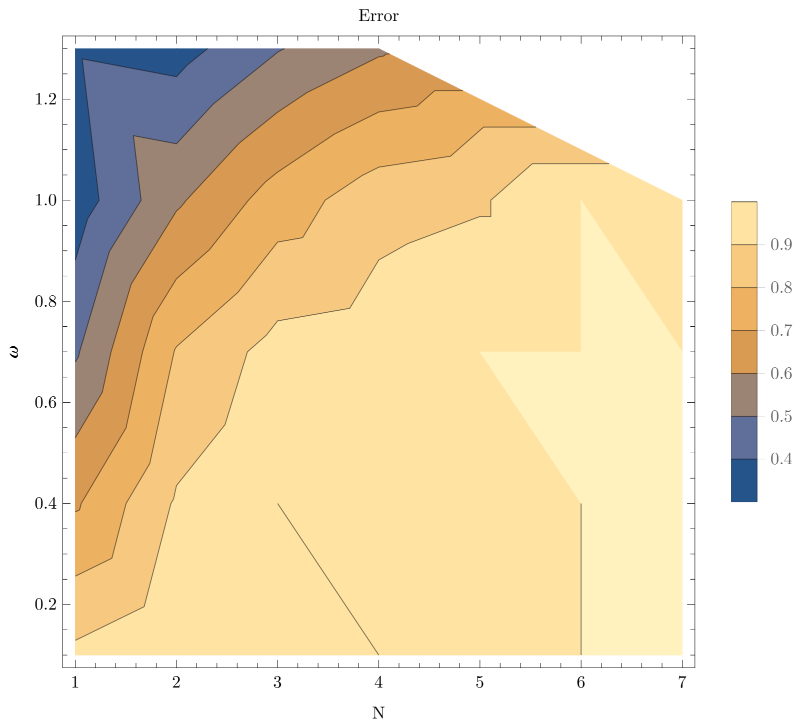
1.4.1 Butterworth

Funkcje błędu w zależności od rzędu filtra i częstości granicznej:

In[34]:=
```
errors =
  Flatten[
   MapThread[
    {#1[[1]], #1[[2]], First@Flatten@CalcError[Abs@#1[[3]][ⅇ^(ⅈ Ω)], Abs@#2[[3]] /. {z -> I Ω}]} &,
    {DGbmodels, DGbmodelsDc2}, 2], 1];
```

In[124]:=
```
ListContourPlot[errors, FrameLabel → {"N", "ω", "Error"}, PlotRange → Full,
 PlotLegends → Automatic, LabelStyle → Directive[Bold]]
```

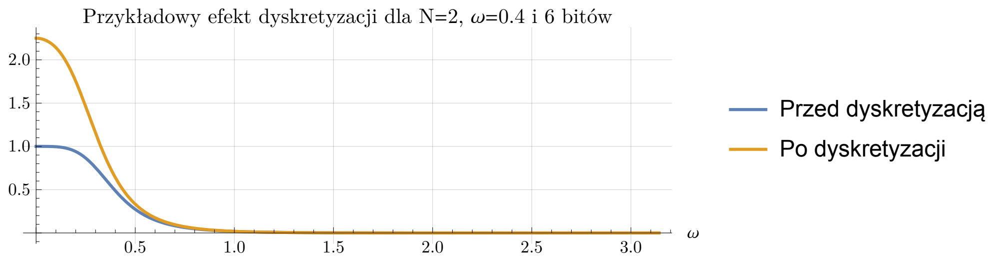
Out[124]=



Poniżej dwa przykłady:

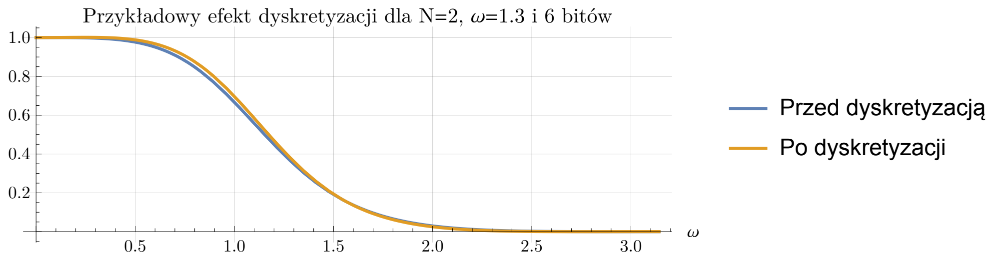
In[130]:=
```
NicePlot[DGbmodels, DGbmodelsDc2, 2, 2]
```

Out[130]=



In[126]:=
```
NicePlot[DGbmodels, DGbmodelsDc2, 2, 5]
```

Out[126]=



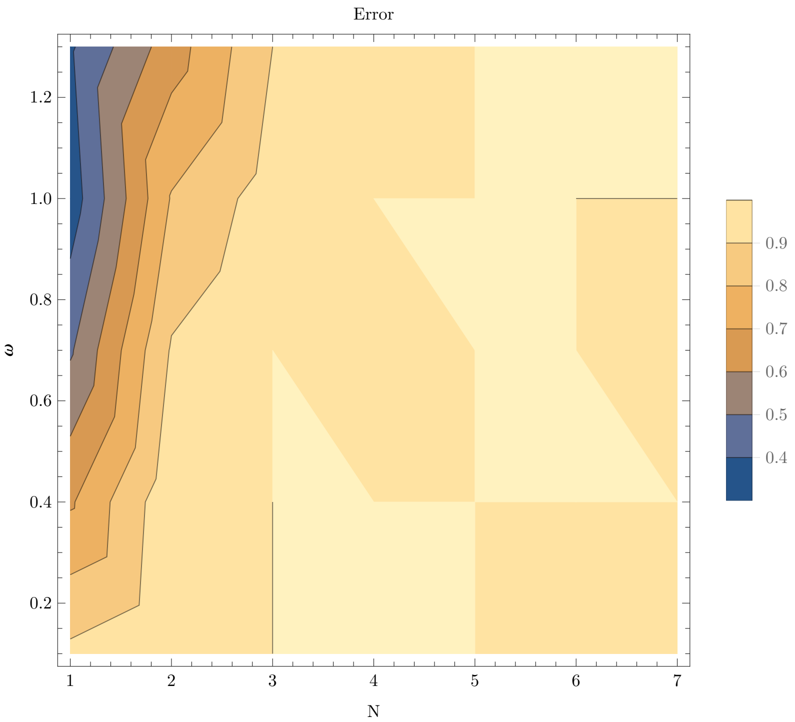
1.4.2 Chebyshev 1

Funkcje błędu w zależności od rzędu filtra i częstości granicznej:

In[62]:=
```
errors =
  Flatten[
   MapThread[
    {#1[[1]], #1[[2]], First@Flatten@CalcError[Abs[#1[[3]][ⅇ^(ⅈ Ω)]], Abs[#2[[3]]] /. {z -> I Ω}]} &,
    {DGc1models, DGc1modelsDc2}, 2], 1];
```


In[65]:=
```
ListContourPlot[errors, FrameLabel → {"N", "ω", "Error"}, PlotRange → Full,
 PlotLegends → Automatic, LabelStyle → Directive[Bold]]
```

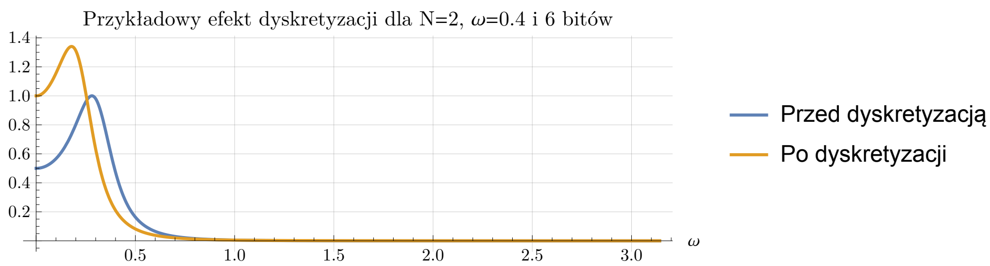
Out[65]=



Poniżej dwa przykłady:

