

**Wydział Elektroniki i Technik Informacyjnych
Politechnika Warszawska**

Sterowanie Procesami

Sprawozdanie z projektu nr 2, zadanie nr 28

Michał Paradowski

Warszawa, 2025

Spis treści

Wstęp	2
Zadanie 1	3
0.1. Wyznaczanie transmitancji dyskretniej	3
0.2. Porównanie transmitancji ciągłej i dyskretniej	3
Zadanie 1 – Podsumowanie	4
Zadanie 2	5
0.3. Wyznaczanie równania różnicowego	5
Zadanie 2 – Podsumowanie	5
Zadanie 3	6
0.4. Wyznaczenie wzmocnienia krytycznego oraz okresu oscylacji	6
0.5. Wyznaczenie parametrów regulatorów PID	7
Zadanie 3 – Podsumowanie	7
Zadanie 4	8
0.6. Symulacja cyfrowego algorytmu PID	8
0.7. Symulacja algorytmu DMC	8
Zadanie 4 – Podsumowanie	9
Zadanie 5	10
a) Początkowe parametry DMC	10
b) Skracanie horyzontu predykcji	11
c) Skracanie horyzontu regulacji	12
d) Manipulacja wartością parametru λ	14
Zadanie 5 – Podsumowanie	15
Zadanie 6	16
0.8. Porównanie regulatorów DMC oraz cyfrowego PID	16
0.9. Krzywe stabilności regulatora DMC oraz cyfrowego PID	16
Zadanie 6 – Podsumowanie	20
Zadanie 7	21
0.10. Deklaracja parametrów modelu oraz algorytmu	21
0.11. Wektor s , macierze M oraz K	21
0.12. Przebieg działania algorytmu	22
Zadanie 7 – Podsumowanie	22
Zadanie 8	23
0.13. a) Porównanie algorytmów GPC oraz DMC przy skokowej zmianie wartości zadanej	23
0.14. b) Porównanie algorytmów GPC oraz DMC przy skokowej zmianie wartości zakłócenia	23
Zadanie 8 – Podsumowanie	24
zadanie 9	25
0.15. Krzywa stabilności algorytmu GPC	25
Zadanie 9 – Podsumowanie	27
Wnioski końcowe	28

Wstęp

Celem niniejszego projektu było zaprojektowanie i porównanie dwóch cyfrowych układów regulacji – klasycznego algorytmu PID oraz nowoczesnego regulatora DMC (Dynamic Matrix Control), z wykorzystaniem modelu obiektu opisanego transmitancją:

$$G(s) = \frac{K_0 e^{-T_0 s}}{(T_1 s + 1)(T_2 s + 1)} \quad (1)$$

gdzie $K_0 = 3,5$, $T_0 = 5$, $T_1 = 2,16$, $T_2 = 5,52$.

W części rozszerzonej projektu opracowano implementację algorytmu GPC oraz dokonano porównania jego działania z algorytmem DMC. Całość realizacji wykonano w środowisku MATLAB.

Zadanie 1

0.1. Wyznaczanie transmitancji dyskretnej

Aby zastosować ekstrapolator zerowego rzędu do uzyskania transmitancji dyskretnej, wykorzystano poniższy skrypt MATLAB

```
K0 = 3.5;  
T0 = 5;  
T1 = 2.16;  
T2 = 5.52;  
Tp = 0.5;  
  
s = tf('s');  
G_s = K0 * exp(-T0*s) / ((T1*s+1)*(T2*s+1));  
  
% Dyskretyzacja transmitancji (ZOH)  
Gz = c2d(G_s, Tp, 'zoh');
```

Otrzymano transmitancję dyskretną:

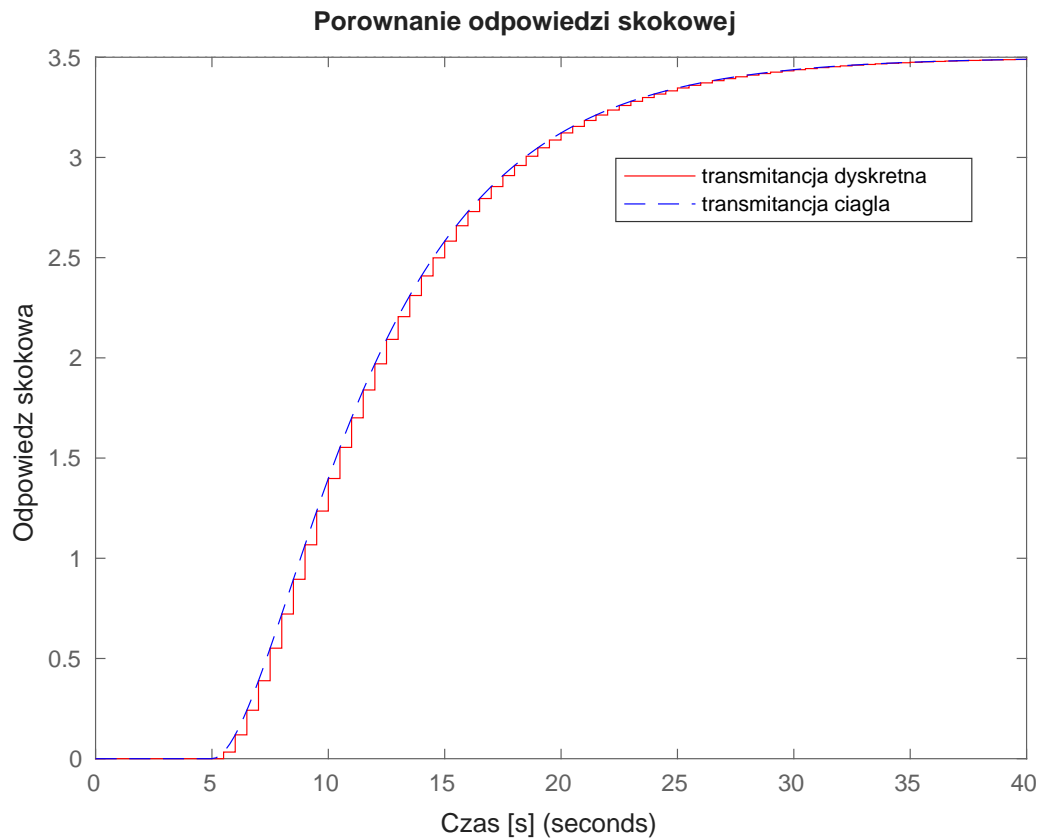
$$G(z) = z^{-10} \cdot \frac{0,033z + 0,029\,64}{z^2 - 1,707z + 0,7247} \quad (2)$$

0.2. Porównanie transmitancji ciągłej i dyskretnej

Aby porównać transmitancje ciągłą i dyskretną wykreślono ich odpowiedzi skokowe za pomocą funkcji MATLAB:

```
step(G_s, 'b', Gz, 'r--');
```

Otrzymane przebiegi zamieszczono na wykresie 1



Rys. 1. Porównanie odpowiedzi skokowej transmitancji ciągłej i dyskretniej

Zgodnie z oczekiwaniami, transmitancje odpowiadają na skok sygnału sterowania w tożsamy sposób.

Współczynniki wzmocnienia statycznego obu transmitancji wyznaczono za pomocą funkcji MATLAB:

```
K_stat_s = dcgain(G_s);  
K_stat_z = dcgain(Gz);
```

Oba wzmocnienia wyniosły dokładnie 3,5.

Zadanie 1 – Podsumowanie

Z powodzeniem przeprowadzono proces dyskretyzacji transmitancji ciągłej przy użyciu ZOH oraz porównano odpowiedzi skokowe i wzmocnienia statyczne obu modeli. Dyskretna reprezentacja dobrze odwzorowuje właściwości układu ciągłego przy zadanym okresie próbkowania.

Zadanie 2

0.3. Wyznaczanie równania różnicowego

Na podstawie transmitancji dyskretnej wyznaczono równanie różnicowe służące do obliczenia wielkości $y(k)$ na podstawie sygnałów wejściowych i wyjściowych z chwil poprzednich.

$$y(k) = -a_1y(k-1) - a_2y(k-2) + b_{11}u(k-11) + b_{12}u(k-12) \quad (3)$$

Indeksy 11, 12 przy parametrach b_i wynikają z opóźnienia w transmitancji (2) z^{-10} . Współczynniki a_1, a_2, b_{11}, b_{12} odczytano z transmitancji 2

$$G(z) = z^{-10} \cdot \frac{0,033z + 0,029\,64}{z^2 - 1,707z + 0,7247} = \frac{b_{11}z^{-11} + b_{12}z^{-12}}{z^2 + a_1z + a_2} \quad (4)$$

Otrzymano

$$y(k) = 1,707y(k-1) - 0,7247y(k-2) + 0,033u(k-11) + 0,029\,64u(k-12) \quad (5)$$

Do wyznaczenia parametrów wykorzystano skrypt MATLAB:

```
[b, a] = tfdata(Gz, 'v');  
b11 = b(2);  
b12 = b(3);  
a1 = a(2);  
a2 = a(3);
```

Zadanie 2 – Podsumowanie

Na podstawie dyskretnej transmitancji wyznaczono poprawne równanie różnicowe, które posłużyło do dalszej implementacji modelu w regulatorach. Równanie przygotowano zgodnie z ogólną strukturą modeli wejście-wyjście.

