Ignacy ROGATTY, Mateusz MARASZEK, Paulina HOCZEK

Politechnika Wrocławska,

Filtr o nieskończonej odpowiedzi impulsowej (NOI)

Streszczenie. *Przedstawiony niżej tekst opisuje projekt filtru o nieskończonej odpowiedzi impulsowej. Zawiera informacje teoretyczne, na których została oparta praktyczna realizacja projektu oraz szczegółową instrukcję stworzonej aplikacji.*

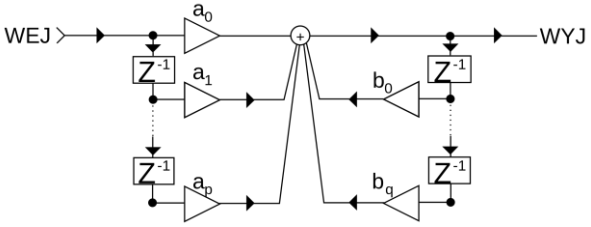
Słowa kluczowe: *filtr NOI, prototyp, sygnał, transformacja biliniowa*

Wstęp

Celem projektu jest stworzenie aplikacji do filtracji zadanego sygnału za pomocą filtra o nieskończonej odpowiedzi impulsowej (*NOI*) oraz wizualizacji otrzymanych efektów. Do realizacji celu wykorzystano środowisko *MATLAB* oraz wbudowanego środowiska *GUIDE* do tworzenia interfejsu graficznego.

Opis teoretyczny

Filtr o nieskończonej odpowiedzi impulsowej *(rys. 1)* jest filtrem cyfrowym rekursywnym, ponieważ wykorzystuje sprzężenie zwrotne. Próbki sygnału wyjściowego tego filtru zależą od poprzednich próbek sygnału wejściowego i poprzednich próbek sygnału wyjściowego przez co pewien skończony ciąg niezerowych wartości wejściowych może spowodować pojawienie się na wyjściu filtra *NOI* ciągu próbek o nieskończonym czasie trwania.



Rysunek 1. Schemat blokowy filtra NOI

Projektowanie filtra *NOI* w porównaniu z filtrem *SOI* jest bardziej skomplikowane. Zaletami filtra o nieskończonej odpowiedzi impulsowej są: niska złożoność obliczeniowa oraz niskie zapotrzebowanie na pamięć operacyjną. Wadami tegoż filtra są: rekursywność, która wprowadza potencjalne zagrożenie utraty stabilności, trudność w projektowaniu w porównaniu do filtrów *SOI* oraz większa wrażliwość na błędy zaokrągleń.

Do najczęściej stosowanych metod projektowania filtrów NOI należą:

Metoda standardowa transformaty ZET – polega na podziale filtra analogowego, na kilka filtrów o pojedynczym biegunie. Każdy taki filtr jest aproksymowany filtrem cyfrowym o również pojedynczym biegunie. Na końcu łączy się wszystkie uzyskane filtry cyfrowe w jeden filtr NOI wyższego rzędu.

Metoda transformacji biliniowej [3] – polega na utworzeniu prototypowego filtra analogowego, a następnie aproksymacji jego transmitancji układu ciągłego do transmitancji układu dyskretnego przy użyciu transformacji biliniowej przez podstawienie (1).

$$s = \frac{1}{T}\ln(z) \approx \frac{2}{T}\cdot\frac{1-z^{-1}}{1+z^{-1}} \tag{1}$$

gdzie: *T* – okres próbkowania.

Metoda transformacji biliniowej charakteryzuje się brakiem aliasingu, krótszym pasmem przejściowym filtra cyfrowego, prostotą *(nie ma potrzeby stosowania transformaty Laplace'a oraz ZET).*

Wszelkie informacje potrzebne do projektu zostały znalezione w skrypcie do wykładu, na stronie *MathWorks*, na *Wikipedii* i na innych stronach, do których linki znajdują się w spisie literatury.