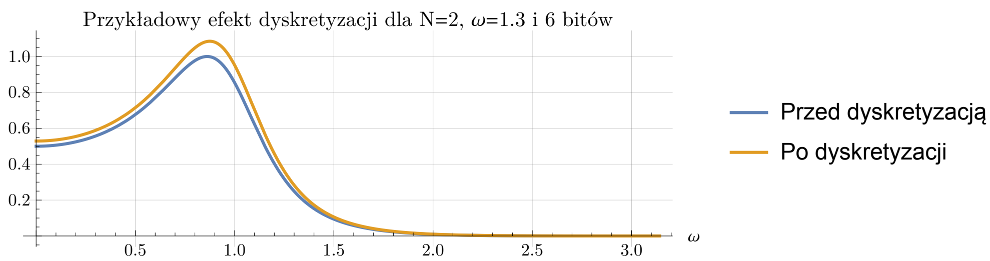
In[70]:=
```
NicePlot[DGc1models, DGc1modelsDc2, 2, 2]
```

Out[70]=



In[71]:=
```
NicePlot[DGc1models, DGc1modelsDc2, 2, 5]
```

Out[71]=

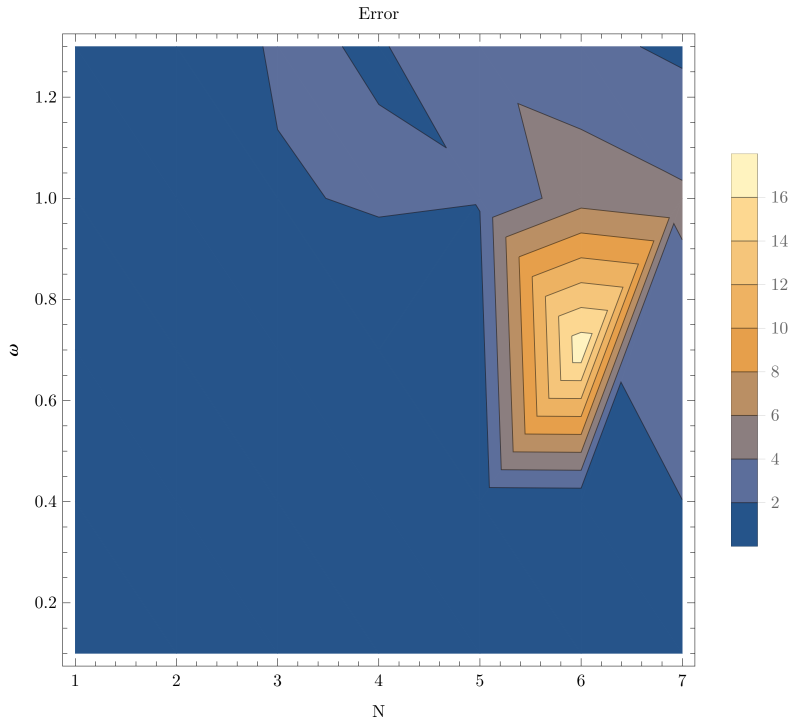


1.4.3 Chebyshev 2

Funkcje błędu w zależności od rzędu filtra i częstości granicznej:

```
In[74]:= errors =
   Flatten[
    MapThread[
     {#1[[1]], #1[[2]], First@Flatten@CalcError[Abs@#1[[3]][ⅇ^(ⅈ Ω)], Abs@#2[[3]] /. {z -> I Ω}]} &,
     {DGc2models, DGc2modelsDc2}, 2], 1];

In[67]:= ListContourPlot[errors, FrameLabel → {"N", "ω", "Error"}, PlotRange → Full,
  PlotLegends → Automatic, LabelStyle → Directive[Bold]]
```

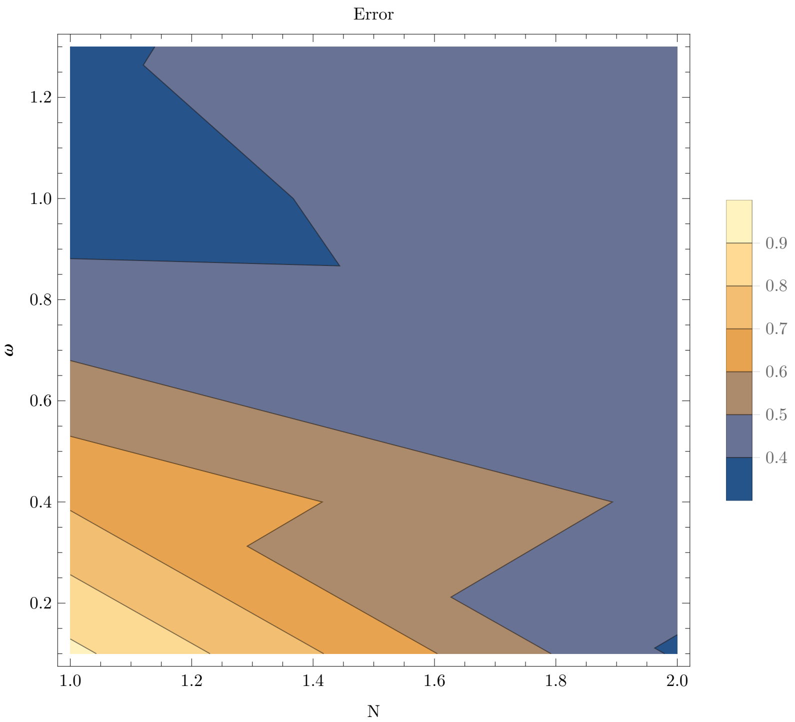
Out[67]=



Powiększam fragment w okolicy małych częstości gdzie błędy są mniejsze od 1.

In[79]:=
```
ListContourPlot[errors[[1 ;; 10]], FrameLabel → {"N", "ω", "Error"}, PlotRange → Full,
 PlotLegends → Automatic, LabelStyle → Directive[Bold]]
```

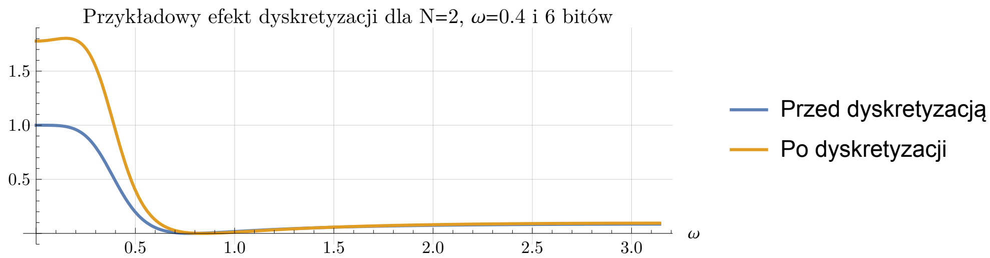
Out[79]=



Poniżej dwa przykłady:

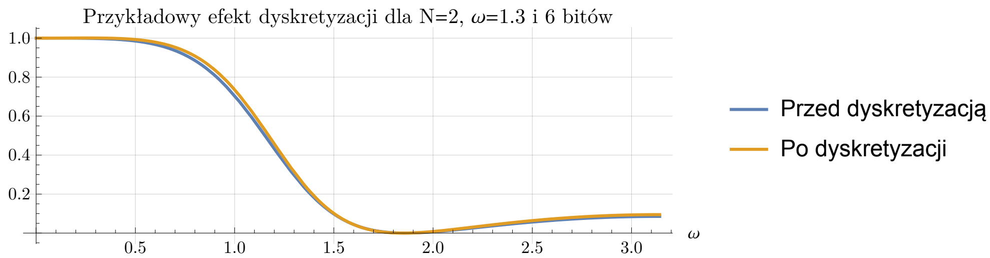
In[72]:=
```
NicePlot[DGc2models, DGc2modelsDc2, 2, 2]
```