Zadanie 3

Do danego obiektu (1) dobrano ciągły regulator PID metodą Ziegler-Nicholsa.

0.4. Wyznaczenie wzmocnienia krytycznego oraz okresu oscylacji

Do wyznaczenia wzmocnienia krytycznego K_k oraz okresu oscylacji T_k wykorzystano skrypt:

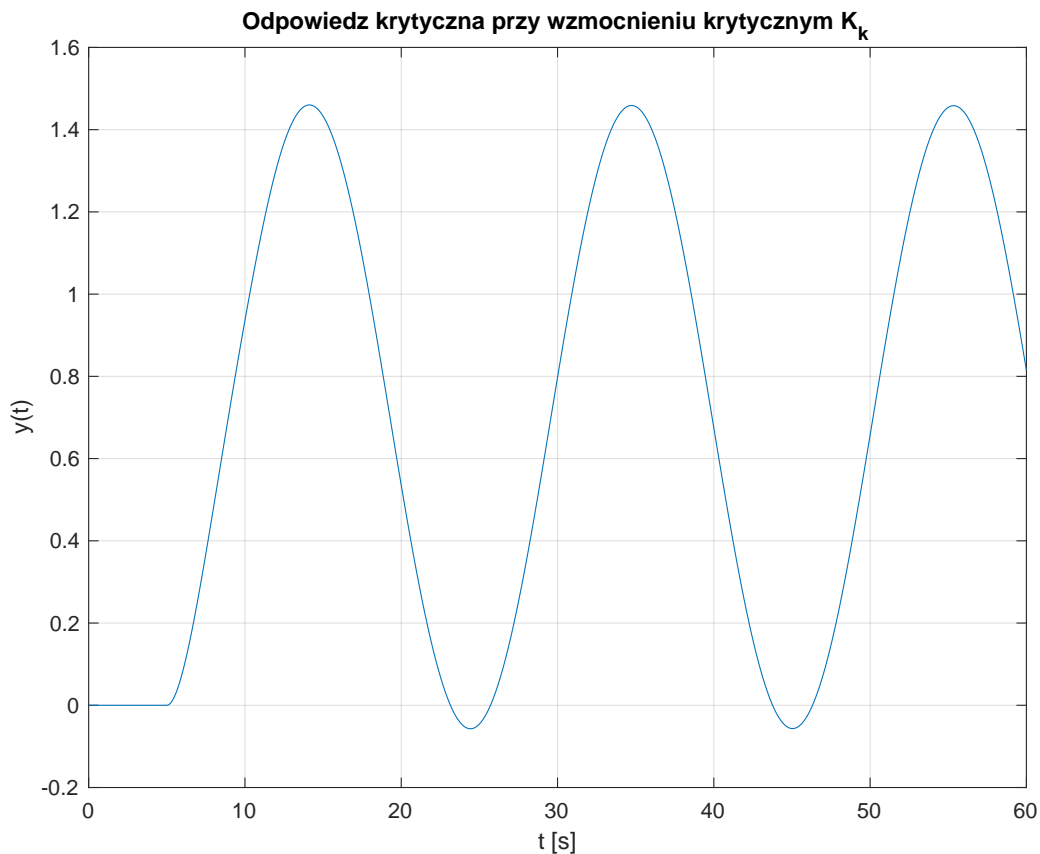
```
[Gm, Pm, Wcg, Wcp] = margin(G_s);  
Kk = Gm;  
Tk = 2*pi/Wcg;
```

Otrzymano $K_k = 0,6698$, $T_k = 20,6072s$.

Aby zweryfikować te rezultaty zasymulowano odpowiedź obiektu za pomocą skryptu MATLAB:

```
t = 0:0.1:60;  
y = step(feedback(Kk*G_s, 1), t);
```

a wynik symulacji przedstawiono na wykresie 2



Rys. 2. Odpowiedź krytyczna przy wzmocnieniu krytycznym K_k

Jak widać na wykresie 2, otrzymane wzmocnienie jest krytyczne, a okres $T_k \approx 21s$ również jest poprawny.

0.5. Wyznaczenie parametrów regulatorów PID

Na podstawie wzmocnienia krytycznego K_k oraz okresu oscylacji T_k wyznaczono parametry ciągłego regulatora PID:

- $K_r = 0,6K_k = 0,4019$
- $T_i = 0,5T_k = 10,3036s$
- $T_d = 0,12T_k = 2,4729s$

określonego wzorem

$$u(t) = u_0 + K_p \left(e(t) + \frac{1}{T_i} \int_0^t e(\tau) d\tau + T_d \frac{de(t)}{dt} \right) \quad (6)$$

Następnie wyznaczono parametry dyskretnego regulatora PID:

- $r_2 = K_r \cdot \frac{T_d}{T_p} = 1,9862$
- $r_1 = K_r \left(\frac{T_p}{2T_i} - \frac{2T_d}{T_p} - 1 \right) = -4,0643$
- $r_0 = K_r \left(1 + \frac{T_p}{2T_i} + \frac{T_d}{T_p} \right) = 2,6516$

określonego wzorem

$$u(t) = \frac{r_2 z^{-2} + r_1 z^{-1} + r_0}{1 - z^{-1}} e(k) \quad (7)$$

Wykorzystano w tych celach skrypt MATLAB:

```
K = 0.6 * Kk;
Ti = 0.5 * Tk;
Td = 0.12 * Tk;
r2 = K*Td/Tp;
r1 = K*(Tp/(2*Ti) - 2*Td/Tp - 1);
r0 = K*(1 + Tp/(2*Ti) + Td/Tp);
```

Zadanie 3 – Podsumowanie

Wyznaczono wzmocnienie krytyczne oraz okres oscylacji zgodnie z metodą Zieglera–Nicholsa. Otrzymano parametry zarówno regulatora ciągłego, jak i jego wersji dyskretniej.

Zadanie 4

Napisano program do symulacji cyfrowego algorytmu PID oraz algorytmu DMC w wersji analitycznej, bez ograniczeń. Przyjęto stałą trajektorie referencyjną dla całego horyzontu predykcji. Model z punktu drugiego wykorzystano do wyznaczenia odpowiedzi skokowej i symulacji obiektu.

0.6. Symulacja cyfrowego algorytmu PID

Cyfrowy algorytm PID określony jest równaniem różnicowym

$$u(k) = r_2 e(k-2) + r_1 e(k-1) + r_0 e(k) + u(k-1) \quad (8)$$

Wyliczone wcześniej parametry r_2, r_1, r_0 wykorzystano do przeprowadzenia symulacji cyfrowego algorytmu PID za pomocą skryptu MATLAB:

```
sim_time = 200;
kk=sim_time/Tp;
y = zeros(1, kk);
u = zeros(1, kk);
yzad = zeros(1, kk);
e = zeros(1, kk);
yzad(10:floor(kk/3))=1;
yzad(floor(kk/3)+1:floor(2*kk/3))=-1;
yzad(floor(2*kk/3)+1:kk)=1;

for k=13:kk
    y(k)=b11*u(k-11)+b12*u(k-12)-a1*y(k-1)-a2*y(k-2);
    e(k)=yzad(k)-y(k);
    u(k)=r2*e(k-2)+r1*e(k-1)+r0*e(k)+u(k-1);
end
```

0.7. Symulacja algorytmu DMC

Do symulacji algorytmu DMC wykorzystano skrypt MATLAB:

```
sim_time = 200;
kk=sim_time/Tp;

s = step(Gz, D);
s = s(:)' ;

M = zeros(N, Nu);
for i = 1:N
    for j = 1:Nu
        if i >= j
            M(i, j) = s(i-j+1);
        end
    end
end
```

```

end

Mp = zeros(N, D-1);
for i = 1:N
    for j = 1:D-1
        if i+j <= D
            Mp(i, j) = s(i+j) - s(j);
        else
            Mp(i, j) = s(D) - s(j);
        end
    end
end

K = inv(M'*M + lambda*eye(Nu)) * M';
K1 = K(1, :);

u_dmc = zeros(1, kk);
y_dmc = zeros(1, kk);
dUp = zeros(D-1, 1);
yzad = zeros(1, kk);
yzad(20:floor(kk/3))=1;
yzad(floor(kk/3)+1:floor(2*kk/3))=-1;
yzad(floor(2*kk/3)+1:kk)=1;

for k = 13:kk
    y_dmc(k) = b11*u_dmc(k-11) + b12*u_dmc(k-12) - ...
               a1*y_dmc(k-1) - a2*y_dmc(k-2);
    y0 = y_dmc(k) + Mp * dUp;
    du = K1 * (yzad(k)*ones(N,1) - y0);
    u_dmc(k) = u_dmc(k-1) + du;
    dUp = [du; dUp(1:end-1)];
end

```

Gdzie parametry D , N , Nu , λ dobrano w Zadaniu 5.

Macierz M^p służy do wyznaczania trajektorii swobodnej procesu w każdej dyskretnej chwili k , zaś macierze M oraz K służą do wyliczania sygnału sterowania.

Działanie obu regulatorów przedstawiono oraz porównano po dostrojeniu regulatora DMC, w zadaniu 6.

