Wydział Elektroniki i Technik Informacyjnych
Politechnika Warszawska

Sterowanie Procesami

Sprawozdanie z projektu II
Zadanie 42

Michał Pióro
324881

Warszawa, czerwiec 2024

Spis treści

1. Modele

1.1. Dyskretyzacja transmitancji ciągłej

Transmitancja ciągła

$$G(s) = \frac{K_o e^{-T_o s}}{(T_1 s + 1)(T_2 s + 1)} \tag{1.1}$$

Gdzie:

$$\begin{aligned} K_o &= 3,8 \\ T_o &= 5 \\ T_1 &= 1,74 \\ T_2 &= 5,23 \end{aligned}$$

1.1.1. Dyskretyzacja w programie MATLAB

Dla transmitancji 1.1 została przeprowadzona dyskretyzacja z wykorzystaniem ekstrapolatora pierwszego rzędu przy okresie próbkowania $T_p = 0,5s$. Obliczenia zostały przeprowadzone w MATLABie z wykorzystaniem funkcji *c2d* oraz zapisu transmitancji w *tf*

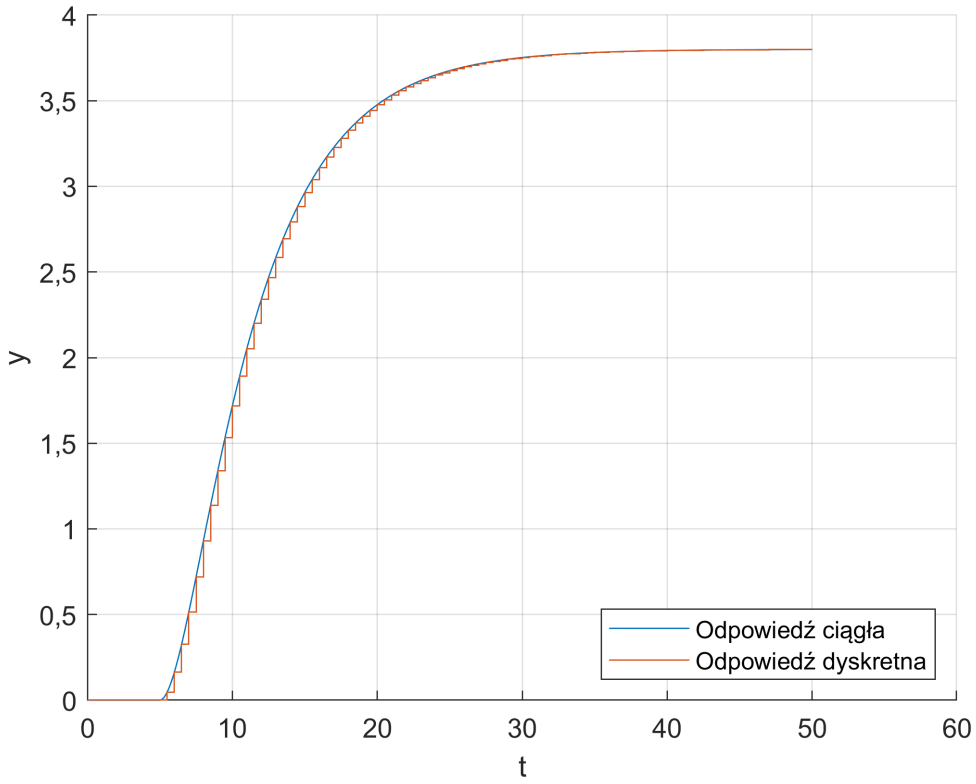
```
s = tf('s');
data = struct();
data.K_0=3.8;
data.T_0=5;
data.T_1=1.74;
data.T_2=5.23;
data.T_p=0.5;
data.Gsnum=data.K_0*exp(-data.T_0*s);
data.Gsden=(data.T_1*s+1)*(data.T_2*s+1);
data.Gstf=data.Gsnum/data.Gsden;
data.Gztf=c2d(data.Gstf,data.T_p,'zoh');
```

Transmitancja dyskretna

$$G(s) = \frac{0,046z^{-11} + 0,0405z^{-12}}{1 - 1,6591z^{-1} + 0,6818z^{-2}} \tag{1.2}$$

1.2. Odpowiedzi skokowe

```
figure;
hold on;
[yc, tc]=step(data.Gstf, [0 50]);
[yd, td]=step(data.Gztf, [0 50]);
plot(tc,yc);
stairs(td,yd);
```



Rys. 1.1. Odpowiedzi skokowe transmitancji ciągłej oraz dyskretnej.

W celu uzyskania odpowiedzi skokowej transmitancji ciągłej i dyskretnej wykorzystano ich zapis w *tf* oraz funkcję *step*, co pozwoliło na otrzymanie wektorów wartości do wyświetlenia na wykresie.

Wnioski

Model dyskretny został wyznaczony poprawnie, ponieważ na tyle na ile jest to możliwe w przypadku ekstrapolacji pierwszego rzędu oraz przyjętego czasu próbkowania.

1.3. Wzmocnienia statyczne

Do wyznaczenia wzmocnień statycznych wykorzystano postacie symboliczne transmitancji ciągłej i dyskretnej.

```
% Symboliczny zapis obu transmitancji
syms s z
data.Gs=poly2sym(data.Gstf.Numerator, s);
data.Gs=data.Gs/poly2sym(data.Gstf.Denominator, s);
data.Gs=collect(data.Gs)*exp(-data.Gstf.OutputDelay*s);
data.Gz=poly2sym(data.Gztf.Numerator, z);
data.Gz=data.Gz/poly2sym(data.Gztf.Denominator, z);
data.Gz=collect(data.Gz)*z^(-data.Gztf.OutputDelay);
```

W przypadku wzmocnienia statycznego transmitancji ciągłej obliczono granicę dla $s \to 0$ korzystając z podstawienia $s = 0$.

```
function [K_stat]=wzm_stat_Gs(Gs)
syms s
K_stat=collect(subs(Gs, s,0));
end
```

$$K_{stat} = \lim_{s \longrightarrow 0} G(s) = 3,8 \tag{1.3}$$

W przypadku wzmocnienia statycznego transmitancji dyskretnej obliczono granicę dla $z \rightarrow 1$ korzystając z podstawienia $z = 1$.

```
function [K_stat]=wzm_stat_Gz(Gz)
syms z
K_stat=collect(subs(Gz,z,1));
end
```

$$K_{stat} = \lim_{z \longrightarrow 1} G(z) = 3,8 \tag{1.4}$$

1.4. Porównanie

Wzmocnienie statyczne modelu ciągłego 1.3 oraz modelu dyskretnego 1.4 są sobie równe, co jest zgodne z teorią, ponieważ na wartość w po ustabilizowaniu się odpowiedzi nie ma wpływu to czy model jest ciągły czy dyskretny.

1.5. Równanie różnicowe

W celu wyznaczenia równania różnicowego została wykorzystana wcześniej obliczona transmitancja dyskretna 1.2.

```
function [b, c]=row_roznic(Gztf)
num = Gztf.Numerator{1};
den = Gztf.Denominator{1};
del = Gztf.OutputDelay;
% Wyciągnięcie i przeliczenie wektorów parametrów b i c
b=(-1).*den(2:end)./den(1);
c=zeros(1,del+length(num)-1);
c(1+del:end)=num(2:end)./den(1);
end
```