Out[72]=



In[73]:=
```
NicePlot[DGc2models, DGc2modelsDc2, 2, 5]
```

Out[73]=



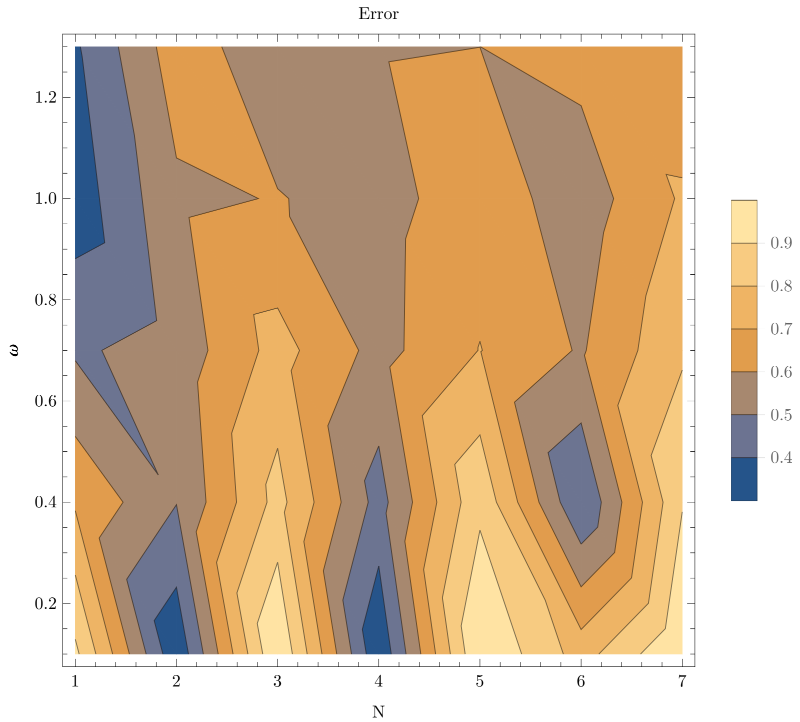
1.4.4 Eliptic

Funkcje błędu w zależności od rzędu filtra i częstości granicznej:

In[80]:=
```
errors =
  Flatten[
   MapThread[
    {#1[[1]], #1[[2]], First@Flatten@CalcError[Abs@#1[[3]][ⅇ^(ⅈ Ω)], Abs@#2[[3]] /. {z -> I Ω}]} &,
    {DGemodels, DGemodelsDc2}, 2], 1];
```

In[69]:=
```
ListContourPlot[errors, FrameLabel → {"N", "ω", "Error"}, PlotRange → Full,
 PlotLegends → Automatic, LabelStyle → Directive[Bold]]
```

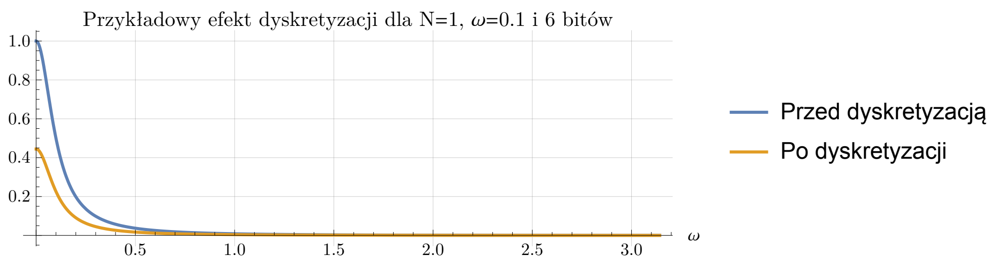
Out[69]=



Poniżej dwa przykłady:

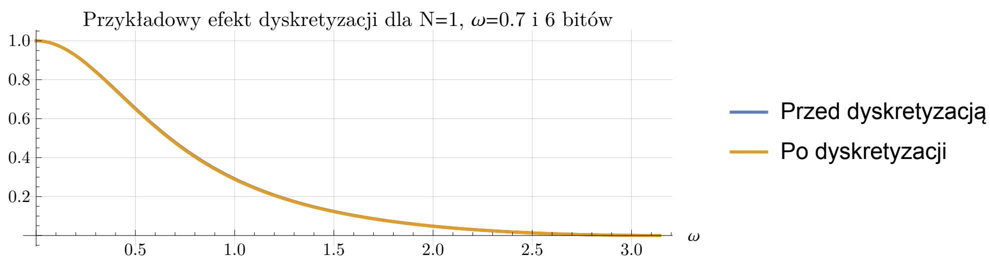
In[85]:=
```
NicePlot[DGemodels, DGemodelsDc2, 1, 1]
```

Out[85]=



In[89]:=
```
NicePlot[DGemodels, DGemodelsDc2, 1, 3]
```

Out[89]=



2 Badanie stabilności dyskretyzowanych filtrów cyfrowych

2.1 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, Δω}

2.1.1 Definicje

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

2.1.2 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}];
```

2.1.3 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}];
```

2.1.4 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}];
```

2.1.5 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}];
```

2.2 Funkcje pomocnicze

Pomocnicza funkcja by pobrać bieguny zdyskretyzowanych filtrów

```
ExtractPoles[tf_] := Module[{coeff},
  coeff = First[Flatten[ExpandNumerator@Numerator[tf]]];
  coeff = If[Length[coeff] > 0, coeff[[1]], coeff];
  (z /. Solve[Denominator@(tf / coeff) == 0, z])
 ]
```

Ustalamy ilość bitów do symulacji

```
bity = 6;
```

Przygotowywuję funkcję rysującą bieguny i zaznaczającą na czerwono te w których bieguny filtru zdyskretyzowanego wychodzą poza okrąg jednostkowy.

```
PlotPoles[poles_, poles2_] :=
 ListPlot[{Transpose[{Re[poles], Im[poles]}], Transpose[{Re[poles2], Im[poles2]}]},
  PlotRange → 1.2 {{-1, 1}, {-1, 1}}, PlotStyle → {PointSize[Large]},
  Background → If[Count[poles2, _? (Abs[#] ≥ 1 &), {1}] > 0, LightRed, White],
  PlotMarkers → {Automatic, Medium}, AxesOrigin → {0, 0}, Epilog → {Green, Circle[]},
  AspectRatio → Automatic]
```

```
AsymptoticallyStableQ[tfm_?ContinuousTimeModelQ] := If[
  Count[TransferFunctionPoles[tfm], Complex[x_?NonNegative, _] | x_?NonNegative, {3}] > 0,
  False, True]
AsymptoticallyStableQ[tfm_?DiscreteTimeModelQ] := If[
  Count[TransferFunctionPoles[tfm], _? (Abs[#] ≥ 1 &), {3}] > 0, False, True]
```

2.3 Dyskretyzacja współczynników filtru cyfrowego

2.3.1 Definicje

```
DiscretizeList[x_, max_, bits_] :=
  If[ # ≥ 0 , 1, -1] Round[ (Abs[#] / max) (Power[2, bits - 1] - 1)] max / (Power[2, bits - 1] - 1)  & /@ x;
DiscretizeComplexList[x_, y_, bits_] := Module[{xD, yD, coeffD, max},
   If[y == {}, {},
    max = Max[Abs[Join[Re@x, Im@x, Re@y, Im@y]]];
    xD = DiscretizeList[Re@x, max, bits] + ⅈ DiscretizeList[Im@x, max, bits];
    yD = DiscretizeList[Re@y, max, bits] + ⅈ DiscretizeList[Im@y, max, bits];
    {xD, yD}]
  ];
DiscretizeModelCoeffs[tf_, bits_] := Module[{zeros, poles, zerosD, polesD},
      zeros = CoefficientList[Numerator[TransferFunctionExpand[tf][z]][[1, 1]], z];
   poles = CoefficientList[Denominator[TransferFunctionExpand[tf][z]][[1, 1]], z];
   {zerosD, polesD} = DiscretizeComplexList[zeros, poles, bits];
      Chop[FromDigits[Reverse[zerosD], z] / (FromDigits[Reverse[polesD], z])]
  ];
```