Zadanie 4 – Podsumowanie

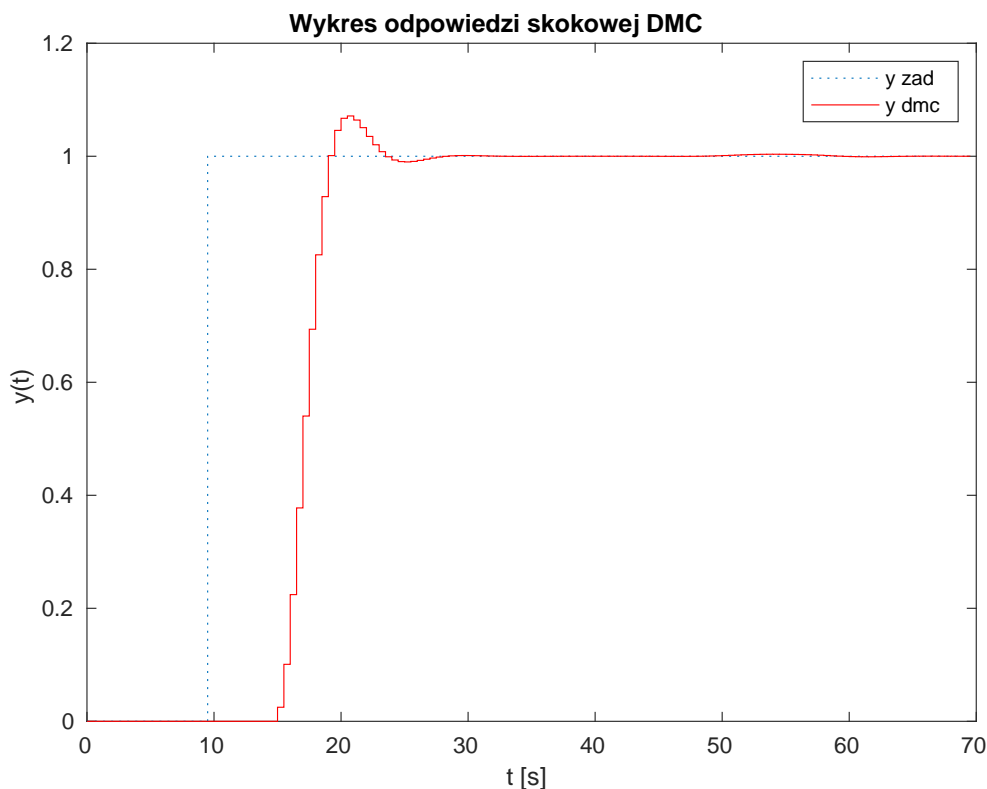
Zaimplementowano od podstaw cyfrowe wersje algorytmów PID i DMC w wersjach analitycznych, bez ograniczeń. Potwierdzono poprawność ich działania oraz możliwość dalszego doszkalania algorytmu DMC w kolejnych zadaniach.

Zadanie 5

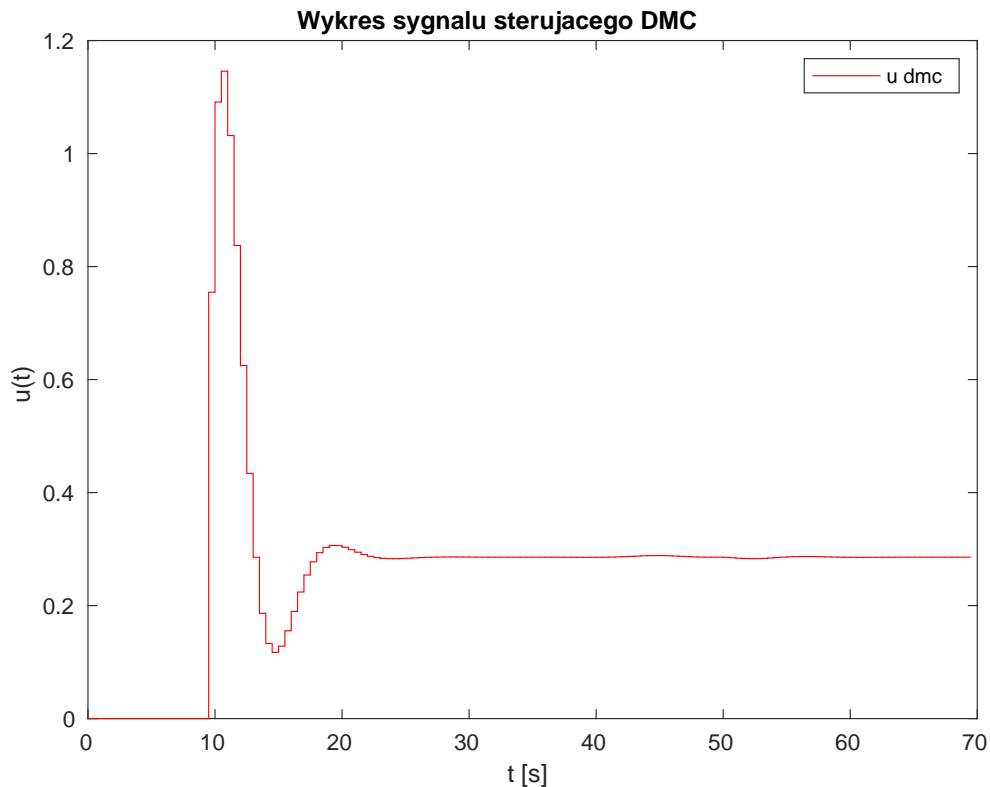
W zadaniu 5 dobrano parametry regulatora DMC aby zminimalizować przestrzał oraz znaleźć kompromis między szybkością w osiągnięciu wartości zadanej a jakością sygnału sterującego.

a) Początkowe parametry DMC

Na podstawie wykresu (1) odpowiedzi skokowej wyznaczonej w zadaniu 1 wyznaczono horyzont dynamiki D na $\frac{40s}{T_p} = 80$. Założono początkowe wartości długości horyzontów predykcji i sterowania takie same, jak horyzontu dynamiki ($N_u = N = D = 80$). Przyjęto początkową wartość współczynnika $\lambda = 1$. Do symulacji wykorzystano kod przedstawiony w Zadaniu 4. Wyniki symulacji przedstawiono na wykresach 3 i 4.



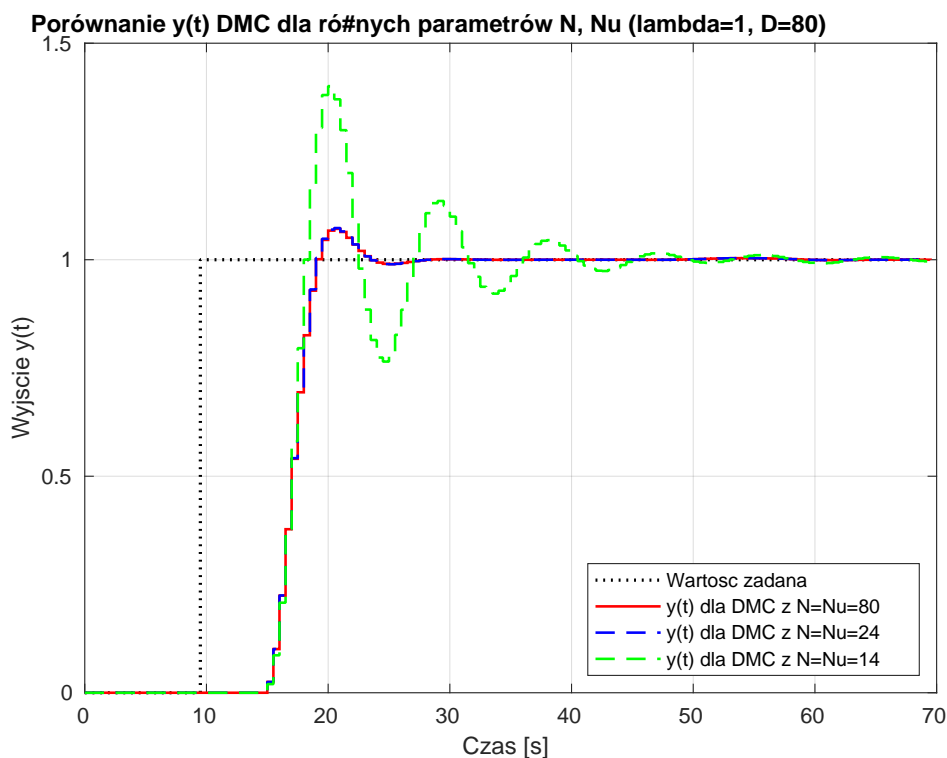
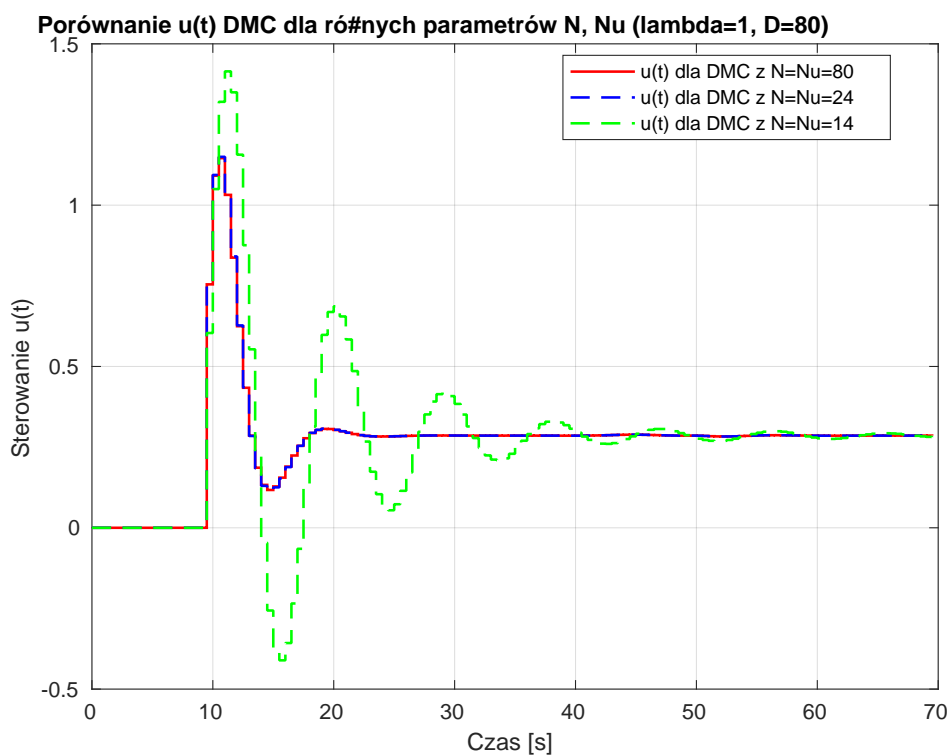
Rys. 3. Wykres $y(t)$ odpowiedzi skokowych przy regulacji DMC