Otrzymana postać równania różnicowego:

$$y(k) = 1,6591y(k-1) - 0,6818y(k-2) + 0,046u(k-11) + 0,0405u(k-12) \tag{1.5}$$

Wnioski

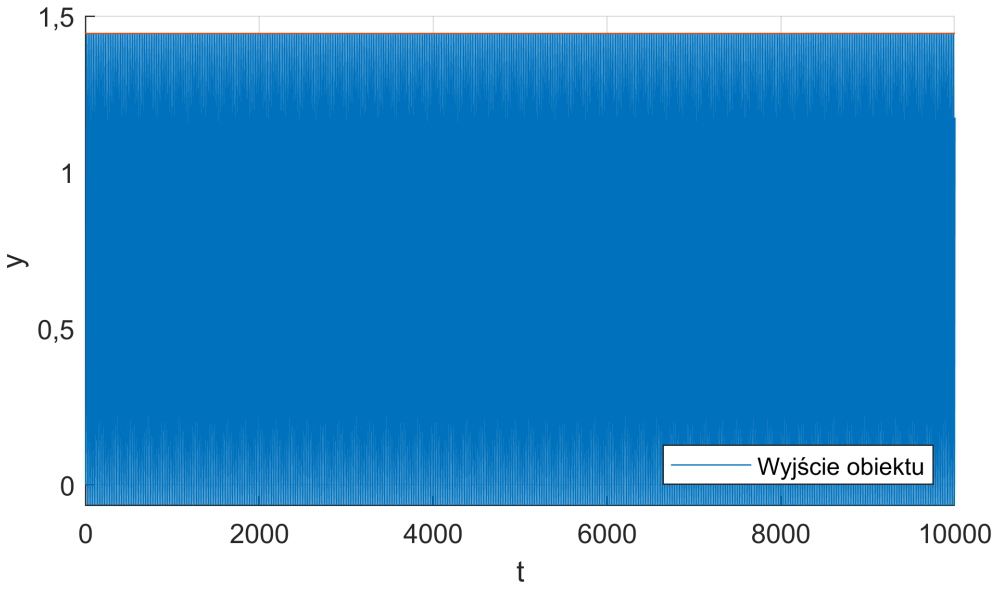
Otrzymane równanie różnicowe zostało przedstawione z zaokrąglonymi wartościami współczynników b_i oraz c_i. Taka postać modelu znacznie ułatwia implementację regulatorów , symulatorów oraz ich sprawdzanie ponieważ, pozwala wyznaczyć wartość wyjścia w mało wymagający obliczeniowo sposób, oraz w prosty sposób można modyfikować wartość opóźnienia oraz wzmocnienia modelu obiektu.

2. Regulator PID

2.1. Strojenie ciągłego regulatora PID

W celu doboru parametrów ciągłego regulatora PID wykorzystano metodę Zieglera–Nicholsa. Po pierwsze wprowadzono symulowany obiekt w niegasnące i nierosnące oscylacje, dobierając odpowiednie wzmocnienie K_k oraz ustawiając T_i mozliwie jak największe i zerując T_d. W ten sposób otrzymano wzmocnienie krytyczne K_k.

```
% Ustawienie parametrów ciągłego regulatora PID do strojenia
K_k = 0.58769;
T_i = 10000000;
T_d = 0;

% Stworzenie transmitancji regulatora PID oraz jego symulacja
PID = pidstd(K_k, T_i, T_d);
T = feedback(data.Gstf*PID, 1);
t = 0:0.01:10000;
[y, t] = step(T, t);
```



Rys. 2.1. Symulacja regulatora PID i obiektu wprowadzonego w oscylacje

Następnie analizując otrzymany przebieg za pomocą prostego skryptu w MATLABie otrzymano okres oscylacji T_k.

```
% Wyznaczenie okresu oscylacji T_k
% % Znjadowanie wierzchołków sinusoidy
for i=1:5
    y_max(i)=max(y(2000*(i-1)+1:2000*i));
end
for i=1:5
    k_max(i)=find(y==y_max(i));
end
% % Obliczanie odstępów między wierzchołkami
for i=1:4
    T_k(i)=(k_max(i+1)-k_max(i))*0.01;
end
% % Oblienie wartości T_k ze średniej odległości między wierzchołkami
T_k=mean(T_k);
```

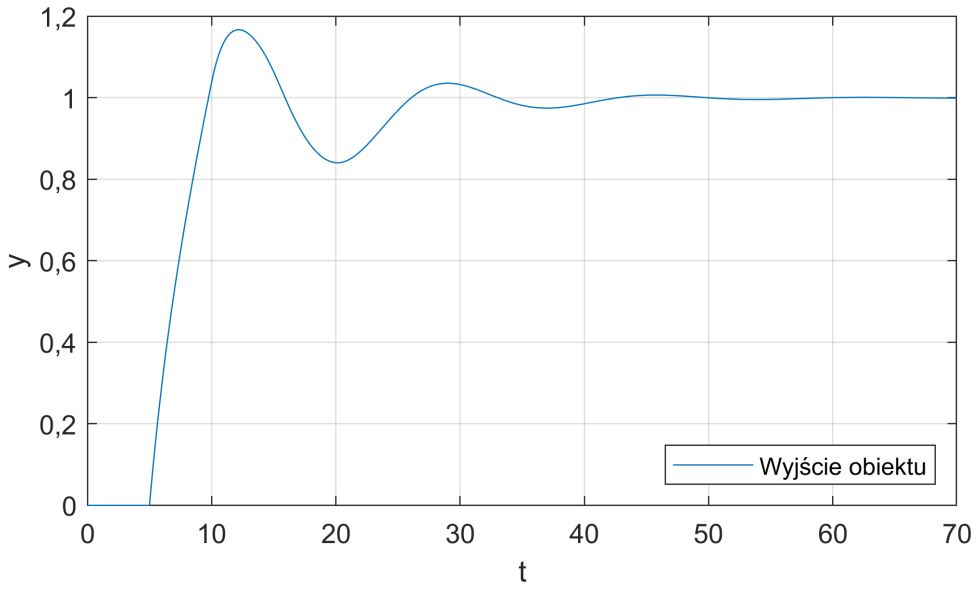
Z symulacji otrzymano:

— $K_k = 0,58769$
— $T_k = 19,625$

Korzystając z metody Zieglera–Nicholsa otrzymujemy parametry ciągłego regulatora PID.

— $K_r = 0,352614$
— $T_i = 9,8125$
— $T_d = 2,355$

```
% Obliczenie parametrów ciągłego regulatora PID
K_r=0.6*K_k;
T_i=0.5*T_k;
T_d=0.12*T_k;
```



Rys. 2.2. Symulacja dobranego ciągłego regulatora PID

2.2. Wyznaczanie parametrów dyskretnego regulatora PID

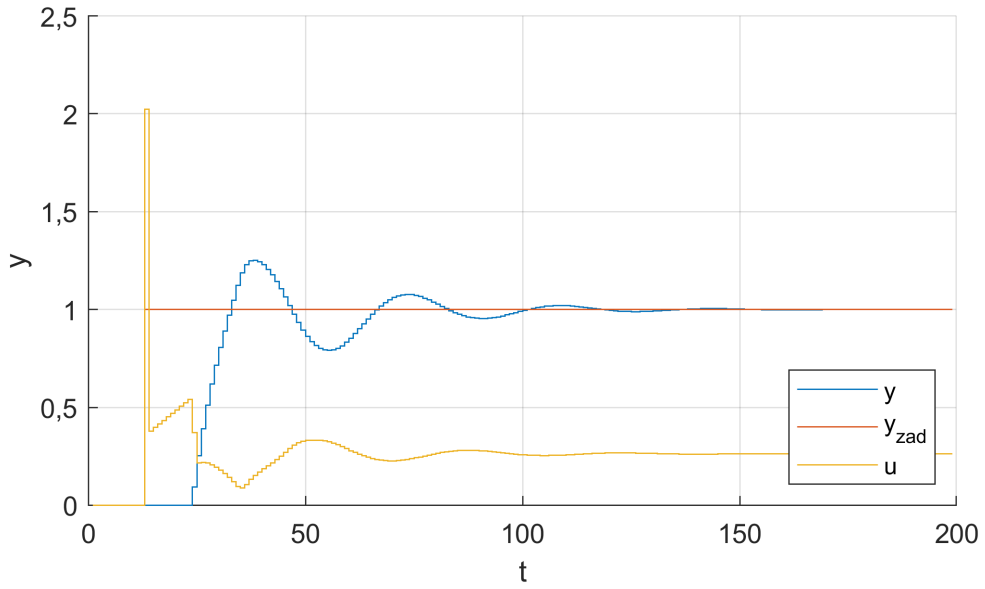
Wyznaczone powyżej parametry ciągłego regulatora PID wykorzystano w celu wyznaczenia ze wzorów parametrów r_0, r_1, r_2 dla dyskretnego regulatora.

```
% Wyznaczenie parametrów dykretnego regulatora PID
K=K_r;
T=data.T_p;

r_0=K*(1+T/(2*T_i)+T_d/T);
r_1=K*(T/(2*T_i)-2*T_d/T-1);
r_2=K*T_d/T;
```

Otrzymane parametry dyskretnego regulatora PID:

— $r_0 = 2,022409736178344$
— $r_1 = -3,665254083821656$
— $r_2 = 1,66081194$

Symulacja została przeprowadzona z wykorzystaniem symulatora regulatora PID opisanego w dalszej części sprawozdania.