2.3.2 Testy

```
{a, c} = DiscretizeComplexList[{1, 2, 3}, {ⅈ, ⅈ 2, ⅈ 3}, 2]
```

```
{{0, 3, 3}, {0, 3 ⅈ, 3 ⅈ}}
```

Dyskretyzujemy na poziomie filtrów cyfrowych

```
buttModel = TransferFunctionExpand@DGbmodels[[2, 2]]
```

$$\left(\frac{0.0302379 + 0.0604758\, z + 0.0302379\, z^2}{(0.572371 + 0.\, i) - (1.45142 + 0.\, i)\, z + z^2}\right)_1^{\mathcal{T}}$$

```
polesButt = TransferFunctionPoles[buttModel][[1, 1]]
```

```
{0.72571 - 0.213814 ⅈ, 0.72571 + 0.213814 ⅈ}
```

```
buttModelD = N@DiscretizeModelCoeffs[buttModel, bity]
```

$$\frac{0.04682 + 0.04682\, z + 0.04682\, z^2}{0.56184 - 1.45142\, z + 0.98322\, z^2}$$

```
polesButtD = ExtractPoles[buttModelD]
```

```
{0.738095 - 0.16323 ⅈ, 0.738095 + 0.16323 ⅈ}
```

2.3.3 Dyskretyzacja

```
DGbmodelsDc = Map[DiscretizeModelCoeffs[#, bity] &, DGbmodels, {2}];
```

```
DGc1modelsDc = Map[DiscretizeModelCoeffs[#, bity] &, DGc1models, {2}];
```

```
DGc2modelsDc = Map[DiscretizeModelCoeffs[#, bity] &, DGc2models, {2}];
```

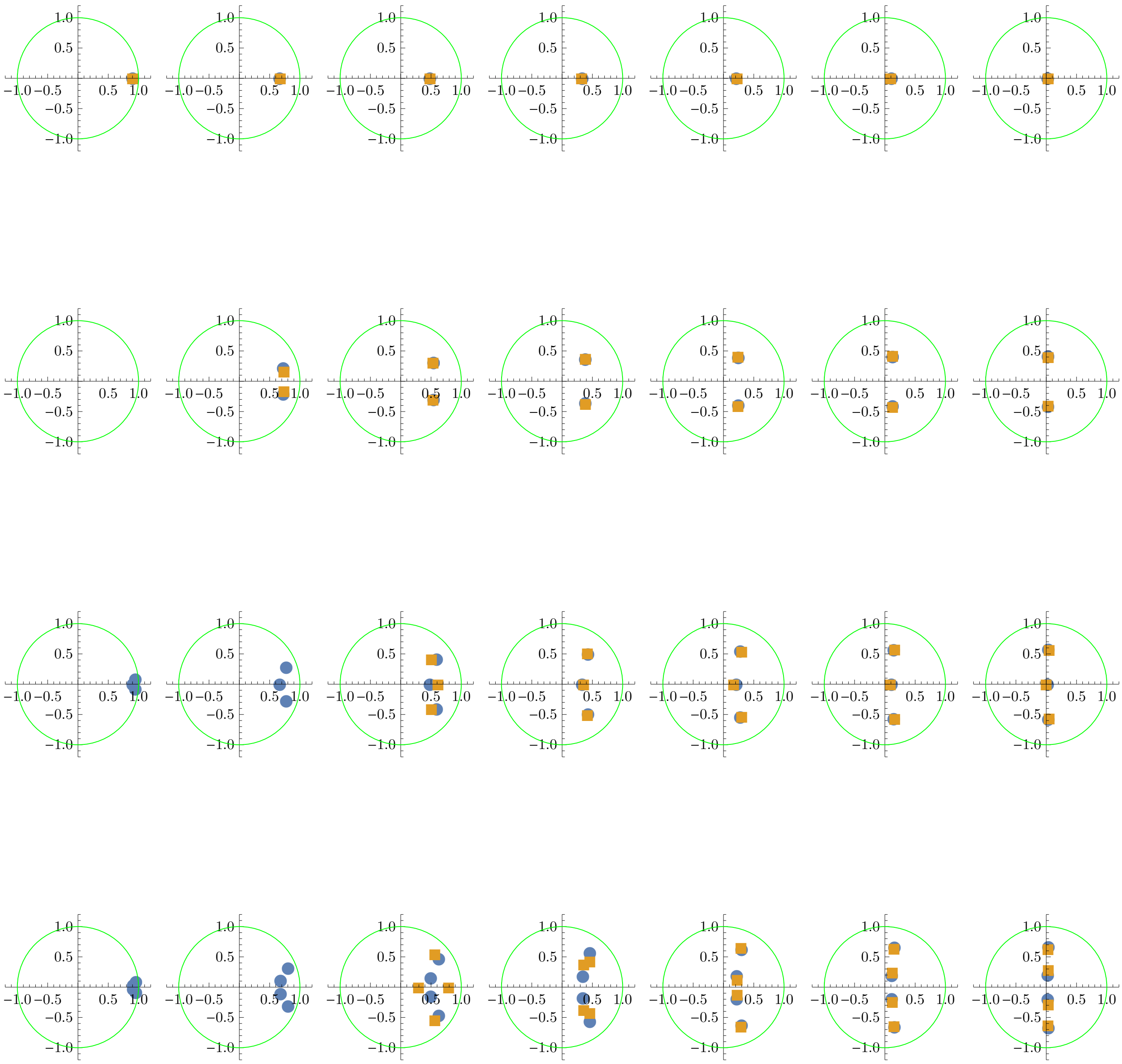
```
DGemodelsDc = Map[DiscretizeModelCoeffs[#, bity] &, DGemodels, {2}];
```