Opis projektu

W projekcie do uzyskania filtra cyfrowego NOI wykorzystano metodę transformacji biliniowej. Do stworzenia filtra prototypowego wykorzystano funkcje *butter, cheby1, cheby2, ellip*. Wykorzystują one pięcio-krokowy algorytm:

1. Znajdują bieguny, zera i wzmocnienie analogowego filtra prototypowego przy użyciu funkcji *ellipap.*
2. Konwertują bieguny, zera i wzmocnienie do postać zmiennych stanu.
3. Jeżeli to konieczne, używają postaci zmiennych stanu, aby zamienić dolnoprzepustowy filtr na górnoprzepustowy, środkowoprzepustowy lub środkowozaporowy z określonymi wcześniej częstotliwościami granicznymi.
4. Dla projektowania filtra cyfrowego od razu używają funkcji *bilinear* [2] do utworzenia filtra NOI poprzez transformację biliniową.
5. Jeśli to konieczne to konwertują filtr w postaci zmiennych stanu [2] z powrotem do postaci transmitancji lub zero-biegunowej.

W celu wyświetlania charakterystyki amplitudowej prototypowego filtra analogowego i filtra cyfrowego na jednym wykresie wykorzystano funkcje *butter, cheby1, cheby2, ellip* [2] do stworzenia filtra analogowego, a następnie, oddzielnie, funkcję *bilinear* do utworzenia filtra cyfrowego o nieskończonej odpowiedzi impulsowej. W przypadku postaci zero-biegunowej, która została wykorzystana, funkcja *bilinear* wykorzystuje cztero-krokowy algorytm:

1. Jeżeli częstotliwość do skalowania została zdefiniowana to wykonana zostaje operacja skalowania częstotliwości próbkowania. Jeżeli nie to częstotliwość próbkowania zostaje podwojona.
2. Usuwa wszystkie nieskończone zera:
```
z = z(finite(z));
```
3. Wykorzystuje transformację biliniową do obliczenia zer, biegunów i wzmocnienia.
4. Dodaje dodatkowe zera o wartości -1 tak, aby otrzymany filtr miał odpowiedni rząd licznika i mianownika.

W programie najpierw definiowane są: rodzaj filtra *(dolnoprzepustowy/górnoprzepustowy/środkowoprzepustowy/środkowozaporowy)*, rząd, częstotliwości graniczne oraz próbkowania. Następnie z pliku tekstowego zostaje wczytany sygnał *(rys. 2)*.

```
function pushbutton6_Callback(hObject, eventdata, handles)
    [filename, pathname] = uigetfile({'*.txt'},'File Selector');
    if ~ischar(filename)
        return;
    end
    file = fullfile(pathname, filename);
    [fid, msg] = fopen(file, 'r');
    if fid == -1
        error(msg);
    end
    Data = fscanf(fid, '%g\n', [1, inf]);
    fclose(fid);
    handles.Data = Data;
    guidata(hObject, handles);
    set(handles.text15, 'String', filename);

% pozniej gdzies w programie
u = handles.Data';
```

Rysunek 2. Funkcja wczytująca sygnał wejściowy z pliku tekstowego

Funkcja *uigetfile* [2] otwiera okno dialogowe i umożliwia wybór pliku. Plik zostaje otwarty w trybie do odczytu przy użyciu funkcji *fopen.* Funkcja *fscanf* pobiera dane zapisane w wybranym formacie, w tym przypadku jest to jedna kolumna wartości o niezdefiniowanej ilości wyrazów, i przechowuje je w zmiennej *Data*. Funkcja *fclose* zamyka plik. Funkcja set wyświetla nazwę wybranego pliku w postaci statycznego tekstu w GUI. Następnie transponowany wektor *Data* jest zapisany w zmiennej *u* jako sygnał wejściowy.

```
wgr = 2*pi*fgr;
wp = 2*pi*fp;
Rp = 0.3;
Rs = 30;
N=size(u,1);
t=0:1/fp:(N-1)/fp;
Nw=500;
axes(handles.axes5)
plot(t(1:Nw),u(2091:Nw+2090),'k'); grid;
xlabel('t');
ylabel('U');
```