Rys. 2.3. Symulacja dobranego dyskretnego regulatora PID

Wnioski

Dobrane parametry regulatora sprawiają, że regulator ten osiąga w około 150 iteracjach zadaną wartość wyjścia i maksymalna wartość wyjścia różni się o około 30% od zadanej. Porównując go z regulatorem ciągłym można dostrzec pogorszenie się działania regulatora PID.

2.3. Program symulujący

W celu stworzenia programu symulującego działanie regulatora PID wykorzystano wyznaczone wcześniej parametry r_0, r_1 i r_2 dla tego regulatora oraz równanie różnicowe 1.5 modelu pozwalające na wyznaczanie wyjścia w kolejnych chwilach k. Wykorzystanie poniżej przedstawionego symulatora zostało przedstawione na wykresie 2.3. W programie symulującym została uwzględniona możliwość zmiany wzmocnienia obiektu oraz zamiany jego opóźnienia.

```
function [k,y,y_zad,u] = pid_d_sim(r_0,r_1,r_2,b,c,k_num,y_zad_val,Tm,Km)

if nargin < 8
        Tm=1;
        Km=1;
end
delay = find(c == 0, 1, 'last');

% Zmiana opóźnienia obiektu
if Tm>1
    c=[zeros(1,int32((Tm-1)*delay)), c];
end

% Parametry n, m, T_o obiektu
m=length(c);
n=length(b);

% Warunki początkowe
k_start=max([n m])+1;
k_kon=k_start+k_num;
u(1:k_start-1)=0;
y(1:k_start-1)=0;
e(1:k_start-1)=0;
y_zad(1:k_start-1)=0;
y_zad(k_start:k_kon)=y_zad_val;


% Wyznaczanie wartości: y, e, u dla kolejnych chwil k w czasie
for k=k_start:1:k_kon
    % Wyznaczanie wartości wyjscia obiektu y
    y(k)=Km*(b*y(k-1:-1:k-n)'+c*u(k-1:-1:k-m)');
    % Wyznaczanie wartości błędu e
    e(k)=y_zad(k)-y(k);
    % Wyznaczanie wartości sygnału sterującego u
    u(k)=r_2*e(k-2)+r_1*e(k-1)+r_0*e(k)+u(k-1);
end
end
```

3. Regulator DMC

3.1. Program symulujący

W celu stworzenia programu symulującego działanie regulatora predykcyjnego DMC wykorzystano tak jak w przypadku regulatora PID wyznaczone wcześniej równanie różnicowe 1.5 modelu pozwalające na wyznaczanie wyjścia w kolejnych chwilach k oraz odpowiedzi skokowej. Wykorzystanie poniżej przedstawionego symulatora z parametrami $D = N = N_u$ zostało przedstawione na wykresach 4.4 oraz 4.3. W programie symulującym została uwzględniona możliwość zmiany wzmocnienia obiektu oraz zamiany jego opóźnienia, a także możliwe jest wywołanie skoku niemierzalnego zakłócenia.

```
function [k,y,y_zad,u,z] = dmc_d_sim(Nu,N,D,lambda,b,c,k_num, ...
    y_zad_val,Tm,Km, z_val)
if nargin < 9
    Tm=1;
    Km=1;
    z_val=0;
elseif nargin < 11
    z_val=0;
end

% Parametry n, m, T_o modelu
delay = find(c == 0, 1, 'last');
n=length(b);
m=length(c);

k_start=max([n m])+1;

% Warunki początkowe skoku
u(1:k_start-1)=0;
delta_u(1:k_start-1)=0;
y(1:k_start-1)=0;
u(k_start:k_start+D)=1;

% Wyznaczanie wartości s
for k=k_start:1:k_start+D
    y(k)=b*y(k-1:-1:k-n)'+c*u(k-1:-1:k-m)';
end

s=y(k_start+1:end)';

% Wyznaczenie Macierzy M
M=zeros(N, Nu);
for i=1:Nu
    M(i:N,i)=s(1:N-i+1);
end

% Wyznaczenie Macierzy Mp

Mp=zeros(N, D-1);
```

```
for i=1:(D-1)
    if i+N<=D
        Mp(:,i)=s(i+1:i+N);
    else
        ogr=N+i-D;
        Mp(1:N-ogr,i)=s(i+1:D);
        Mp(N-ogr+1:N,i)=ones(ogr,1).*s(D);
    end
    Mp(:,i)=Mp(:,i)-s(i);
end

% Wyznaczenie parametrów regulatora
I=eye(Nu);
K=((M'*M+lambda*I)^(-1))*M';
Ku=K(1,:)*Mp;
Ke=sum(K(1,:));


% Zmiana opóźnienia obiektu
if Tm>1
    c=[zeros(1,int32((Tm-1)*delay)), c];
end
m=length(c);

% Warunki początkowe symulacji
k_start=max([k_start,m,D]);
k_kon=k_start+k_num;
u(1:k_kon)=0;
delta_u(1:k_kon)=0;
y(1:k_start-1)=0;
e(1:k_start-1)=0;
y_zad(1:k_start-1)=0;
y_zad(k_start:k_kon)=y_zad_val;
delta_u=zeros(k_kon,1);
y=zeros(1,k_kon);
z(1:k_start-1)=0;
z(k_start:k_kon)=z_val;

% Obliczanie wartości: y,e,u dla kolejnych chwil czasu
for k=k_start:1:k_kon
    % Wartość wyjścia w chwili k
    y(k)=Km*(b*y(k-1:-1:k-n)'+c*u(k-1:-1:k-m)')+z(k);
    % Wartość błędu w chwili k
    e(k)=y_zad(k)-y(k);
    % Zmiana u w chwili k
    delta_u(k)=Ke*e(k)-Ku*delta_u(k-1:-1:k-D+1);
    % Wartość sterowania w chwili uk
    u(k)=u(k-1)+delta_u(k);
end
y=y(k_start-5:k_kon);
y_zad=y_zad(k_start-5:k_kon);
u=u(k_start-5:k_kon);
z=z(k_start-5:k_kon);
end
```