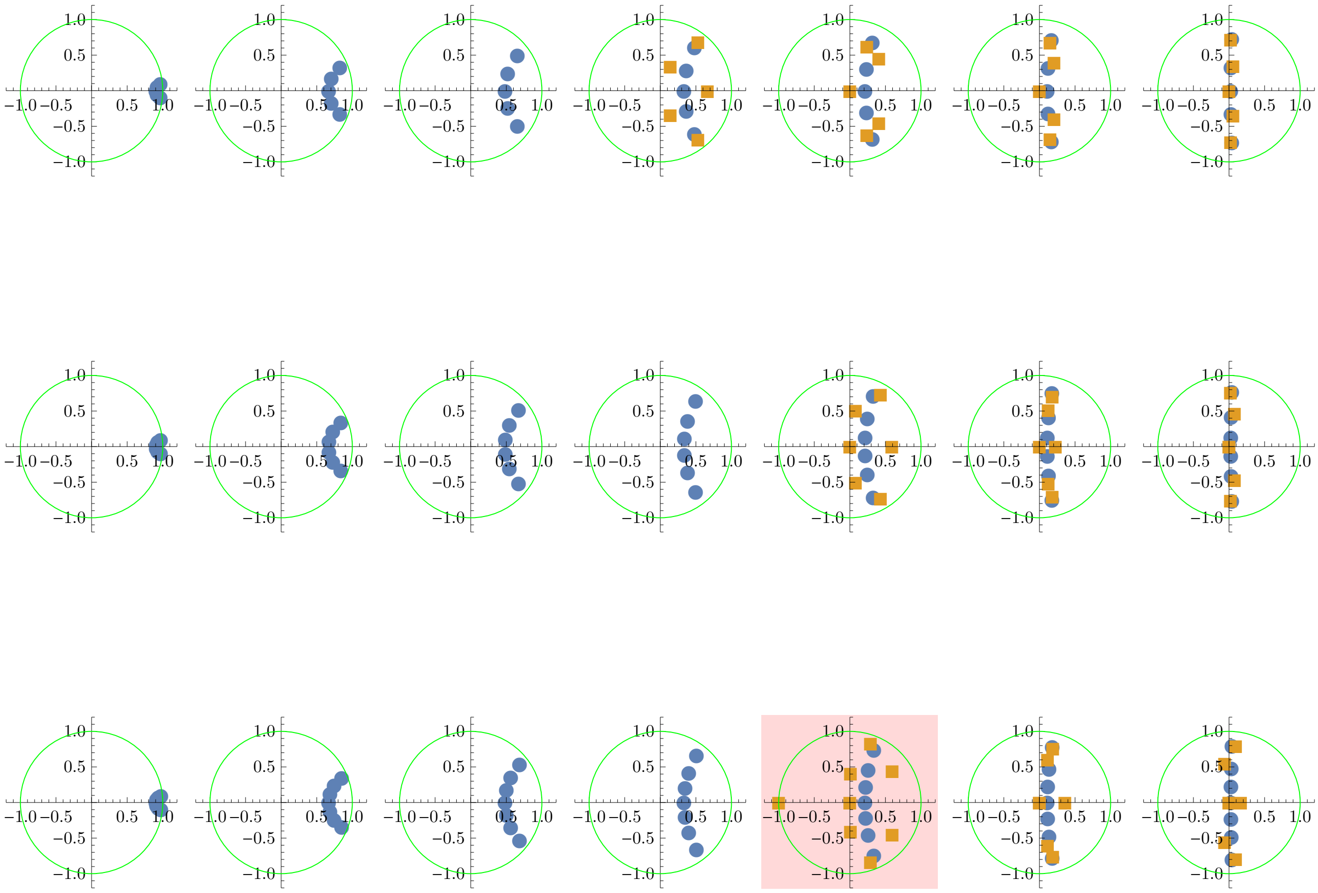
2.3.4 Porównanie położenia biegunów

★ Butterworth

W prawo rośnie częstotliwość, w dół rośnie rząd filtra.

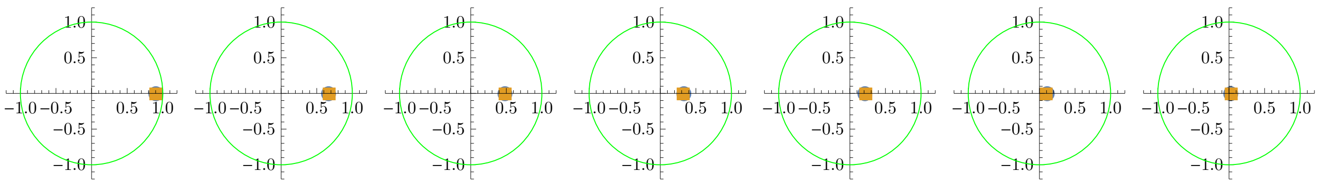
```
Grid@MapThread[PlotPoles[TransferFunctionPoles[#1][[1, 1]], ExtractPoles[#2]] &,
  {DGbmodels, DGbmodelsDc}, 2]
```

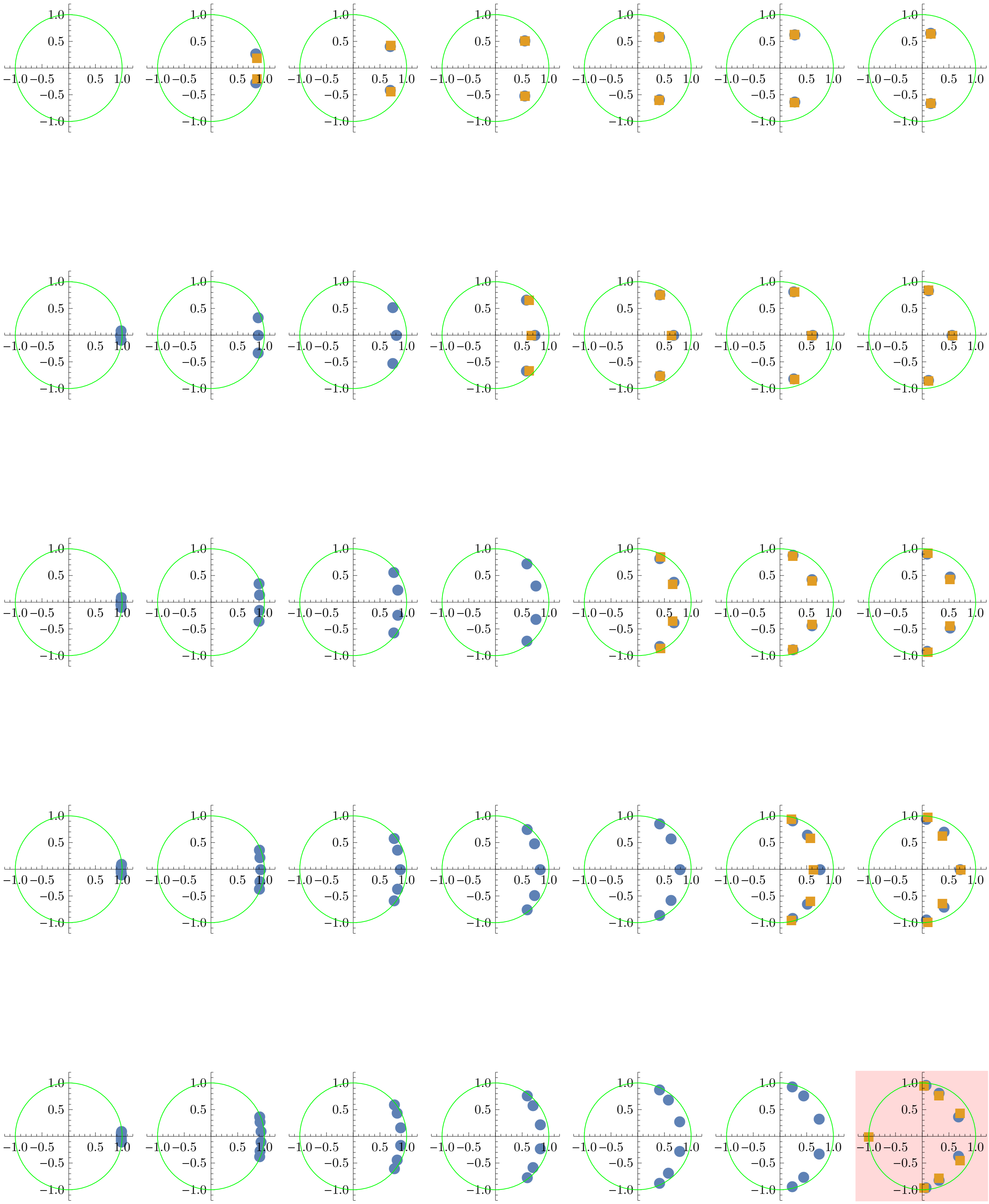


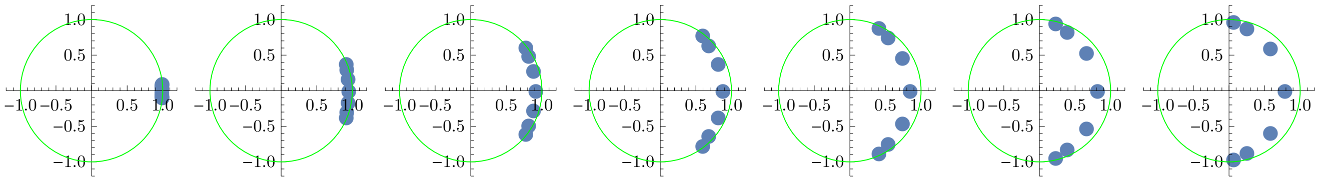


★ Chebyshev 1

```
Grid@MapThread[PlotPoles[TransferFunctionPoles[#1][[1, 1]], ExtractPoles[#2]] &,
  {DGc1models, DGc1modelsDc}, 2]
```