Rysunek 3. Kod definiujący niektóre zmienne i wyświetlający przebieg sygnału wejściowego

Na *rysunku 3* widać kod definiujący zmienne na podstawie wcześniej wczytanych wartości częstotliwości granicznej *(w przypadku filtrów środkowoprzepustowego oraz środkowozaporowego są to dwie częstotliwości)* oraz częstotliwości próbkowania. Częstotliwości graniczne oraz częstotliwość próbkowania zostają przekonwertowane na pulsację. Zafalowania pasma przepustowego i zaporowego poszczególnych filtrów zostały zdefiniowane na stałe w programie, i dobrane tak, aby były dobrze widoczne na wykresie. Następnie została zdefiniowana dziedzina czasu i odległość między próbkami. Zmienna *Nw* określa ilość próbek wyświetlonych na przebiegu sygnału wejściowego, a także sygnału wyjściowego. Funkcja *axes* łączy wykres z GUI. Funkcja plot wykreśla wykres sygnału wejściowego od czasu przesunięty o pewną ilość próbek – zabieg ten zostanie wytłumaczony w rozdziale *Badania*. Funkcje *xlabel* oraz *ylabel* wyświetlają podpisy osi na wykresie.

```
[b,a] = ellip(n,Rp,Rs,[wd wg],'bandpass','s');
[z,p,k] = ellip(n,Rp,Rs,[wd wg],'bandpass','s');

W = 0:.1: wp ;
H = freqs(b,a,W);
f = W./(2*pi);

[zd,pd,kd] = bilinear(z,p,k,fp)
sos = zp2sos(zd,pd,kd);
[Hz,Wz]=freqz(sos,512,fp);

axes(handles.axes1)
plot(f(1:5*wp),abs(H(1:5*wp)),'b'); hold on;
plot(Wz,abs(Hz),'r'); grid on; hold off;
```

Rysunek 4. Projektowanie filtra eliptycznego środkowoprzepustowego

Przy użyciu funkcji *ellip* uzyskano filtr prototypowy eliptyczny środkowoprzepustowy *(rys. 4)*. Zmienna *n* definiuje rząd filtra, zmienne *Rp* i *Rs* definiują zafalowania filtra, *wd* oraz *wg* to pulsacja graniczna dolna oraz górna. Słowo kluczowe *bandpass* definiuje rodzaj filtra *(środkowoprzepustowy)*. Litera *s* oznacza, że będzie to filtr analogowy. Filtr zostaje zapisany w postaci transmitancji oraz w postaci zero-biegunowej. Postać transmitancji jest wykorzystana do wyświetlenia charakterystyki amplitudowej filtra prototypowego, natomiast postać zero-biegunowa służy do projektowania filtra NOI. Postać transmitancji nie nadaje się do projektowania filtrów NOI – problem ten jest bardziej opisany w rozdziale *Badania*.

Następnie definiowana jest dziedzina pulsacji służąca do wykreślenia charakterystyki amplitudowej filtra analogowego. Funkcja *freqs* wyznacza ciąg próbek odpowiedzi częstotliwościowej dla podanej transmitancji filtra analogowego. Następnie dziedzina pulsacji jest zamieniana na dziedzinę częstotliwości w celu późniejszego wykreślenia wykresu.