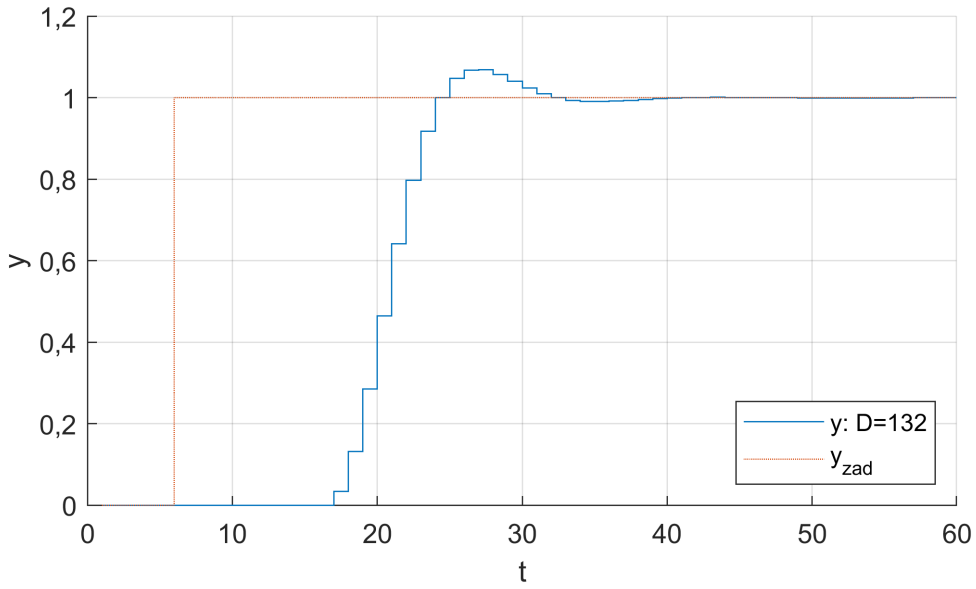
3.2. Strojenie regulatora

3.2.1. Dobór horyzontu dynamiki D

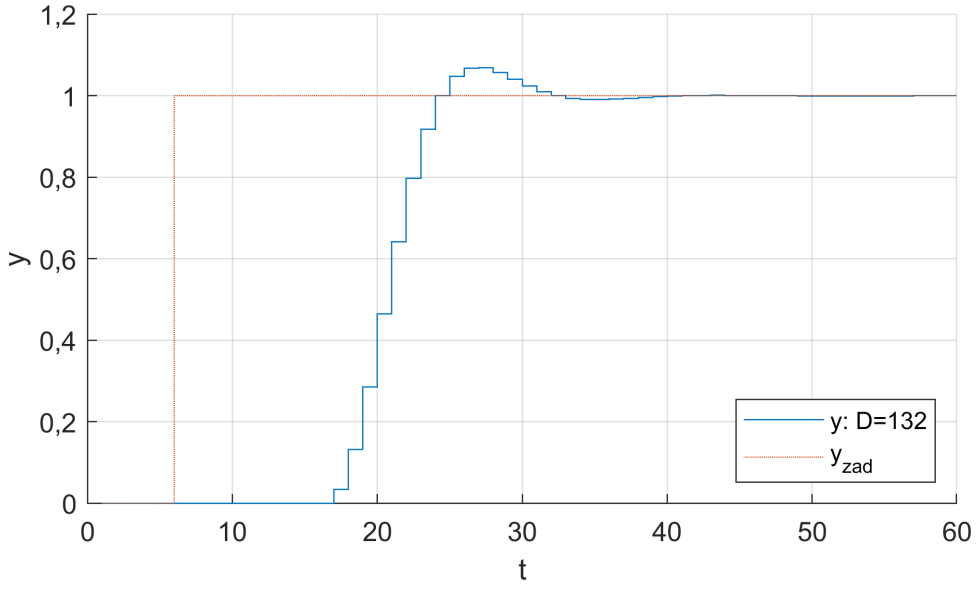
Horyzont dynamiki D został wybrany na podstawie wykresu odpowiedzi skokowej 1.1, a dokładniej dobrany korzystając z wartości przyjmowanych przez wektor s w programie symulującym. Ostatecznie dobrana wartość to:

$$D = 132$$

Dla przeprowadzonych symulacji podczas dobierania D pozostałe parametry zostały przyjęte jako $D = N = N_u$ oraz $\lambda = 1$.



Rys. 3.1. Wyjście obiektu dla regulatora DMC: $D = N = N_u$ oraz $\lambda = 1$



Rys. 3.2. Wejście obiektu dla regulatora DMC: $D = N = N_u$ oraz $\lambda = 1$

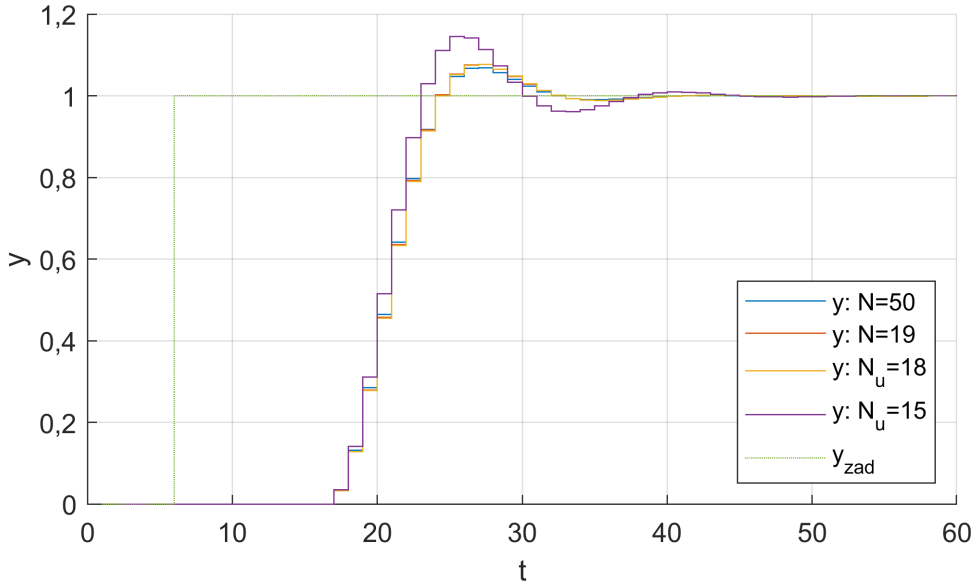
3.2.2. Dobór horyzontu predykcji N

W celu dobrania odpowiednio krótkiego horyzontu predykcji N został przeprowadzony szereg symulacji dla różnych wartości N (na wykresach przedstawiono $N \in \{50, 19, 18, 15\}$) na podstawie, których zdecydowano, że najlepszą regulację, przy jednocześnie możliwie małym nakładzie obliczeń zapewni:

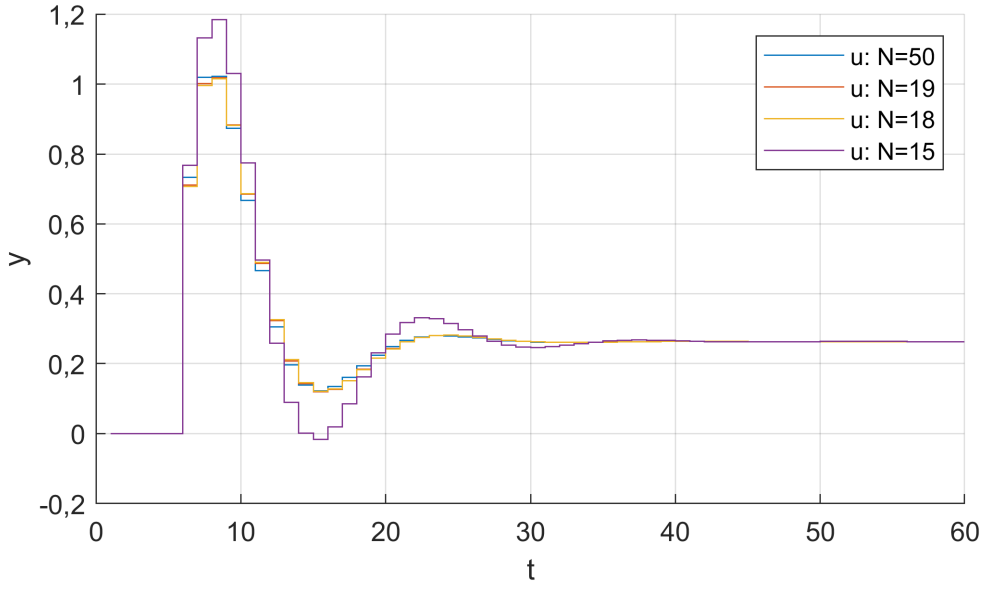
$$N = 132$$

Dodatkowo zauważono, że przy zmniejszaniu wartości N, nie ma to większego znaczenia dla całego procesu regulacji, i dopiero gdy zaczęto zbliżać się wartością N do opóźnienia obiektu, w tedy zaczęły pojawiać się problemy z jakością regulacji.

Dla przeprowadzonych symulacji podczas dobierania N pozostałe parametry zostały przyjęte jako $D = 132$, $N = N_u$ oraz $\lambda = 1$.



Rys. 3.3. Wyjście obiektu dla regulatora DMC dla różnych wartości N



Rys. 3.4. Wejście obiektu dla regulatora DMC dla różnych wartości N

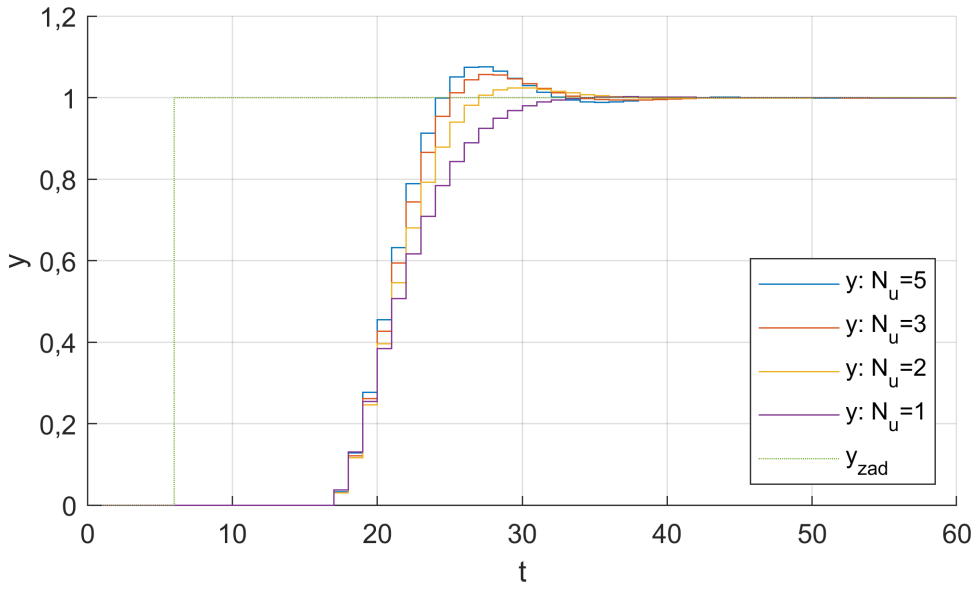
3.2.3. Dobór horyzontu sterowania N_u

W celu dobrania odpowiedniego horyzontu sterowania N_u został przeprowadzony szereg symulacji dla różnych wartości N_u (na wykresach przedstawiono $N_u \in \{5, 3, 2, 1\}$) na podstawie, których zdecydowano, że najlepszą regulację, najmniejsze przeregulowanie oraz możliwie małą zmianę wartości sygnału sterującego:

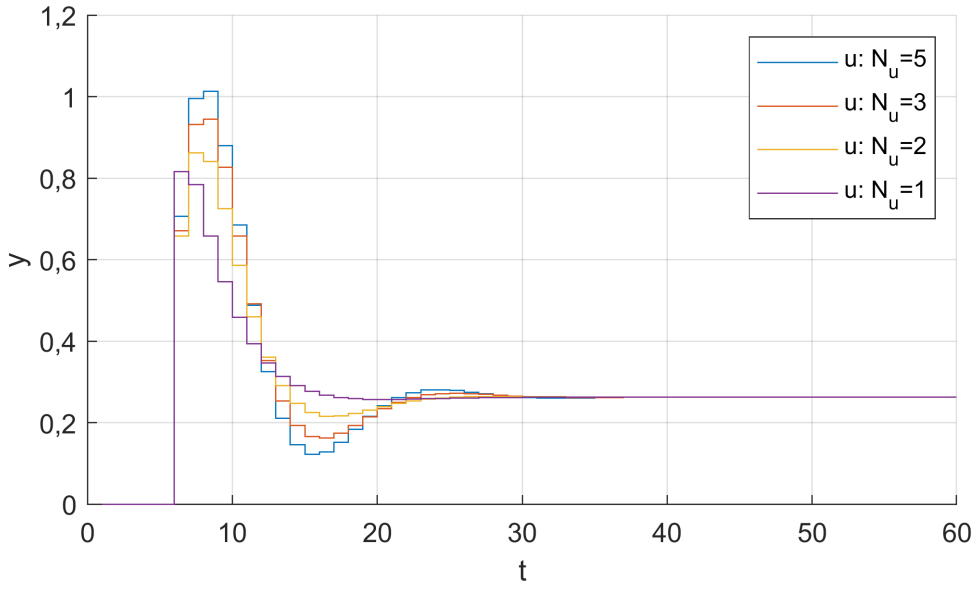
$$N_u = 2$$

Dodatkowo zauważono, że przy zmniejszaniu wartości N_u zmniejsza się przeregulowanie oraz zwiększa się maksymalna zmiana sygnału sterującego Δu. Jedynie dla $N_u = 1$ otrzymano wyjście które prawie nie ma przeregulowania, jednakże skok sygnału sterującego był nieakceptowalny i dla wielu fizycznych obiektów nierealny do osiągnięcia dlatego zdecydowano się na podaną powyżej wartość N_u.

Dla przeprowadzonych symulacji podczas dobierania N_u pozostałe parametry zostały przyjęte jako $D = 132$, $N = 19$ oraz $\lambda = 1$.



Rys. 3.5. Wyjście obiektu dla regulatora DMC dla różnych wartości N_u



Rys. 3.6. Wejście obiektu dla regulatora DMC dla różnych wartości N_u

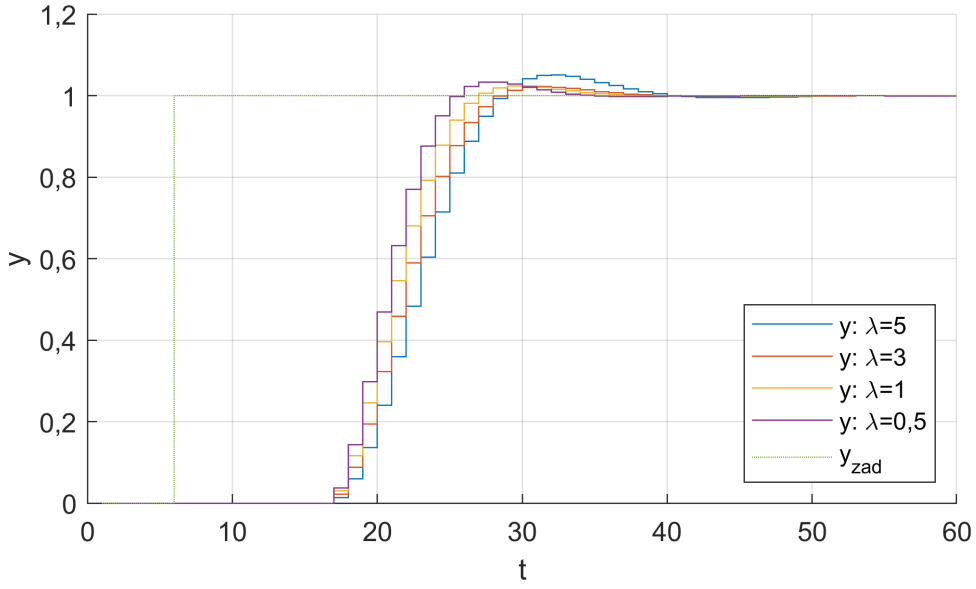
3.2.4. Dobór horyzontu sterowania λ

W celu dobrania odpowiedniego horyzontu sterowania λ został przeprowadzony szereg symulacji dla różnych wartości λ (na wykresach przedstawiono $\lambda \in \{0, 5, 1, 3, 10\}$) na podstawie, których zdecydowano, że najlepszą regulację, najmniejsze przeregulowanie oraz możliwie małą zmianę wartości sygnału sterującego:

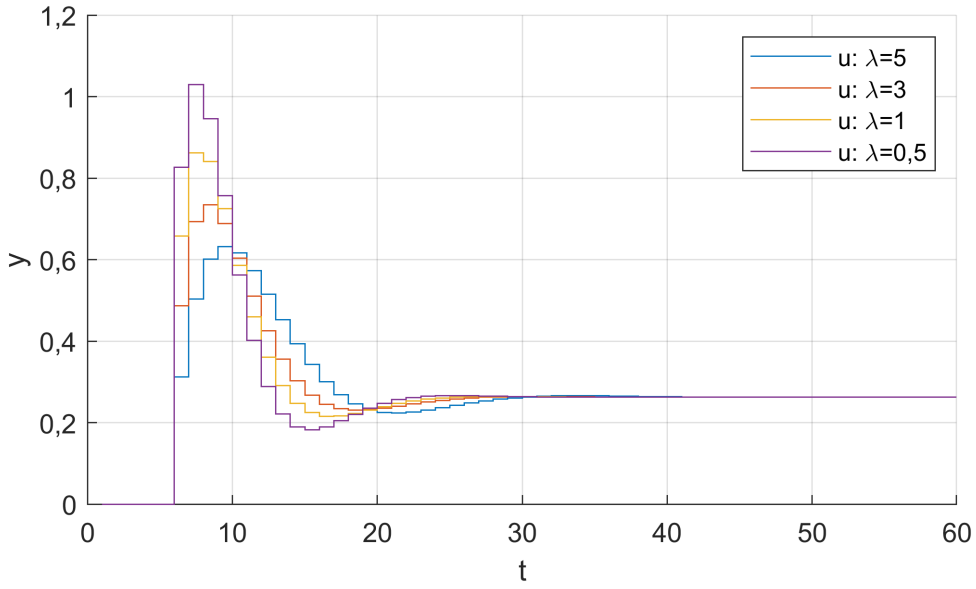
$$\lambda = 3$$

Dodatkowo zauważono, że przy zmniejszaniu wartości λ zwiększa się przeregulowanie oraz zwiększa się maksymalna zmiana sygnału sterującego Δu, za to przy zwiększaniu wartości λ zmiana sygnału się zmniejsza, i do momentu, aż zostanie osiągnięta wartość $\lambda = 3$ przeregulowanie się zmniejszało, a dla jeszcze większych wartości ponownie zwiększało się ono.

Dla przeprowadzonych symulacji podczas dobierania λ pozostałe parametry zostały przyjęte jako $D = 132$, $N = 19$, $N_u = 2$.