Rys. 4. Wykres $u(t)$ odpowiedzi skokowych przy regulacji DMC

b) Skracanie horyzontu predykcji

Stopniowo skracano horyzont predykcji zachowując relację $N_u = N$. Zdecydowano się na $N_u = N = 24$ ponieważ dla mniejszych wartości pojawiało się coraz większe przestrzelenie, a od pewnego momentu duże oscylacje. Wyniki symulacji dla trzech różnych wartości $(N_u, N) \in \{(D, D), (24, 24), (14, 14)\}$ przedstawiono na wykresach 5 i 6.

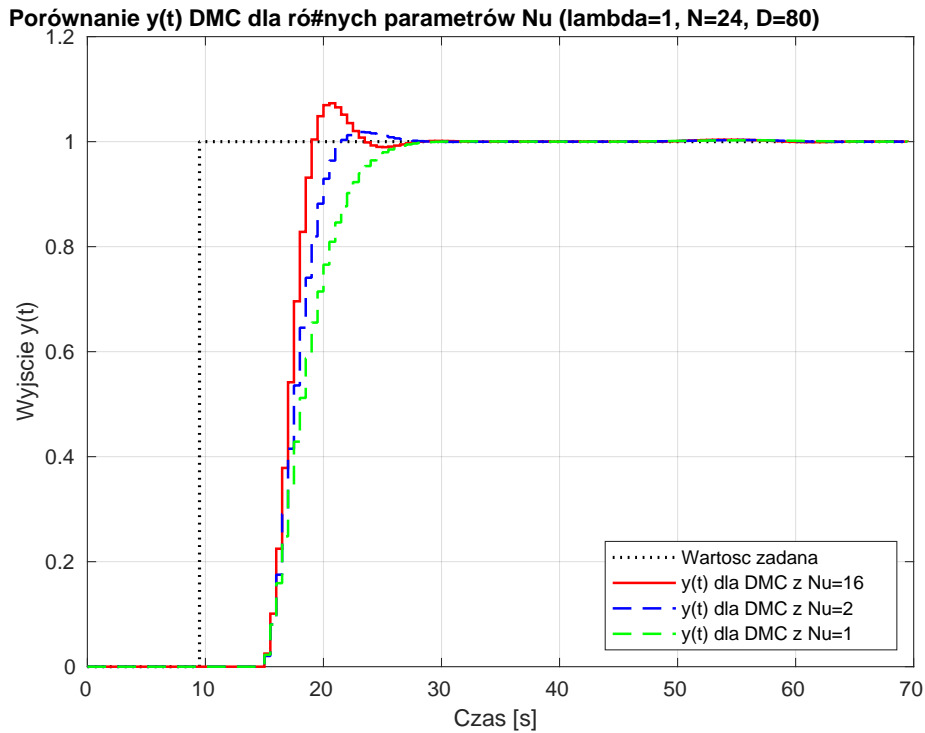
Wartości $N_u = N = 20$ skutkują podobnymi rezultatami co $N_u = N = D$, a wymagają znacznie mniejszego nakładu obliczeniowego. Wartości $N_u = N = 16$ skutkują znacznie większymi oscylacjami.

Rys. 5. Wykresy $y(t)$ odpowiedzi skokowych przy regulacji DMCRys. 6. Wykresy $u(t)$ odpowiedzi skokowych przy regulacji DMC

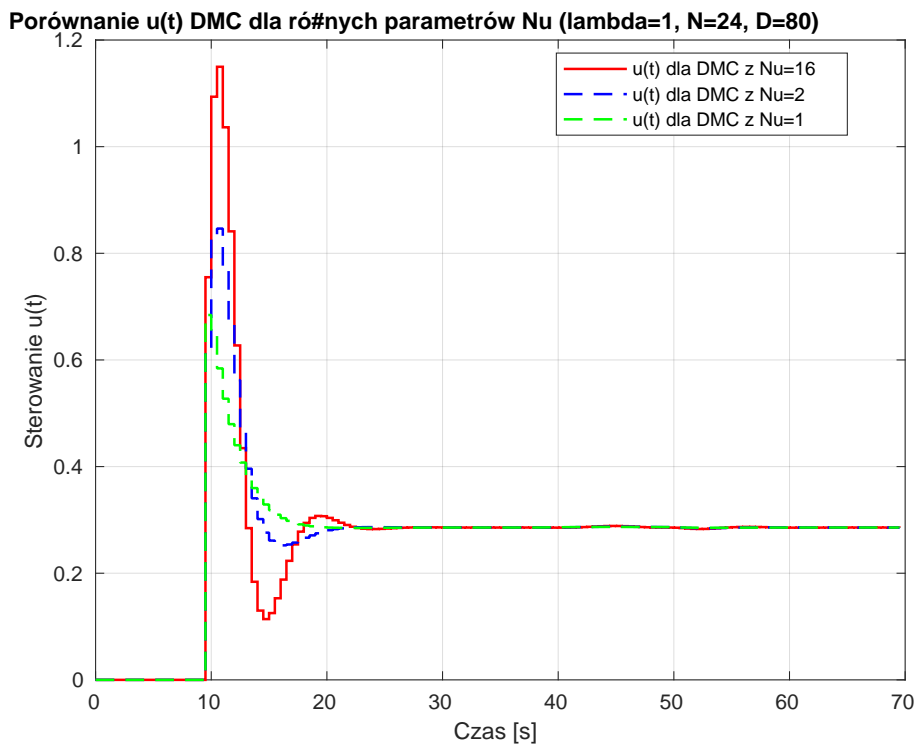
c) Skracanie horyzontu regulacji

Horyzont regulacji określa liczbę kroków w przyszłość na którą model przewiduje wartości \hat{y} . Przy ustalonych parametrach $(D, N, \lambda) = (80, 24, 1)$ testowano różne wartości $N_u \in \{1, 2, \dots, N\}$.

Na wykresach 7 oraz 8 przedstawiono wyniki dla $N_u \in \{16, 2, 1\}$. Zdecydowano się na wybór $N_u = 2$ ze względu na szybkość w osiągnięciu wartości zadanej wyjścia oraz niewielkim prze-strzeleniu które można zmniejszyć zwiększając wartość parametru λ .



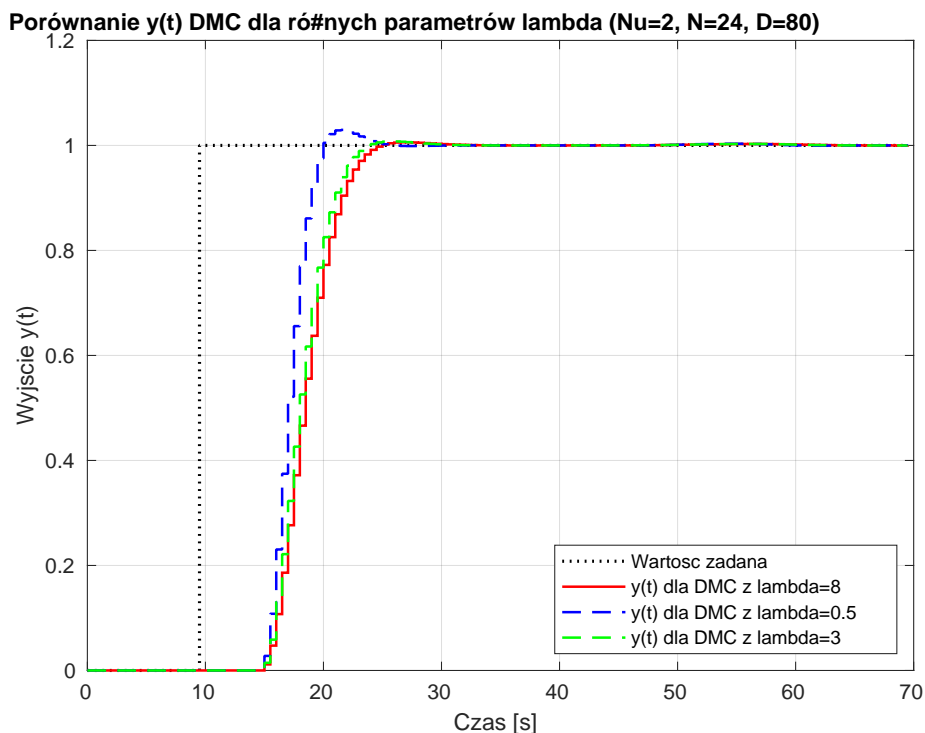
Rys. 7. Wykresy $y(t)$ odpowiedzi skokowych przy regulacji DMC