Funkcja *bilinear* wykonuje transformację biliniową na postaci zero-biegunowej filtra prototypowego i uzyskiwany jest w ten sposób filtr cyfrowy. Funkcja *zp2sos* zamienia postać zero-biegunową filtra NOI na postać sekcji drugiego rzędu w celu późniejszego wyświetlenia filtra oraz wykorzystania go do filtracji sygnału wejściowego. Każdy filtr NOI parzystego rzędu można przedstawić jako kaskadę sekcji drugiego rzędu. Dla nieparzystych rzędów filtru trzeba zastosować dodatkowo jedną sekcję pierwszego rzędu. Funkcja *freqz* [2] wyznacza ciąg próbek odpowiedzi częstotliwościowej dla podanej postaci sekcji drugiego rzędu filtra cyfrowego *(512 próbek)*. Funkcja *axes* łączy wykresy z GUI, a funkcja *plot* wykreśla charakterystykę amplitudową filtra analogowego oraz cyfrowego na jednym wykresie.

```
Le=length(t)-1;
u_fft = abs(fft(u))/(Le/2);
u_fft = u_fft(1:Le/2+1);
u_fft(2:end-1) = 2*u_fft(2:end-1);
ft = fp*(0:(Le/2))/Le;
axes(handles.axes4)
plot(ft,u_fft); grid on;
```

Rysunek 5. Uzyskiwanie widma amplitudowego sygnału wejściowego

Funkcja *length* określa długość wektora dziedziny czasu *(rys. 5)*. Funkcja *fft* to szybka transformata Fouriera. Dzięki zastosowaniu tej funkcji oraz funkcji *abs*, czyli przekształcenia na wartość bezwzględną, uzyskiwane jest

widmo amplitudowe sygnału wejściowego. Następne trzy linijki kodu ustalają dziedzinę częstotliwości do połowy częstotliwości próbkowania, aby pominąć częstotliwości większe od częstotliwości Nyquista. Na końcu wyświetlane jest widmo amplitudowe sygnału wejściowego.

```
u_przefiltrowane = sosfilt(sos,u);
axes(handles.axes2)
plot(t(1:Nw),u_przefiltrowane(2091:Nw+2090),'k'); grid;
```

Rysunek 6. Filtrowanie sygnału wejściowego

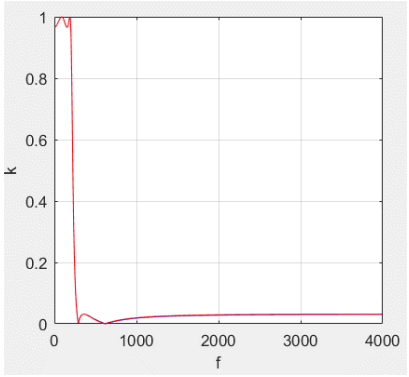
Funkcja *sosfilt* [2] filtruje sygnał wejściowy używając postaci sekcji drugiego rzędu filtra NOI *(rys. 6)*. Następnie wyświetlany jest sygnał wyjściowy przesunięty o 2090 próbek.

```
Le=length(t)-1;
u_fft_przefiltrowane = abs(fft(u_przefiltrowane))/(Le/2);
u_fft_przefiltrowane = u_fft_przefiltrowane(1:Le/2+1);
u_fft_przefiltrowane(2:end-1) = 2*u_fft_przefiltrowane(2:end-1);
ft = fp*(0:(Le/2))/Le;
axes(handles.axes3)
plot(ft,u_fft_przefiltrowane); grid on;
```

Rysunek 7. Uzyskiwanie widma amplitudowego sygnału wyjściowego

Widmo amplitudowe sygnału wyjściowego *(rys. 7)* jest uzyskiwane i przefiltrowane w ten sam sposób co widmo amplitudowe sygnału wejściowego.