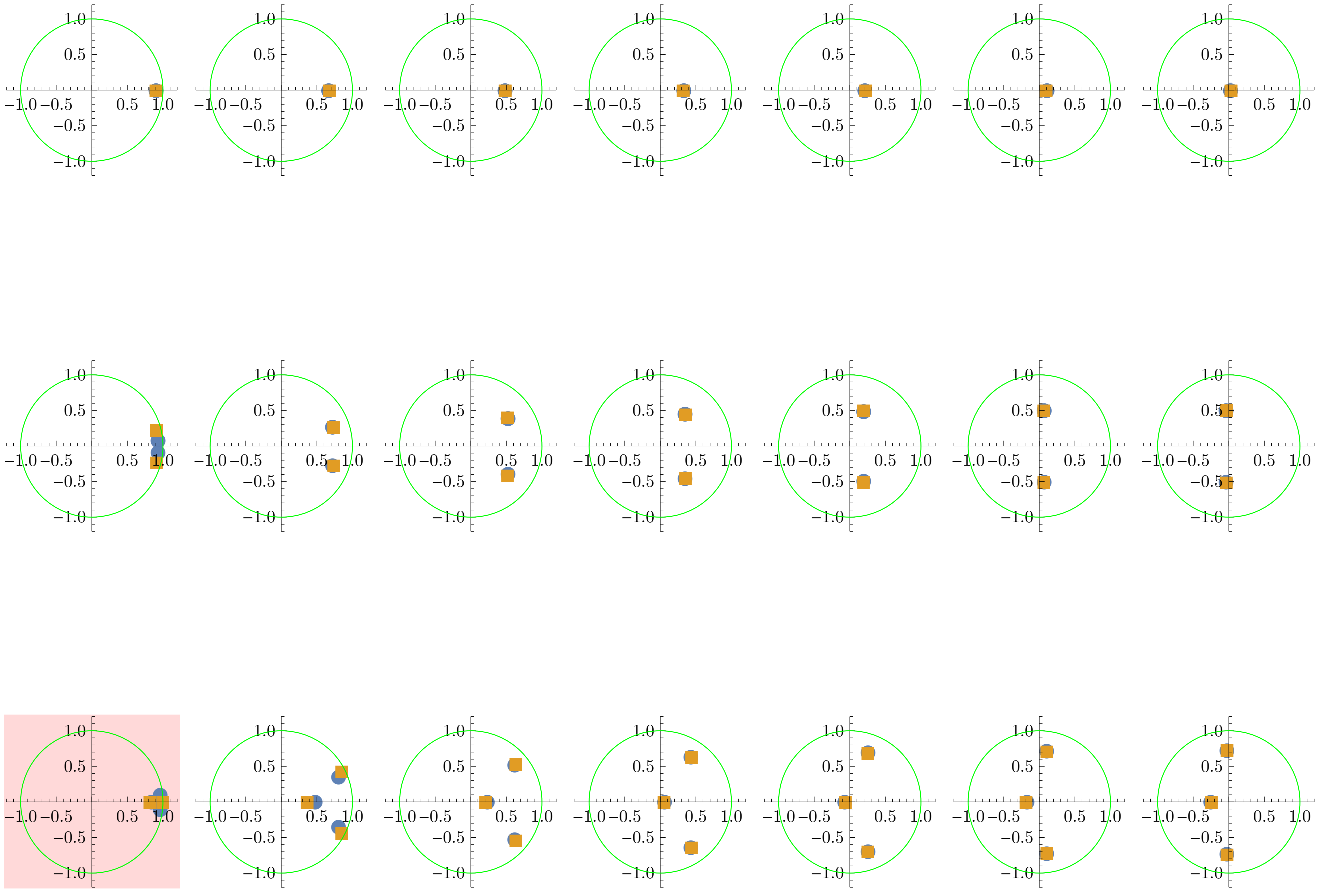


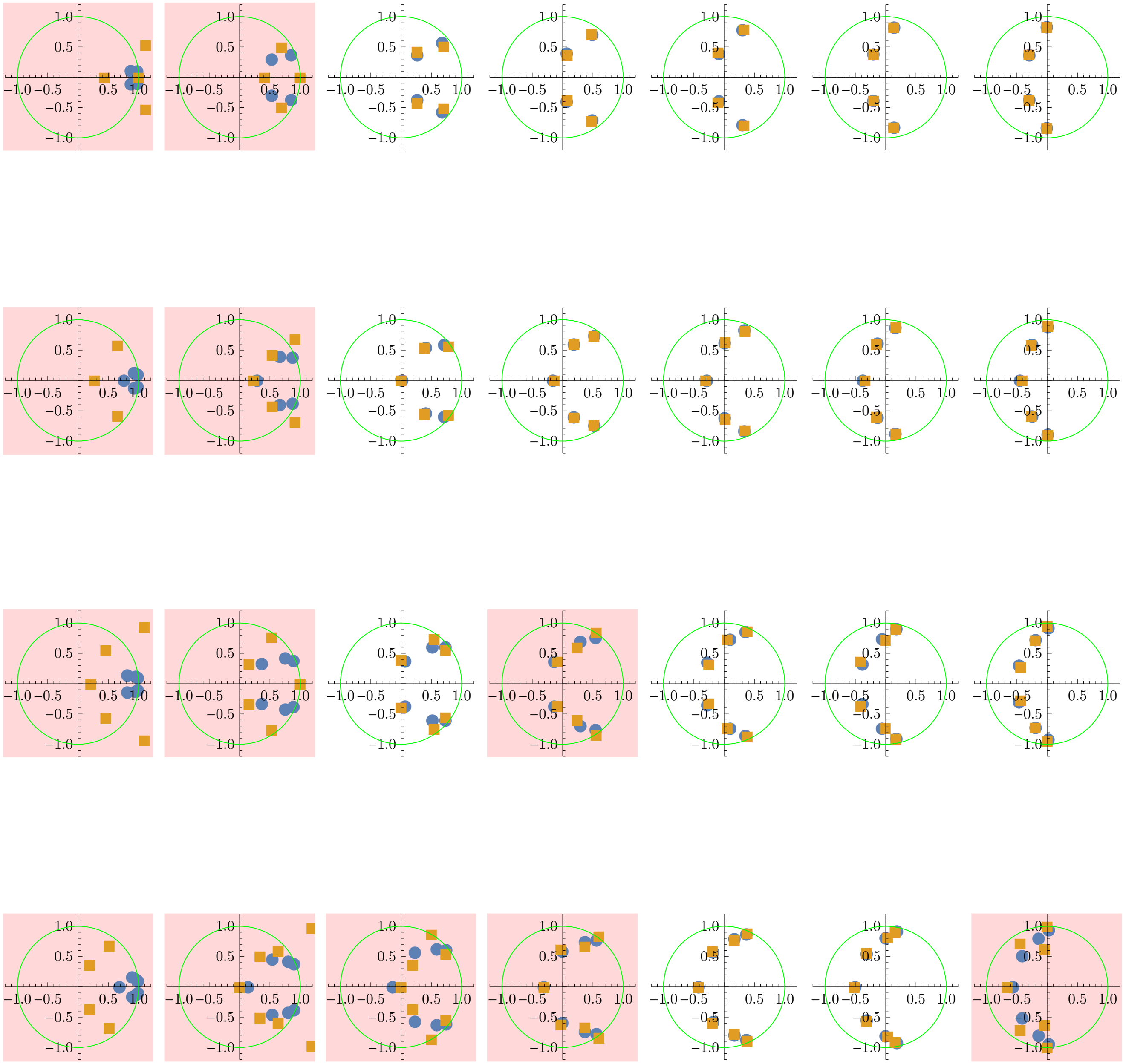

1.0
0.5
−1.0 −0.5
0.5
1.0
−0.5
−1.0



★ Chebyshev 2

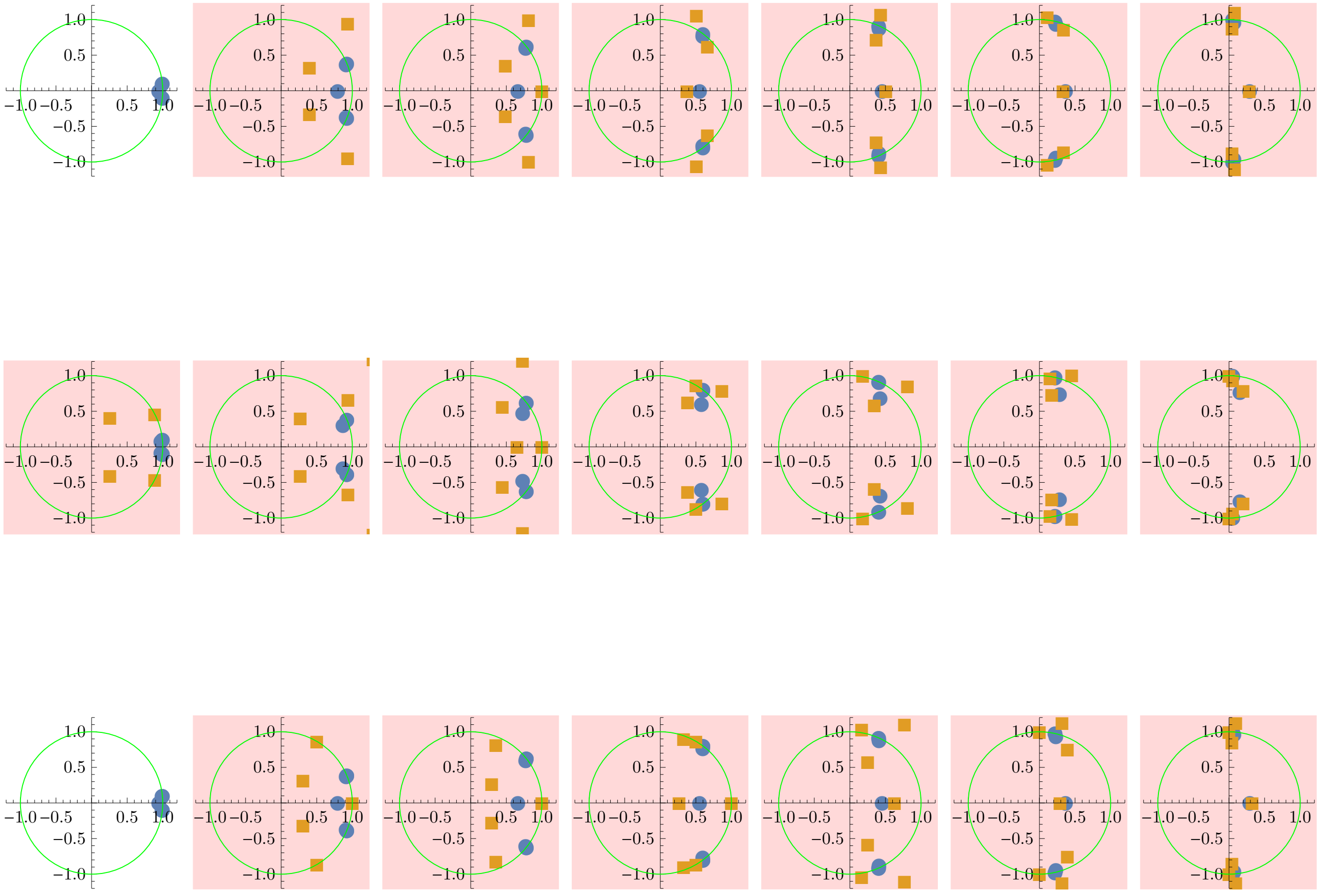
```
Grid@MapThread[PlotPoles[TransferFunctionPoles[#1][[1, 1]], ExtractPoles[#2]] &,
  {DGc2models, DGc2modelsDc}, 2]
```





★ Eliptic

```
Grid@MapThread[PlotPoles[TransferFunctionPoles[#1][[1, 1]], ExtractPoles[#2]] &,
  {DGemodels, DGemodelsDc}, 2]
```



2.4 Dyskusja