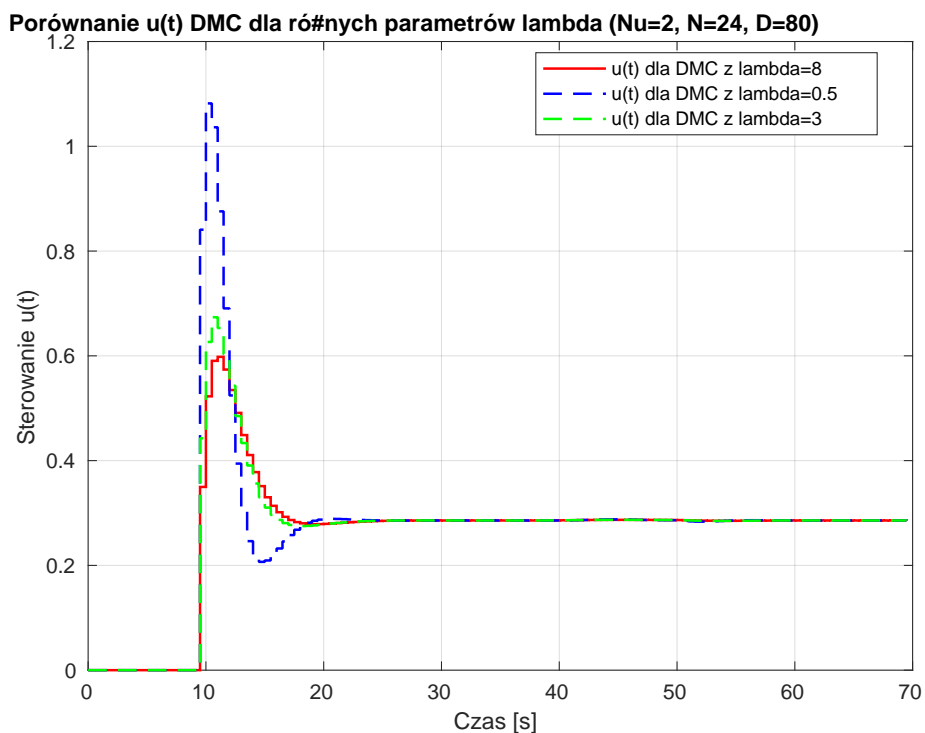
Rys. 8. Wykresy $u(t)$ odpowiedzi skokowych przy regulacji DMC

d) Manipulacja wartością parametru λ

Parametr λ określa „karę” otrzymywaną przez model przy gwałtownych zmianach sterowania. jego zwiększenie może stłumić oscylacje, zapobiec przestrzeleniu oraz zwiększyć jakość sygnału sterującego kosztem ewentualnej szybkości w osiągnięciu wartości zadanej. Przy ustalonych parametrach $(D, N, N_u) = (80, 24, 2)$ Testowano różne wartości parametru $\lambda \in [0,5; 10]$.



Rys. 9. Wykresy $y(t)$ odpowiedzi skokowych przy regulacji DMC



Rys. 10. Wykresy $u(t)$ odpowiedzi skokowych przy regulacji DMC

Na wykresach 9 oraz 10 przedstawiono wyniki dla $\lambda \in \{0,5; 3; 8\}$. Decydując się na kompromis między jakością sygnału sterującego oraz szybkością osiągnięcia wartości zadanej, wybrano wartość parametry $\lambda = 3$, uzyskując tym samym ostatecznie skalibrowany regulator DMC z parametrami $D = 80, N = 24, N_u = 2, \lambda = 3$.

Zadanie 5 – Podsumowanie

Krok po kroku dostrojono parametry regulatora DMC, analizując wpływ długości horyzontów oraz współczynnika kary sterowania. Wybrano finalne wartości $D=80, N=24, N_u=2$, , zapewniające dobre właściwości dynamiczne i jakość sterowania.

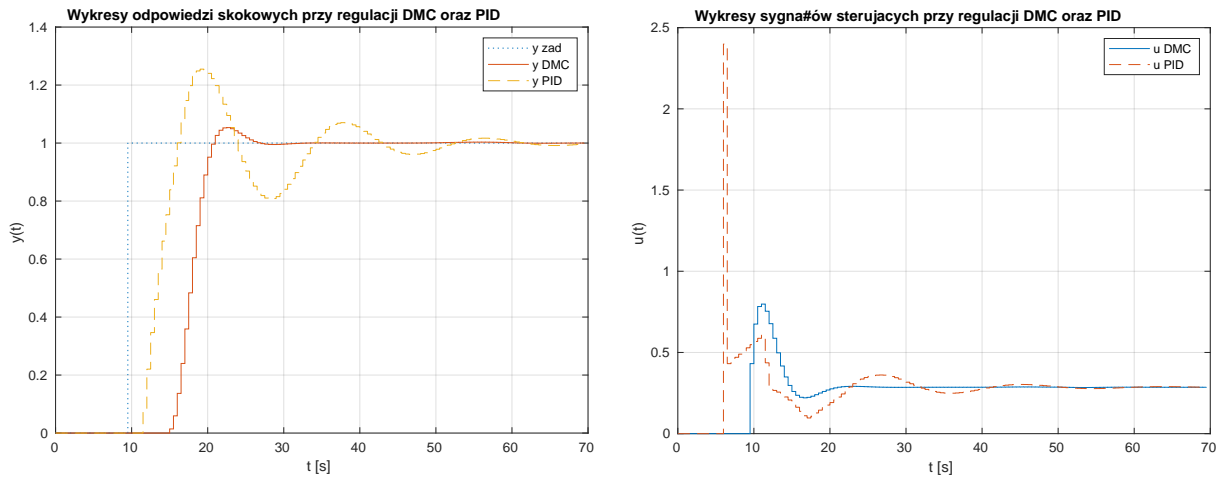
Zadanie 6

0.8. Porównanie regulatorów DMC oraz cyfrowego PID

Porównano odpowiedzi skokowe oraz przebiegi sygnałów sterowania regulatorów DMC oraz cyfrowego PID których parametry dobrano w poprzednich zadaniach:

$$\begin{aligned} T_p &= 0,5 \\ r_2 &= 1,9862 \quad r_1 = -4,0643 \quad r_0 = 2,6516 \\ D &= 80 \quad N = 24 \quad N_u = 2 \quad \lambda = 3 \end{aligned} \quad (9)$$

Przebiegi $y(t)$ oraz $u(t)$ przedstawiono na wykresach 11



Rys. 11. Wykresy $u(t)$ oraz $y(t)$ - Regulatory DMC oraz PID

Algorytm DMC sprawuje się dużo lepiej od cyfrowego algorytmu PID zarówno pod względem przebiegu odpowiedzi skokowej (osiągnięcie stabilnej wartości zadanej po zaledwie około $26s$ w przeciwieństwie do niemal $70s$ w przypadku cyfrowego algorytmu PID, brak oscylacji i niewielkie przestrzelenie w przeciwieństwie do algorytmu PID), jak i pod względem jakości przebiegu sygnału sterowania (Dużo mniejsze wartości i mniej gwałtowne zmiany sterowania).

0.9. Krzywe stabilności regulatora DMC oraz cyfrowego PID

W celu wyznaczenia aproksymacji krzywej stabilności algorytmów DMC oraz cyfrowego PID, zmieniano wartości opóźnienia modelu oraz wzmocnienia w transmitancji modelu. Po wyliczeniu parametrów r_2, r_1, r_0 regulatora PID oraz macierzy regulatora DMC z wykorzystaniem oryginalnej transmitancji 1 w sposób przedstawiony w poprzednich zadaniach zamieniono transmitancję obiektu na

$$G(s) = \frac{K_{\text{test}} e^{-T_{\text{test}}}}{(T_1 s + 1)(T_2 s + 1)} \quad (10)$$

gdzie $T_{\text{test}} \in \{\frac{10}{2}, \frac{11}{2}, \frac{12}{2}, \frac{13}{2}, \frac{14}{2}, \frac{15}{2}, \frac{16}{2}, \frac{17}{2}, \frac{18}{2}, \frac{19}{2}, \frac{20}{2}\}$. Dla każdego z algorytmów i dla każdej z tych wartości zwiększano K_{test} od $K_{\text{test},0} = 3,5$ z krokiem $\Delta K = 0,01$ aż do uzyskania niestabilnego przebiegu odpowiedzi skokowej przy regulacji oryginalnymi regulatorami. W tym celu

użyto skryptu MATLAB który automatycznie oceniał stabilność przebiegu prostą, empirycznie przetestowaną funkcją:

```
function stable = check_stability(y, yzad)
    y = y(0.7* length(y):end);
    tol = 0.1;
    stable = true;
    for i = 2:length(y)
        cond1 = abs(y(i) - yzad) > abs(y(i-1) - yzad)+tol;
        cond2 = abs(y(i) - yzad) > tol ;
        if cond1 && cond2
            stable = false;
        end
    end
end
```