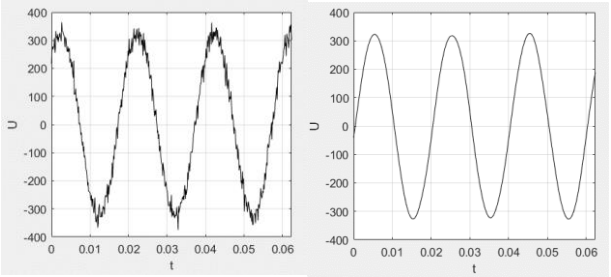
Badania



Rysunek 8. Charakterystyka amplitudowa dolnoprzepustowego filtra NOI na podstawie filtra eliptycznego czwartego rzędu

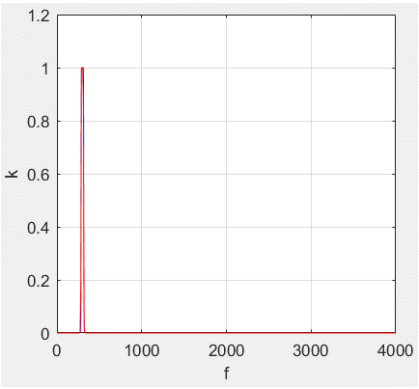
Badania przeprowadzono dla wszystkich rodzajów filtrów i dla wszystkich filtrów prototypowych. W pierwszym przykładzie wybrano filtr prototypowy eliptyczny dolnoprzepustowy czwartego rzędu. Częstotliwość graniczna została ustalona na 200Hz *(rys. 8)*.

Wczytano sygnał nr 14 z paczki sygnałów – zaszumiony sygnał sinusoidalny o częstotliwości 50Hz, częstotliwość próbkowania 8000Hz *(rys. 9)*.



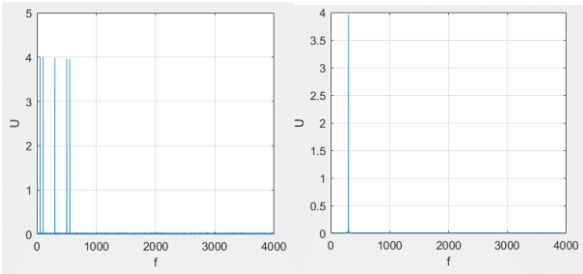
Rysunek 9. Sygnał wejściowy (syg14) oraz wyjściowy filtra dolnoprzepustowego

Na wykresie sygnału wyjściowego widać, że szumy zostały skutecznie odfiltrowane *(rys. 9)*.



Rysunek 10. Charakterystyka amplitudowa środkowoprzepustowego filtra NOI na podstawie filtra Butterwortha dziesiątego rzędu

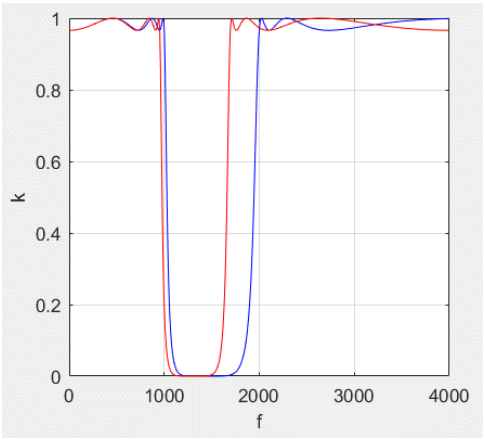
W drugim przykładzie wybrano filtr prototypowy Butterwortha środkowoprzepustowy dziesiątego rzędu *(rys. 10)*. Częstotliwość graniczna dolna została ustalona na 285Hz, a górna na 315Hz. Można powiedzieć, że uzyskano filtr pasmowoprzepustowy o częstotliwości środkowej 300Hz i szerokości pasma 30Hz.



Rysunek 11. Widmo amplitudowe sygnału wejściowego (syg16) oraz sygnału wyjściowego filtra środkowoprzepustowego

Wczytano sygnał nr 16 z paczki sygnałów – zaszumiony sygnał składający się z pięciu składowych o różnych częstotliwościach *(rys. 11)*.