Rys. 3.7. Wyjście obiektu dla regulatora DMC dla różnych wartości λ



Rys. 3.8. Wejście obiektu dla regulatora DMC dla różnych wartości λ

3.2.5. Podsumowanie

Ostatecznie dobrane parametry regulatora DMC to:

$$dvsv \tag{3.1}$$

$$a + b = c \tag{3.2}$$

$$e = mc^2 \tag{3.3}$$

4. Regulator GPC

4.1. Symulator dyskretnego regulatora GPC

W celu stworzenia programu symulującego działanie regulatora predykcyjnego GPC wykorzystano tak jak w przypadku regulatora PID oraz DMC wyznaczone wcześniej równanie różnicowe 1.5 modelu pozwalające na wyznaczanie wyjścia w kolejnych chwilach k, odpowiedzi skokowej oraz swobodnej odpowiedzi skokowej. Wykorzystanie poniżej przedstawionego symulatora z parametrami $D = N = N_u$ zostało przedstawione na wykresach **??** oraz **??**. W programie symulującym została uwzględniona możliwość zmiany wzmocnienia obiektu oraz zamiany jego opóźnienia, a także możliwe jest wywołanie skoku niemierzalnego zakłócenia.

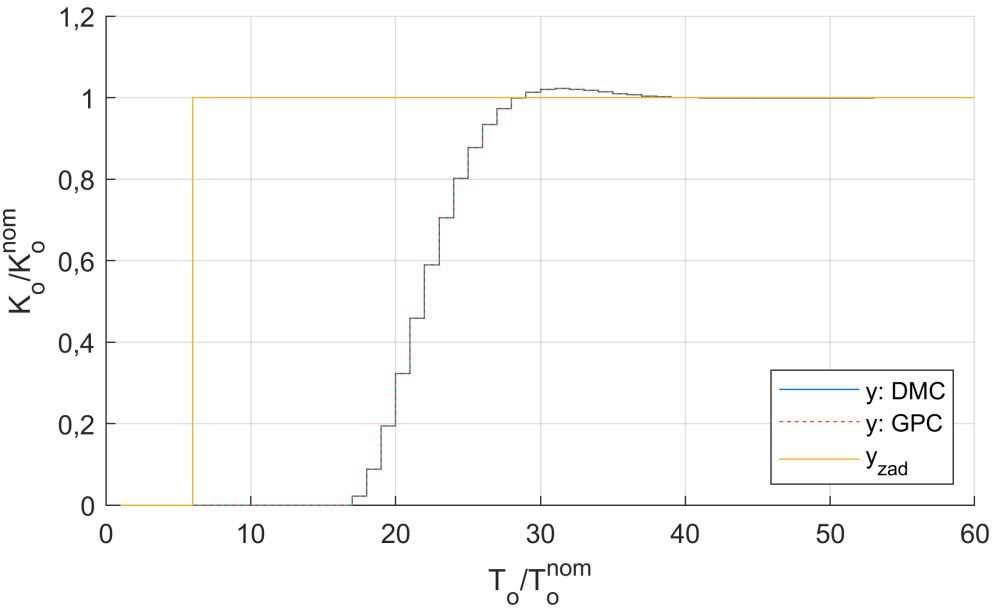
```
function [k,y,y_zad,u,z] = gpc_d_sim(Nu,N,lambda,b,c,k_num, ...
    y_zad_val,Tm,Km,z_val)

if nargin < 8
    Tm=1;
    Km=1;
    z_val=0;
elseif nargin < 10
    z_val=0;
end

% Parametry n, m, T_o modelu
delay = find(c == 0, 1, 'last');
n=length(b);
m=length(c);

% Warunki początkowe skoku
k_start=max([n m])+1;
u(1:k_start-1)=0;
delta_u(1:k_start-1)=0;
y(1:k_start-1)=0;
u(k_start:k_start+N)=1;

% Wyznaczanie wartości s
for k=k_start:1:k_start+N
    y(k)=b*y(k-1:-1:k-n)'+c*u(k-1:-1:k-m)';
end

s=y(k_start+1:end)';

% Wyznaczenie Macierzy M
M=zeros(N, Nu);
for i=1:Nu
    M(i:N,i)=s(1:N-i+1);
end


% Wyznaczenie parametrów regulatora
I=eye(Nu);
```

```
K=((M'*M+lambda*I)^(-1))*M';


% Zmiana opóźnienia obiektu
if Tm>1
    c_o=[zeros(1,int32((Tm-1)*delay)), c];
else
    c_o=c;
end
m_o=length(c_o);


% Warunki początkowe symulacji
k_start=max(k_start,m_o+1);
k_kon=k_start+k_num;
u(1:k_kon)=0;
delta_u(1:k_kon)=0;
y(1:k_start-1)=0;
e(1:k_start-1)=0;
y_zad(1:k_start-1,1)=0;
y_zad(k_start:k_kon+N,1)=y_zad_val;
delta_u=zeros(k_kon,1);
y=zeros(1,k_kon);
d=zeros(1,k_kon);
up=zeros(1,N);
yp=zeros(1,N);
y0=zeros(N,1);
z(1:k_start-1)=0;
z(k_start:k_kon)=z_val;


% Obliczanie wartości: y,e,u dla kolejnych chwil czasu
for k=k_start:1:k_kon

    % Wartość wyjścia w chwili k
    y(k)=Km*(b*y(k-1:-1:k-n)'+c_o*u(k-1:-1:k-m_o)');
    y(k)=y(k)+z(k);

    % Wyznaczanie odpowiedzi swobodnej y_0(k)
    d(k)=y(k)-(b*y(k-1:-1:k-n)'+c*u(k-1:-1:k-m)');
    for p=1:1:N
        up(p)=u(k-1);
        u(k+p-1)=up(p);

        yp(p)=b*y(k-1+p:-1:k-n+p)'+c*u(k-1+p:-1:k-m+p)';
        y(k+p)=yp(p);

        y0(p)=yp(p)+d(k);
    end
    disp(y_zad(k+1:k+N));
    delta_u(k)=K(1,:)*(y_zad(k+1:k+N)-y0);

    % Wartość sterowania w chwili uk
    u(k)=u(k-1)+delta_u(k);
end
y=y(k_start-5:k_kon);
y_zad=y_zad(k_start-5:k_kon);
u=u(k_start-5:k_kon);
```