Funkcja przyjmuje na wejściu przebieg $y(t)$ oraz wartość zadaną y_{zad} i sprawdza czy dla ostatnich 30% przebiegu spełniony jest chociaż jeden z warunków:

$$\begin{aligned} \forall_{i \in \{2,3,\dots,\text{len}(y)\}} |y(i) - y_{zad}| &\leq 0,1 \\ \forall_{i \in \{2,3,\dots,\text{len}(y)\}} |y(i) - y_{zad}| - |y(i-1) - y_{zad}| &\leq 0,1 \end{aligned} \quad (11)$$

Sprawdzone jest więc, czy odległość moduł sygnału $|y|$ od wartości zadanej y_{zad} od pewnego momentu maleje (lub tylko bardzo powoli rośnie) lub czy od pewnego momentu wartości sygnału y są odpowiednio blisko wartości zadanej. Do przetestowania wszystkich wartości T_{test} wykorzystano skrypt MATLAB:

```
%% Obszary stabilności algorytmów DMC i PID
K_nom = 3.5;
sim_time = 1000;
kk=sim_time/Tp;
delay_values = 10:1:20;
max_K = 20;
step_K = 0.01;
K_stability_PID = zeros(size(delay_values));
K_stability_DMC = zeros(size(delay_values));

for i = 1:length(delay_values)
    delay = delay_values(i);
    % Znajdowanie granic stabilności dla PID
    for K = K_nom:step_K:max_K
        % Obliczenie parametrów modelu dla zmodyfikowanych wartości
        G_s = K / ((T1*s+1)*(T2*s+1));
        Gz = c2d(G_s, Tp, 'zoh');
        [bt, at] = tfdata(Gz, 'v');
        b1 = b(2);
        b2 = b(3);
        a = [at(2) at(3)];
        b = zeros(1, delay+2);
        b(delay+1) = bt(2); b(delay+2) = bt(3);
        is_stable_PID = true;
        y_test = zeros(1, kk);
        u_test = zeros(1, kk);
        e_test = zeros(1, kk);
        yzad_test = zeros(1, kk);
        yzad_test(delay+3:kk) = 1;

        for k=delay+3:kk
```

```

        y_test(k) = 0;
        for j=1:delay+2
            y_test(k) = y_test(k) + b(j)*u_test(k-j);
        end
        for j=1:2
            y_test(k) = y_test(k) - a(j)*y_test(k-j);
        end
        e_test(k) = yzad_test(k) - y_test(k);
        u_test(k) = r2*e_test(k-2) + r1*e_test(k-1) + ...
            r0*e_test(k) + u_test(k-1);
    end
    if ~check_stability(y_test, yzad_test(kk))
        K_stability_PID(i) = K/K_nom;
        break;
    end
end

% Znajdowanie granic stabilności dla DMC
for K = K_nom:step_K:max_K
    % Obliczenie parametrów modelu dla zmodyfikowanych wartości
    G_s = K / ((T1*s+1)*(T2*s+1));
    Gz = c2d(G_s, Tp, 'zoh');
    [bt, at] = tfdata(Gz, 'v');
    b1 = b(2);
    b2 = b(3);
    a = [at(2) at(3)];
    b = zeros(1, delay+2);
    b(delay+1) = bt(2); b(delay+2) = bt(3);
    is_stable_DMC = true;

    y_test = zeros(1, kk);
    u_test = zeros(1, kk);
    dUp_test = zeros(D-1, 1);
    yzad_test = zeros(1, kk);
    yzad_test(delay+3:kk) = 1;

    for k = delay+3:kk
        y_test(k) = 0;
        for j=1:delay+2
            y_test(k) = y_test(k) + b(j)*u_test(k-j);
        end
        for j=1:2
            y_test(k) = y_test(k) - a(j)*y_test(k-j);
        end
        y0_test = y_test(k) + Mp_test * dUp_test;
        du_test = K1_test * (yzad_test(k)*ones(N,1) - y0_test);
        u_test(k) = u_test(k-1) + du_test;
        dUp_test = [du_test; dUp_test(1:end-1)];
    end
    if ~check_stability(y_test, yzad_test(kk))
        K_stability_DMC(i) = K/K_nom;
        break;
    end
end
end
end

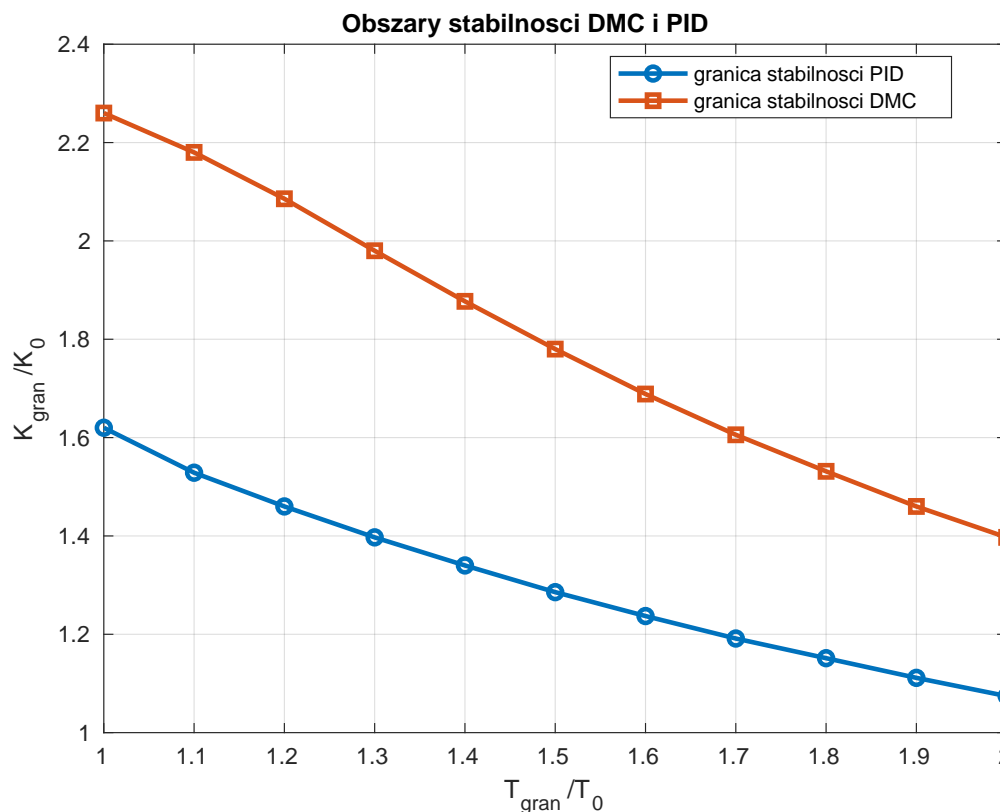
```

Natępnie wyliczone wartości z tablic `K_stability_PID(i)` oraz `K_stability_DMC(i)` zaokrąglono oraz zweryfikowano ręcznie obserwując wykresy odpowiedzi skokowych dla każdej z nich.

Algorytm wyliczył wartości graniczne prawie poprawnie, wymagały jedynie drobnych korekt. Dla danego opóźnienia T_{gran} dobierano takie K_{gran} , aby otrzymać stabilne oscylacje o stałej amplitudzie. Ostateczne wyznaczone wartości przedstawiono na wykresie 12 oraz w tabeli 0.9

$T_{\text{gran}} : T_0$	$K_{\text{gran,PID}} : K_0$	$K_{\text{gran,DMC}} : K_0$
1,0	1,61	2,25
1,1	1,53	2,18
1,2	1,46	2,09
1,3	1,40	1,98
1,4	1,34	1,88
1,5	1,29	1,78
1,6	1,23	1,69
1,7	1,19	1,61
1,8	1,15	1,53
1,9	1,11	1,46
2,0	1,07	1,40

Tab. 1. Wyniki analizy stabilności dla różnych wartości T_{gran}



Rys. 12. Aproksymacje krzywych stabilności PID oraz DMC

Algorytm DMC oferuje stabilność na znacznie większym obszarze niż cyfrowy PID. Stabilność obu algorytmów maleje wraz ze wzrostem współczynnika K_0 oraz ze wzrostem opóźnienia T_0 co jest zgodne z oczekiwaniami.

Zadanie 6 – Podsumowanie

Przeprowadzono porównanie PID i DMC dla tej samej wartości zadanej. Regulator DMC wykazał znacznie lepszą jakość regulacji, krótszy czas ustalania oraz mniejsze oscylacje. Dodatkowo wyznaczono obszary stabilności obu regulatorów, pokazując większą odporność DMC.

Zadanie 7

Na podstawie modelu 5 zaimplementowano algorytm GPC.

0.10. Deklaracja parametrów modelu oraz algorytmu

Parametry modelu oraz algorytmu GPC zadeklarowano następująco:

```
zadanie_2
na = 2;
nb = 12;
D = 40/Tp; % 80
N = 12/Tp; % 24
Nu = 1/Tp; % 2
lambda = 3;
Tp = 0.5;
sim_time = 60;
kk = sim_time/Tp;
a = [a1 a2];
b = [0 0 0 0 0 0 0 0 0 0 0 b11 b12];
```

W skrypcie `zadanie_2` znajduje się fragment kodu przedstawiony w sprawozdaniu z Zadania 2. Parametry D, N, N_u dobrano te same co uzyskane w Zadaniu 5.