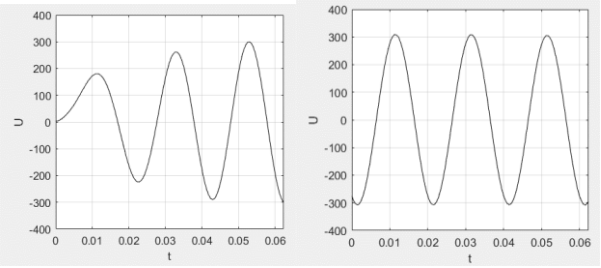
Na wykresie widma amplitudowego sygnału wyjściowego widać, że udało się wydobyć z sygnału wejściowego składową o częstotliwości 300Hz.



Rysunek 12. Charakterystyka amplitudowa środkowozaporowego filtra NOI na podstawie filtra Czebyszewa I typu

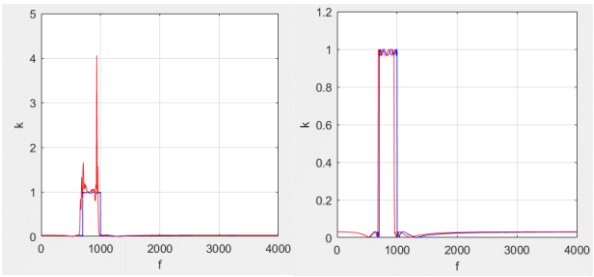
Jak widać na przykładzie filtra prototypowego Czebyszewa I typu środkowozaporowego *(rys. 12)* wraz ze wzrostem częstotliwości granicznych zwiększa się różnica

między charakterystyką amplitudową filtra analogowego oraz cyfrowego. Jest to wynikiem aproksymacji użytej w transformacji biliniowej. Aby zapobiec takiej sytuacji stosuje się operację skalowania częstotliwości (prewarping [1]).



Rysunek 13. Sygnał wyjściowy filtra dolnoprzepustowego o częstotliwości granicznej 70Hz wyświetlany od pierwszej próbki (po lewej) oraz od 2091 próbki (po prawej)

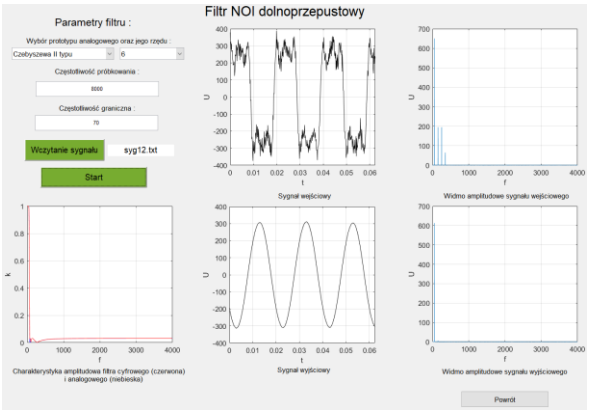
Sygnały wejściowy oraz wyjściowy są wyświetlane od 2091 próbki. Sygnał nie może być wyświetlany od próbki pierwszej, ponieważ sygnał wyjściowy, w zależności od parametrów filtra, może mieć większy lub mniejszy czas ustalenia, czego efekt widać na *rysunku 13*.



Rysunek 14. Charakterystyka amplitudowa filtra NOI (czerwony) oraz analogowego (niebieski) środkowoprzepustowego w postaci transmitancji (po lewej) i w postaci zero-biegunowej (po prawej)

Na przykładzie prototypowego filtra eliptycznego dziesiątego rzędu *(rys. 14)* przy projektowaniu filtra NOI w postaci transmitancji możemy zauważyć, że charakterystyka amplitudowa jest mocno zniekształcona. Jest to spowodowane błędami zaokrągleń. Po zmianie na postać zero-biegunową problem zniknął.