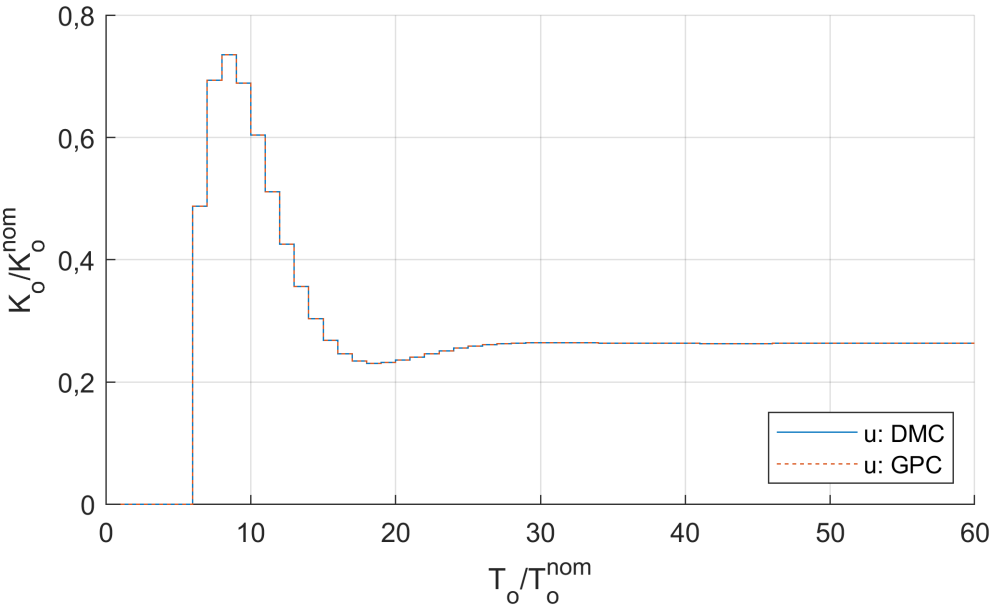
```
z=z(k_start-5:k_kon);
end
```

4.2. Porównanie GPC z DMC

4.2.1. Skok wartości zadanej

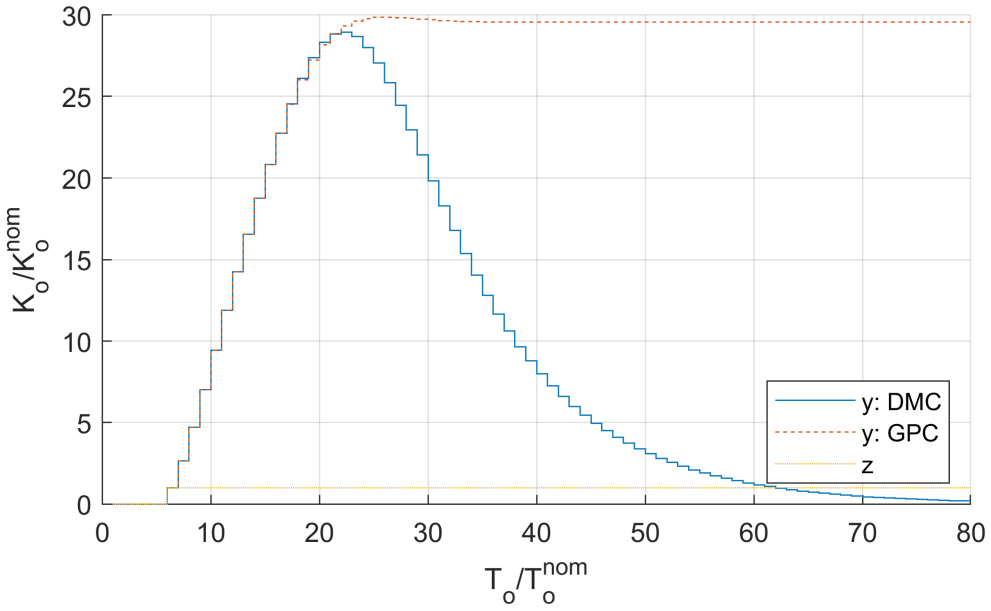


Rys. 4.1. Porównanie przebiegów y DMC i GPC dla skoku wartości zadanej

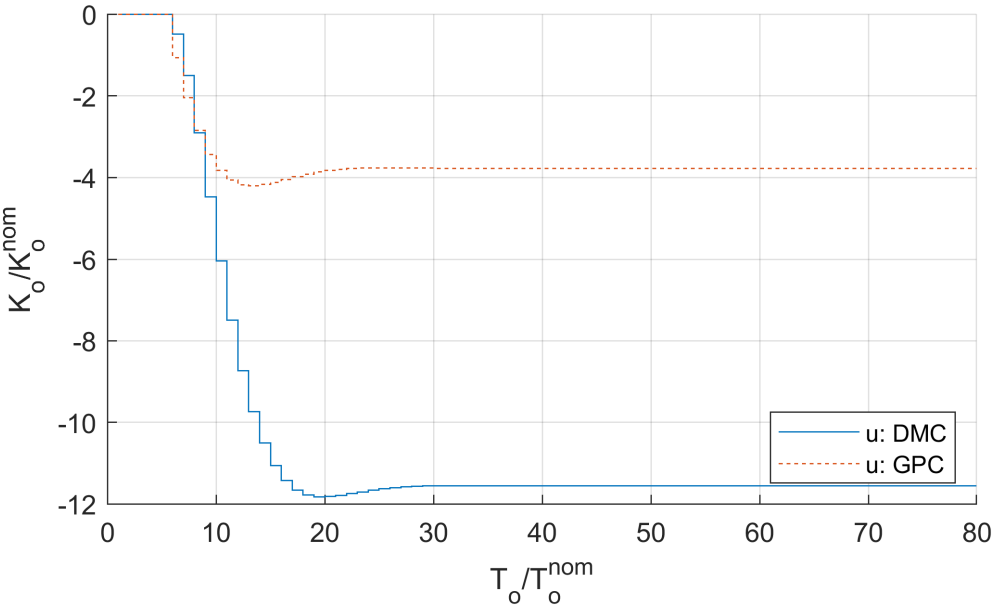


Rys. 4.2. Porównanie przebiegów u DMC i GPC dla skoku wartości zadanej

4.2.2. Skok niemierzalnych zakłóceń



Rys. 4.3. Porównanie przebiegów y DMC i GPC dla skoku niemierzalnych zakłóceń



Rys. 4.4. Porównanie przebiegów u DMC i GPC dla skoku niemierzalnych zakłóceń

4.2.3. Podsumowanie

Dla skoku wartości zadanej otrzymane przebiegi y oraz u dla obu regulatorów są identyczne, jednakże w przypadku skoku zakłóceń występuje uchyb ustalony spowodowany prawdopodobnie przez błąd w implementacji regulatora GPC, którego niestety nie udało się znaleźć oraz naprawić.

5. Obszary stabilności regulatorów

Do określenia obszarów stabilności każdego z regulatorów sporządzono 3 krzywe $\frac{K_o}{K_o^{nom}}$ w funkcji $\frac{T_o}{T_o^{nom}}$.

5.1. Wyznaczanie obszarów stabilności

Dla każdego z regulatorów przeprowadzono symulacje polegające na zmianie wzmocnienia obiektu dla $\frac{T_o}{T_o^{nom}} \in \{1; 1,1; 1,2; 1,4; 1,5; 1,6; 1,7; 1,8; 1,9; 2\}$. Dla każdego regulatora wykorzystano podobne do siebie programy.

5.1.1. PID

```
clear all
close all
data=dane();

r_0=2.022409736178344;
r_1=-3.665254083821656;
r_2=1.66081194;

Tm=linspace(1,2,11);
Km=ones(1,11);
Km(1)=1.030785;
Km(2)=1.027139;
Km(3)=1.023373;
Km(4)=1.019702;
Km(5)=1.016304;
Km(6)=1.013277;
Km(7)=1.010633;
Km(8)=1.008343;
Km(9)=1.006353;
Km(10)=1.004614;
Km(11)=1.003078;

for i=1:11
    [k,y,y_zad,u]=pid_d_sim(r_0,r_1,r_2,data.b,data.c, ...
        10000,1,Tm(i),Km(i));
    figure;
    hold on;
    stairs(y);
    stairs(y_zad);
    grid on;
    pbaspect([16 9 1]);
    ylabel('y');
    xlabel('k');
end

figure;
```

```
plot(Tm,Km,'-*');
pbaspect([16 9 1]);
grid on;
ylabel('K_o/K_{o}^{nom}');
xlabel('T_o/T_{o}^{nom}');
etykiety = get(gca,'YTickLabel');
etykiety = strrep(etykiety (:),'.',',');
set(gca,'YTickLabel',etykiety);
etykiety = get(gca,'XTickLabel');
etykiety = strrep(etykiety (:),'.',',');
set(gca,'XTickLabel',etykiety);
legend("krzywa stabilności PID",Location="southeast");
exportgraphics(gcf,'wykresy/stabilnosc_PID.png','Resolution',400);
```

5.1.2. DMC

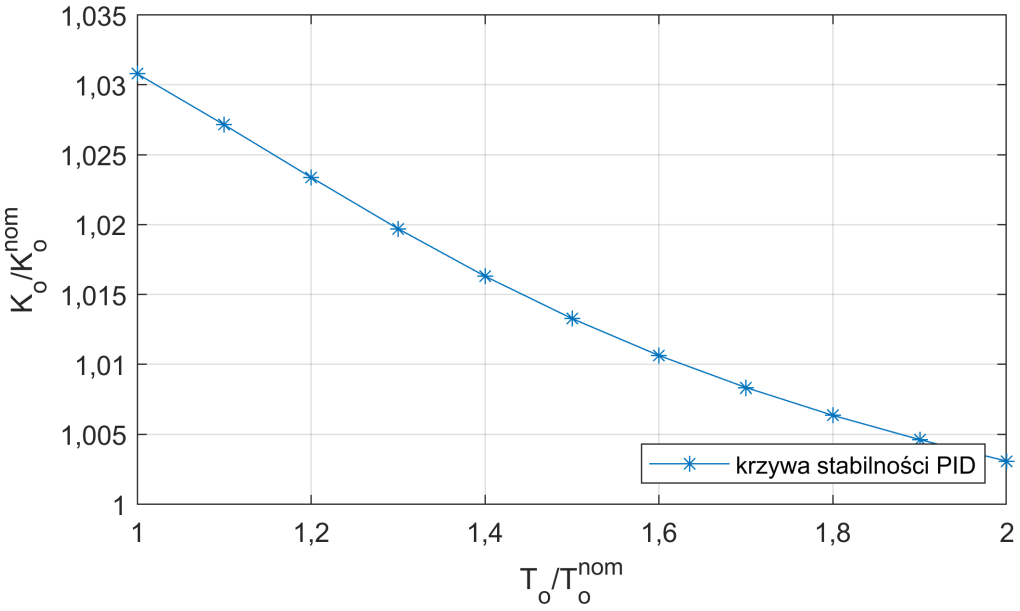
```
clear all
close all
data=dane();

Tm=linspace(1,2,11);
Km=ones(1,11);
Km(1)=1.026545;
Km(2)=1.024261;
Km(3)=1.021976;
Km(4)=1.019729;
Km(5)=1.017559;
Km(6)=1.015494;
Km(7)=1.013551;
Km(8)=1.011742;
Km(9)=0.93365;
Km(10)=0.90245;
Km(11)=0.88508;

for i=1:11
    [k, y, y_zad, u] = dmc_d_sim(2,19,132,3,data.b,data.c, ...
        10000,1,Tm(i),Km(i));
    figure;
    hold on;
    stairs(y);
    stairs(y_zad);
    grid on;
    pbaspect([16 9 1]);
    ylabel('y');
    xlabel('k');
end

figure;
plot(Tm,Km,'-*');
grid on;
pbaspect([16 9 1]);
ylabel('K_o/K_{o}^{nom}');
xlabel('T_o/T_{o}^{nom}');
etykiety = get(gca,'YTickLabel');
etykiety = strrep(etykiety (:),'.',',');
set(gca,'YTickLabel',etykiety);
etykiety = get(gca,'XTickLabel');
```

```
etykiety = strrep(etykiety (:),'.',',');
set(gca,'XTickLabel',etykiety);
legend("krzywa stabilności DMC",Location="southeast");
exportgraphics(gcf,'wykresy/stabilnosc_DMC.png','Resolution',400);
```