0.11. Wektor s , macierze M oraz K

Do wyznaczenia wektora s oraz macierzy M, K potrzebnych do działania algorytmu GPC wykorzystano skrypt MATLAB:

```
s = zeros(1, N);
for j = 1:N
    for i = 1:min(j, nb)
        s(j) = s(j) + b(i);
    end
    for i = 1:min(j-1, na)
        s(j) = s(j) - a(i)*s(j-i);
    end
end

M = zeros(N, Nu);
for i = 1:N
    for j = 1:Nu
        if i >= j
            M(i, j) = s(i-j+1);
        end
    end
end

K = inv(M'*M + lambda*eye(Nu)) * M';
K1 = K(1, :);
```

0.12. Przebieg działania algorytmu

Przebieg algorytmu realizuje skrypt MATLAB:

```
u = zeros(1, kk);
yzad = zeros(1, kk);
yzad(20:kk) = 1;
y = zeros(1, kk);
interference = zeros(1, kk);
interference(20:kk) = 0.1;

for k = 13:kk
    y(k) = b11*u(k-11) + b12*u(k-12) - a1*y(k-1) - a2*y(k-2);
    % jeśli uwzględniamy zakłócenie:
    % y(k) = y(k)+interference(k);
    d = y(k);
    for i = 1:nb
        d = d - b(i)*u(k-i);
    end
    for i = 1:na
        d = d + a(i)*y(k-i);
    end
    y0 = zeros(1, N);
    for p = 1:N
        for i = 1:min(p, nb)
            y0(p) = y0(p) + b(i)*u(k-1);
        end
        for i = min(p, nb)+1:nb
            y0(p) = y0(p) + b(i)*u(k-i+p);
        end
        for i = 1:min(p-1, na)
            y0(p) = y0(p) - a(i)*y0(p-i);
        end
        for i = min(p-1, na)+1:na
            y0(p) = y0(p) - a(i)*y(k-i+p);
        end
        y0(p) = y0(p) + d;
    end

    du = K1 * (yzad(k)*ones(N,1) - y0');
    u(k) = u(k-1) + du;
end
```

Macierze M oraz K , tak samo jak w przypadku algorytmu DMC, służą do wyliczania wartości zmiany sygnału sterowania w chwili bieżącej na podstawie trajektorii swobodnej oraz wartości zadanej. Trajektoria swobodna wyliczana jest jednak w sposób iteracyjny z uwzględnieniem czynnika $d(k) = y(k) - y(k|k-1)$. Pozwala to na lepszą predykcję trajektorii swobodnej procesu. Działanie algorytmu testowano w następujących zadaniach.

Zadanie 7 – Podsumowanie

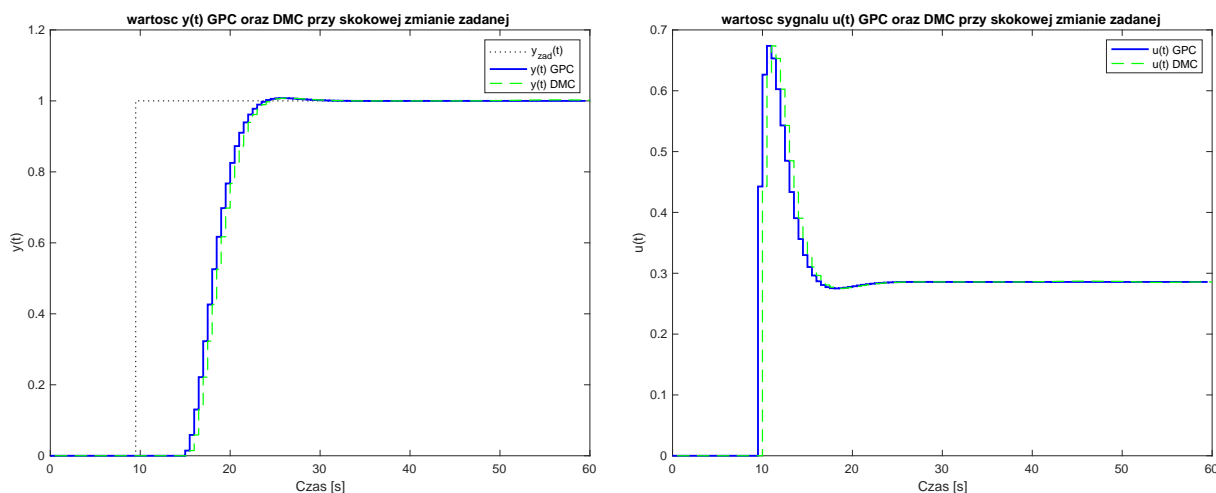
Na podstawie wcześniej wyznaczonego modelu opracowano i zaimplementowano algorytm GPC. Jego struktura pozwala na bardziej zaawansowaną predykcję niż w przypadku DMC, uwzględniając dokładniej dynamikę i zakłócenia.

Zadanie 8

Porównano algorytmy GPC oraz DMC z tymi samymi parametrami pod względem szybkości odpowiedzi na zmianę zakłócenia, szybkości w osiągnięciu wartości zadanej oraz jakości przebiegu sygnału sterującego.

0.13. a) Porównanie algorytmów GPC oraz DMC przy skokowej zmianie wartości zadanej

Przyjęto zerowe warunki początkowe oraz jednostkowy skok wartości zadanej w chwili $t = 10$. Odpowiedzi obu algorytmów oraz przebiegi sygnałów sterowania zamieszczono na wykresach 13

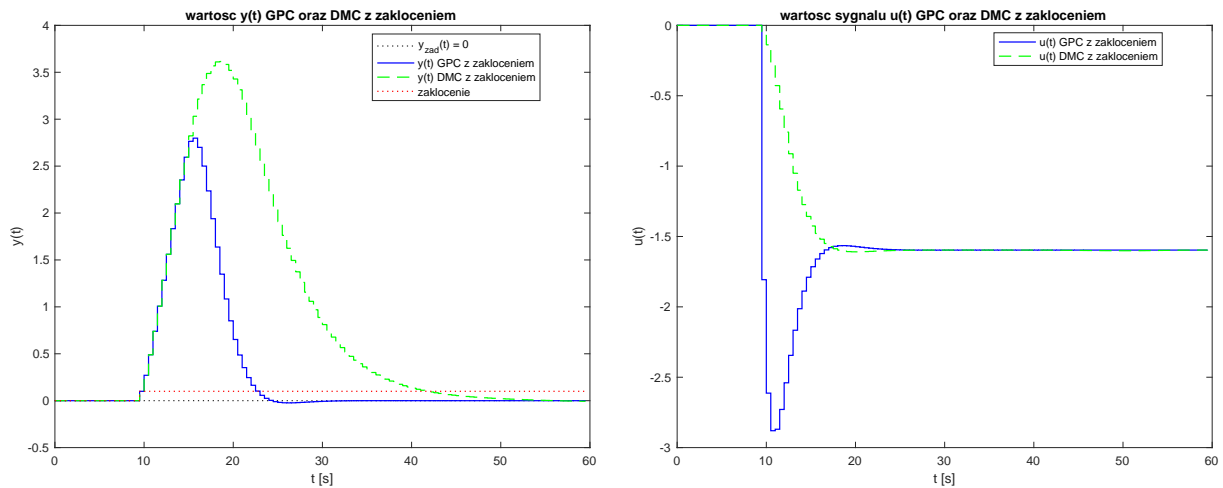


Rys. 13. Wykresy $u(t)$ oraz $y(t)$ - Regulatory DMC oraz GPC. skokowa wartość y_{zad}

Regulatory DMC oraz GPC zachowały się niemal identycznie. Oba szybko (po około 25 sekundach) osiągnęły stabilną wartość zadaną. Uwzględniając opóźnienie 5s potrzebowały około 20 sekund.

0.14. b) Porównanie algorytmów GPC oraz DMC przy skokowej zmianie wartości zakłócenia

Przyjęto zerowe warunki początkowe oraz skok zakłócenia do wartości 0,1 w chwili $t = 10$. Odpowiedzi obu algorytmów oraz przebiegi sygnałów sterowania zamieszczono na wykresach 14



Rys. 14. Wykresy $u(t)$ oraz $y(t)$ - Regulatory DMC oraz GPC. skokowa wartość zakłócenia

W przypadku regulacji ze skokiem zakłócenia, regulator GPC znacznie szybciej sprowadził wyjście do wartości zadanej po danym skoku. Kosztem tego szybkiego sprowadzenia jest gwałtowny skok sygnału sterowania. Szybkość algorytmu GPC wynika z uwzględnieniu zakłócenia w chwili obecnej do wyznaczenia przewidywanych wartości wyjścia w chwilach następnych w przeciwieństwie do DMC który tego nie robi.