Interfejs graficzny użytkownika



Rysunek 15. Interfejs graficzny programu

Przejrzysty interfejs graficzny zapewnia łatwość korzystania z programu *(rys. 15)*. Po wybraniu rodzaju filtra w nowym oknie istnieje możliwość wybrania prototypu analogowego z listy rozwijanej na podstawie którego zostanie utworzony filtr NOI oraz wybrania jego rzędu również z listy rozwijanej. Istnieje również możliwość wybrania sygnału, który będzie badany oraz podania wartości częstotliwości próbkowania oraz granicznej. Kiedy zostanie wybrany prototyp, rząd, sygnał oraz zostaną uzupełnione pola z wartościami istnieje możliwość wczytania wybranego sygnału dzięki przyciskowi *Wczytanie sygnału*. Po użyciu przycisku *Start* program wykreśla charakterystykę amplitudową filtra prototypowego i filtra NOI, wykreśla przebiegi sygnału wejściowego i wyjściowego oraz widmo amplitudowe sygnału wejściowego i wyjściowego. Przycisk powrót umożliwia powrót do wyboru rodzaju filtra.

Wnioski

Środowisko *MATLAB* oraz narzędzie *DSP System Toolbox* pozwala na stworzenie praktycznie dowolnej aplikacji z zastosowaniem cyfrowego przetwarzania sygnałów.

Środowisko *GUIDE* umożliwia stworzenie prostego interfejsu graficznego użytkownika dla programu *MATLAB'a* lub *Simulink'a*. Pomimo tego, że wkrótce zostanie usunięte z *MATLAB'a* nadal jest wystarczająco funkcjonalne. Jeszcze lepszym i bardziej funkcjonalnym środowiskiem jest *App Designer,* do którego istnieje możliwość migracji projektów z środowiska *GUIDE*.

Transformacja biliniowa powoduje rozbieżność charakterystyki amplitudowej utworzonego filtra cyfrowego od filtra prototypowego. Aby tej sytuacji zapobiec należy stosować operację skalowania częstotliwości (prewarping). Jest ona na przykład wbudowana w funkcje *butter, cheby1, cheby2, ellip,* które to używają jej przy bezpośrednim tworzeniu filtra cyfrowego.

Projektując filtry o nieskończonej odpowiedzi impulsowej należy pamiętać o stosowaniu postaci zero-biegunowej zapisu filtra, ponieważ postać transmitancji jest bardzo podatna na błędy zaokrągleń, które występują tym częściej im wyższy jest rząd filtra.

Przy wyświetlaniu przebiegu sygnału należy pamiętać o różnym czasie ustalenia przebiegu dla różnych parametrów filtra i wyświetlać przebieg dopiero po pewnym czasie, aby nie uchwycić na wykresie jego stanu nieustalonego *(o ile nie jest to pożądane)*.

ODNOŚNIKI

[1] *Wikibooks*
https://en.wikibooks.org/wiki/Digital_Signal_Processing/Bilinear_Transform#Prewarping

[2] *MathWorks*
https://www.mathworks.com/help/signal/ref/freqz.html
https://www.mathworks.com/help/signal/ref/bilinear.html
https://www.mathworks.com/help/simulink/slref/statespace.html
https://www.mathworks.com/help/signal/ref/butter.html
https://www.mathworks.com/help/signal/ref/cheby1.html
https://www.mathworks.com/help/signal/ref/cheby2.html
https://www.mathworks.com/help/signal/ref/ellip.html
https://www.mathworks.com/help/matlab/ref/uigetfile.html
https://www.mathworks.com/help/signal/ref/sosfilt.html

[3] *Wikipedia*
https://pl.wikipedia.org/wiki/Metoda_Tustina

***Autorzy**:*
Paulina Hoczek, Politechnika Wrocławska, E-mail: 250441@student.pwr.edu.pl

Mateusz Maraszek, Politechnika Wrocławska, E-mail: 250452@student.pwr.edu.pl

Ignacy Rogatty, Politechnika Wrocławska, E-mail: 250414@student.pwr.edu.pl