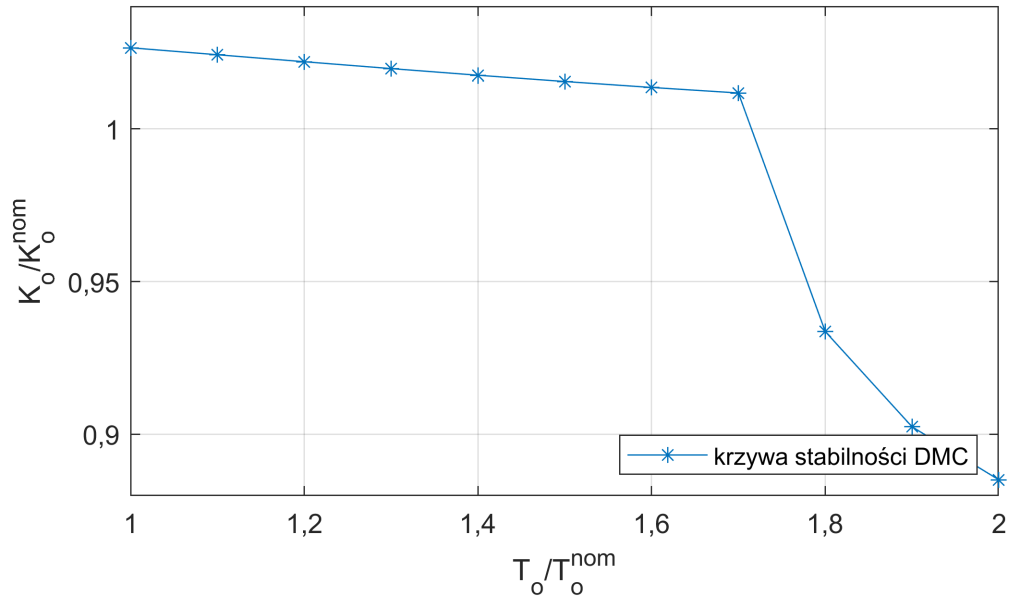
5.1.3. GPC

```
clear all
close all
data=dane();

Tm=linspace(1,2,11);
Km=ones(1,11);
Km(1)=1;
Km(2)=1;
Km(3)=1;
Km(4)=1;
Km(5)=1;
Km(6)=1;
Km(7)=1;
Km(8)=1;
Km(9)=1;
Km(10)=1;
Km(11)=1;

for i=1:11
    [k, y, y_zad, u] = gpc_d_sim(2,19,1,data.b,data.c, ...
        1000,1,Tm(i),Km(i));
    figure;
    hold on;
    stairs(y);
    stairs(y_zad);
    grid on;
    pbaspect([16 9 1]);
    ylabel('y');
    xlabel('k');
    legend("GPC","y_{zad}",Location="southeast");
end

figure;
plot(Tm,Km,'-*');
pbaspect([16 9 1]);
ylabel('K_o/K_{o}^{nom}');
xlabel('T_o/T_{o}^{nom}');
etykiety = get(gca,'YTickLabel');
etykiety = strrep(etykiety (:),'.',',');
set(gca,'YTickLabel',etykiety);
etykiety = get(gca,'XTickLabel');
etykiety = strrep(etykiety (:),'.',',');
set(gca,'XTickLabel',etykiety);
legend("krzywa stabilności GPC",Location="southeast");
exportgraphics(gcf,'wykresy/stabilnosc_GPC.png','Resolution',400);
```

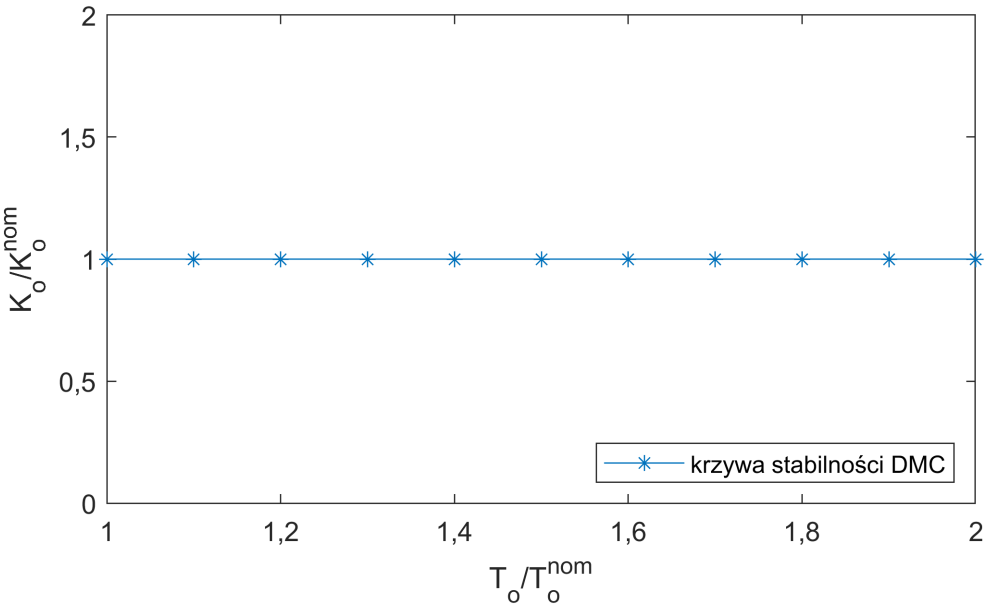
5.2. Krzywe stabilności



Rys. 5.1. Krzywa stabilności regulatora PID



Rys. 5.2. Krzywa stabilności regulatora DMC



Rys. 5.3. Krzywa stabilności regulatora GPC

5.3. Porównanie

Jak można zauważyć regulator PID ma lepszy zakres stabilności w przypadku, gdy opóźnienie obiektu jest mniejsze, a w przypadku gdy to opóźnienie się zwiększa to i jego krzywa stabilności spada coraz bardziej, aż $\frac{T_o}{T_o^{nom}} = 1,4$ regulator DMC okazuje się bardziej stabilny. Jednakże w przypadku regulatora DMC dochodzi do interesującego załamania krzywej stabilności dla $\frac{T_o}{T_o^{nom}} = 1,8$. Po przeprowadzeniu dodatkowych doświadczeń wywnioskowane, że jest to spowodowane przez zbyt mały horyzont sterowania regulatora DMC dla większych opóźnień. Gdyby była potrzeba otrzymania lepszej elastyczności wykorzystania tego regulatora należałoby zwiększyć wartość horyzontu sterowania N. Niestety z powodu prawdopodobnego błędu w implementacji regulatora GPC nie udało się sprawdzić odpowiednio jego stabilności. Jedyne co zaobserwowano, to że wraz rosnącym opóźnieniem obiektu, coraz dłużej zajmowało regulatorowi osiągnięcie wartości zadanej.