Istnieje problem polegający na znikaniu licznika (a więc również całego filtru) widoczny jest on przy niskich częstotliwościach filtru Butterwortha i Chebysheva 1
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)

3 Główne wyniki dotyczące stabilności

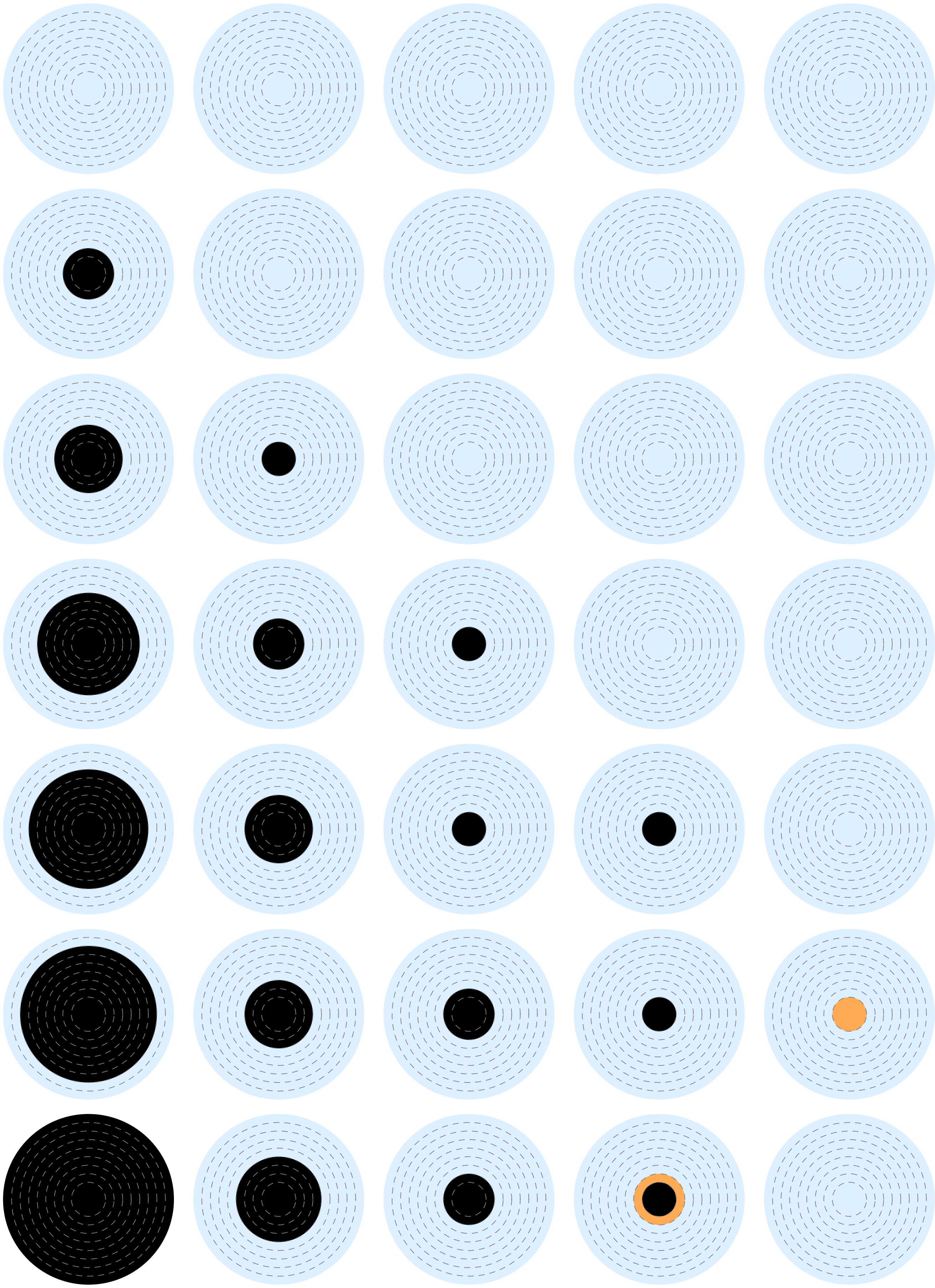
W tej części zostaną przedstawione najważniejsze wyniki dla dyskretyzacji w której współczynniki licznika i mianownika są dyskretyzowane razem.