Zadanie 8 – Podsumowanie

Regulatory GPC i DMC zostały porównane zarówno przy zmianie wartości zadanej, jak i w obecności zakłócenia. GPC lepiej radził sobie z zakłóceniami, szybciej sprowadzając wyjście do wartości zadanej kosztem większego działania sterowania.

zadanie 9

0.15. Krzywa stabilności algorytmu GPC

Wyznaczono krzywą stabilności regulatora GPC z parametrami takimi jak w poprzednich zadaniach. Zastosowano tę samą metodę co w zadaniu 6.7.2. Zastosowano też tę samą funkcję określającą stabilność przebiegu. Do znalezienia przybliżeń granic stabilności dla $T_{\text{test}} \in \{\frac{10}{2}, \frac{11}{2}, \frac{12}{2}, \frac{13}{2}, \frac{14}{2}, \frac{15}{2}, \frac{16}{2}, \frac{17}{2}, \frac{18}{2}, \frac{19}{2}, \frac{20}{2}\}$ wykorzystano skrypt MATLAB:

```
delay_values = 10:1:20; % Wartości opóźnienia do testowania
K_nom = 3.5; % Nominalna wartość K
max_K = 200; % Maksymalna wartość K
step_K = 0.01; % Krok zmiany K
K_stability_GPC = zeros(1, length(delay_values));

for i = 1:length(delay_values)
    delay = delay_values(i);
    is_stable_GPC = true;

    % Znajdowanie granic stabilności dla GCP
    for K = K_nom:step_K:max_K
        % Obliczenie parametrów modelu dla zmodyfikowanych wartości
        G_s = K / ((T1*s+1)*(T2*s+1));
        Gz = c2d(G_s, Tp, 'zoh');
        [b, a] = tfdata(Gz, 'v');
        b1 = b(2);
        b2 = b(3);
        a1 = a(2);
        a2 = a(3);
        a = [a1 a2];
        b = zeros(1, delay+2);
        na = length(a);
        nb = length(b);
        b(delay+1) = b1;
        b(delay+2) = b2;
        u = zeros(1, kk);
        yzad = zeros(1, kk);
        yzad(delay+3:kk) = 1;
        y = zeros(1, kk);

        for k = delay+3:kk
            y(k) = 0;
            for j = 1:delay+2
                y(k) = y(k) + b(j)*u(k-j);
            end
            for j = 1:2
                y(k) = y(k) - a(j)*y(k-j);
            end
            y0 = zeros(1, N);
            for p = 1:N
```

```

        for j = 1:min(p, nb_original)
            y0(p) = y0(p) + b_original(j)*u(k-1);
        end
        for j = min(p, nb_original)+1:nb_original
            y0(p) = y0(p) + b_original(j)*u(k-j+p);
        end
        for j = 1:min(p-1, na_original)
            y0(p) = y0(p) - a_original(j)*y0(p-j);
        end
        for j = min(p-1, na_original)+1:na_original
            y0(p) = y0(p) - a_original(j)*y(k-j+p);
        end
        y0(p) = y0(p);
    end

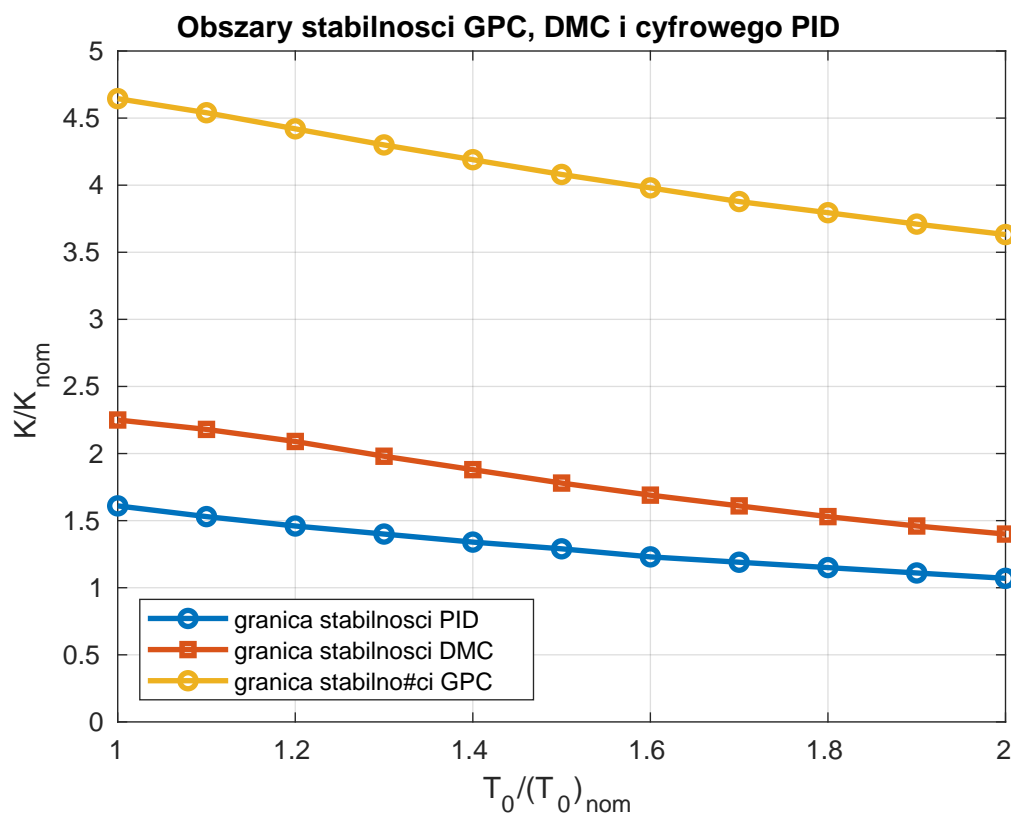
    du = K1 * (yzad(k)*ones(N,1) - y0');
    u(k) = u(k-1) + du;
end
if ~check_stability(y, yzad(kk))
    is_stable_GPC = false;
    disp('unstable')
    disp(K)
else
    is_stable_GPC = true;
end
if ~is_stable_GPC
    K_stability_GPC(i) = K/K_nom;
    break;
end
if K == max_K
    K_stability_GPC(i) = K/K_nom;
end
end
end
end

```

Uzyskane w ten sposób wartości K granicznych manualnie dostrojono badając wykres odpowiedzi na skok wartości zadanej. Ostatecznie wyznaczone wartości przedstawiono w tabeli 0.15. Na wykresie 15 zestawiono ze sobą krzywe stabilności wszystkich trzech regulatorów.

$T_{\text{gran}} : T_0$	$K_{\text{gran,GPC}} : K_0$
1,0	4,645
1,1	4,540
1,2	4,420
1,3	4,300
1,4	4,190
1,5	4,080
1,6	3,980
1,7	3,878
1,8	3,795
1,9	3,710
2,0	3,632

Tab. 2. Wyniki analizy stabilności algorytmu GPC dla różnych wartości T_{gran}



Rys. 15. Aproxymacje krzywych stabilności GPC

Na podstawie przeprowadzonej analizy można zauważyć, że algorytm GPC wykazuje istotnie szerszy zakres stabilności w porównaniu z regulatorami PID i DMC. Dla rosnących wartości opóźnienia czasowego systemu (parametru T_{gran}), graniczna wartość wzmacnienia $K_{\text{gran, GPC}}$ malała w sposób płynny, ale pozostawała znacznie wyższa niż w przypadku pozostałych algorytmów. Świadczy to o wysokiej odporności GPC na zmiany dynamiki procesu oraz o jego dużej elastyczności w doborze parametrów. Taka właściwość czyni go szczególnie użytecznym w zastosowaniach, gdzie dokładna znajomość modelu jest ograniczona, a system może podlegać istotnym zmianom w czasie.

Zadanie 9 – Podsumowanie

Wyznaczono obszar stabilności dla algorytmu GPC. W porównaniu do DMC i PID, GPC wykazuje znacznie szerszy obszar stabilności, co potwierdza jego przydatność w systemach o zmiennych parametrach i ograniczonej znajomości modelu.

Wnioski końcowe

Zrealizowany projekt umożliwił dokładne porównanie klasycznego cyfrowego algorytmu PID z nowoczesnymi algorytmami predykcyjnymi DMC i GPC. Na podstawie przeprowadzonych eksperymentów można sformułować następujące wnioski:

- Algorytm DMC, po odpowiednim dostrojeniu parametrów ($D=80$, $N=24$, $Nu=2$, $\lambda = 3$), zapewniał znacznie lepszą jakość regulacji niż klasyczny cyfrowy PID – zarówno pod względem szybkości odpowiedzi, jak i charakterystyki sygnału sterującego.
- Algorytm GPC, implementowany na podstawie tego samego modelu i z identycznymi parametrami jak DMC, wykazał porównywalną skuteczność regulacyjną przy skoku wartości zadanej. Przy zmianie zakłócenia GPC reagował szybciej, jednak kosztem gwałtowniejszych zmian sterowania.
- Analiza obszarów stabilności wykazała, że GPC cechuje się największą odpornością na zmiany parametrów modelu, co czyni go najbardziej uniwersalnym spośród testowanych rozwiązań. DMC również oferuje szeroki zakres stabilności, natomiast PID wykazuje dużą wrażliwość na wzrost opóźnienia i wzmocnienia modelu.
- Proces strojenia regulatora DMC pozwolił zaobserwować wpływ horyzontu predykcji, regulacji i współczynnika λ na jakość regulacji. Skracanie N i Nu zmniejszało zapotrzebowanie obliczeniowe, jednak zbyt krótkie wartości prowadziły do przestrzelin i oscylacji.

Podsumowując, algorytmy predykcyjne, a zwłaszcza GPC, są znacznie bardziej elastyczne i odporne na zmiany parametrów procesu niż klasyczne rozwiązania PID. Ich stosowanie, choć bardziej wymagające obliczeniowo, jest uzasadnione w przypadku procesów o dużym opóźnieniu i zmiennej dynamice jeśli dostępny jest model procesu.