Wizualizacja stabilności filtrów rządzi się zasada jest ta sama - w dół rośnie rząd, w prawo rośnie częstość graniczna.
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[Length[p] > 0, If[Count[p, _?(Abs[#] ≥ 1 &), {1}] > 0, Lighter[Orange], LightBlue],
   Black];

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[p7], Disk[{0, 0}, {0.4, 0.4}],
   EdgeForm[Directive[Dashed, Gray]], SP[p8], Disk[{0, 0}, {0.3, 0.3}],
   EdgeForm[Directive[Dashed, Gray]], SP[p9], Disk[{0, 0}, {0.2, 0.2}]}]
```

3.1 Butterworth

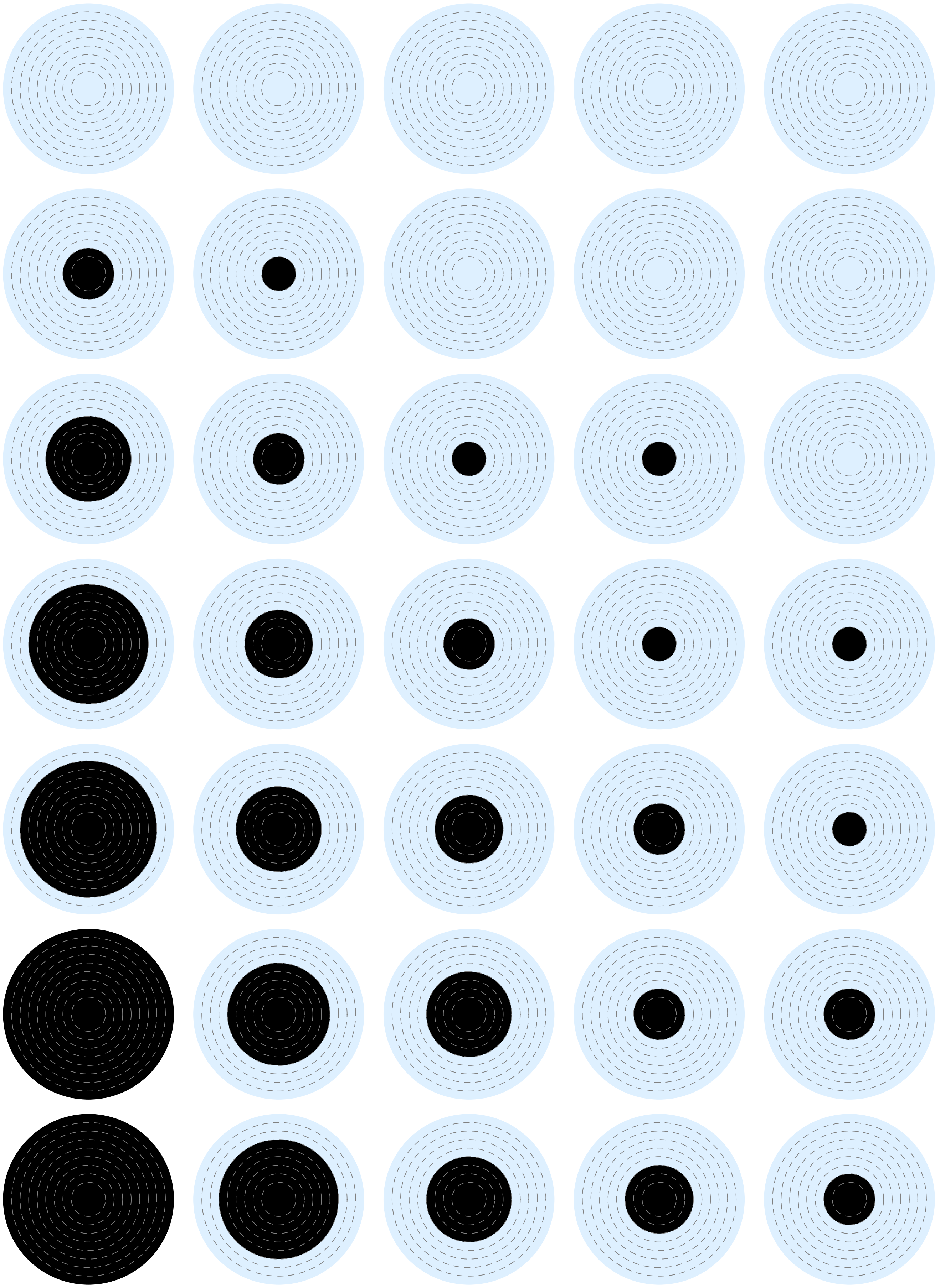
```
Allb = Table[Map[ExtractPoles[DiscretizeModelCoeffs[#, bits]] &, DGbmodels, {2}],
   {bits, 5, 32, 3}];

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



3.2 Chebyshev 1

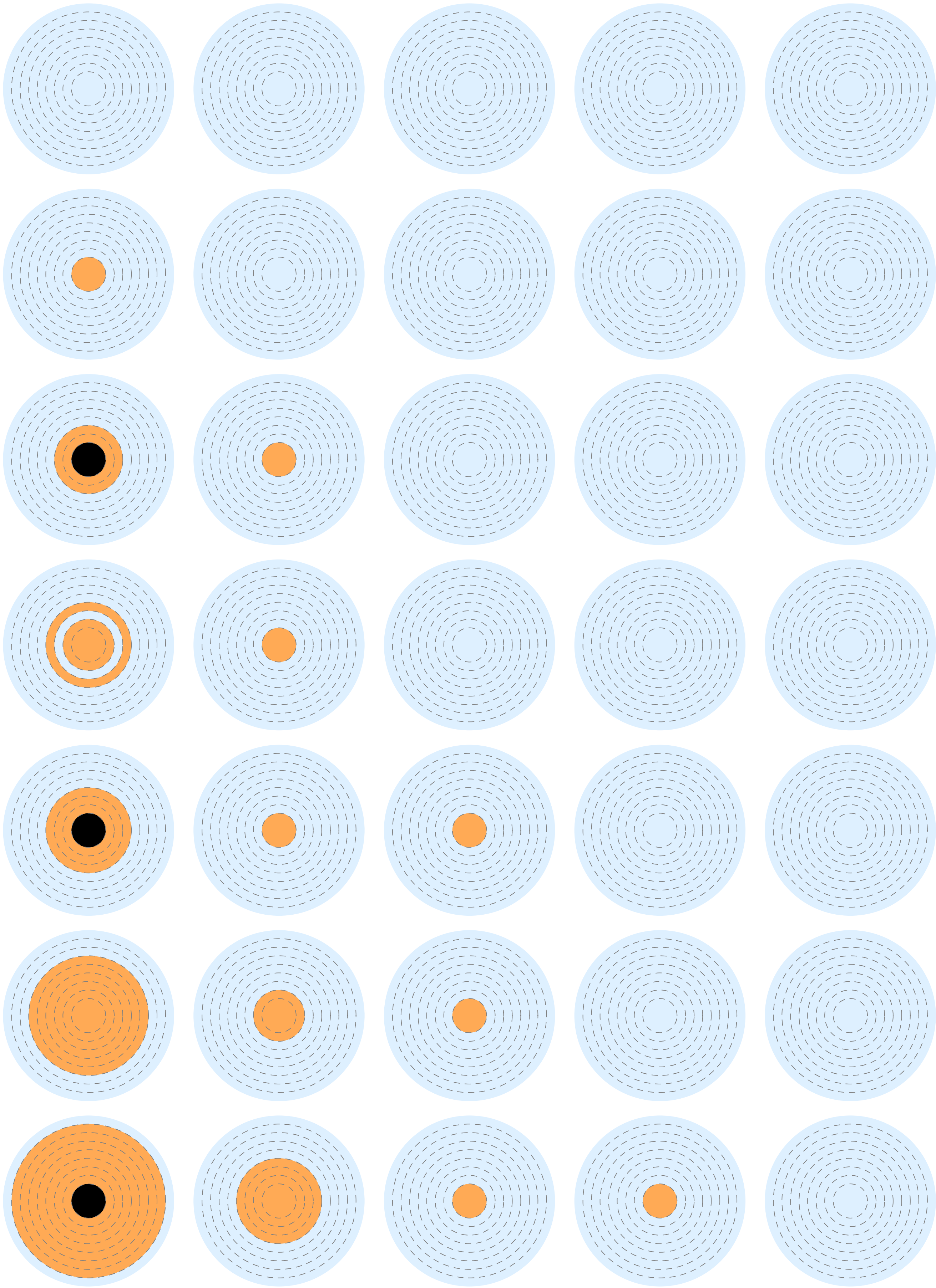
```
Allc1 = Table[Map[ExtractPoles[DiscretizeModelCoeffs[#, bits]] &, DGc1models, {2}],
   {bits, 5, 32, 3}];
```

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



3.3 Chebyshev 2

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



3.4 Eliptyczny

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