Wydział Elektroniki i Technik Informacyjnych
Politechnika Warszawska

Przedmiot: Sterowanie procesami

# Sprawozdanie z laboratorium
# Sterowanie Procesami
# Temat: Regulatory predykcyjne

Autor: Jakub Szubzda

Warszawa, 1 czerwca 2025

# Spis treści

# 1. Wstęp

Celem projektu laboratoryjnego było zapoznanie się z regulatorami predykcyjnymi DMC (Dynamic Matrix Control) i GPC (Generalized Predictive Control) oraz porównanie ich działania z klasycznym regulatorem PID. Wykonano analizę obiektu inercyjnego drugiego rzędu z opóźnieniem, zaprojektowano dla niego regulatory predykcyjne oraz przeprowadzono symulacje w środowisku MATLAB/Simulink.

W ramach projektu zrealizowano następujące zadania:

1. Analiza transmitancji obiektu oraz jej reprezentacja w postaci dyskretnej
2. Wyprowadzenie równania różnicowego na podstawie transmitancji dyskretnej
3. Strojenie regulatora PID metodą Zieglera-Nicholsa
4. Synteza i badanie regulatora DMC (wpływ horyzontu predykcji, horyzontu sterowania oraz współczynnika kary za sterowanie)
5. Porównanie regulatorów DMC i PID
6. Porównanie regulatorów DMC i GPC
7. Badanie obszarów stabilności regulatorów PID, DMC i GPC

Projekt pozwolił na praktyczne poznanie zalet i ograniczeń regulatorów predykcyjnych oraz nabycie umiejętności doboru ich parametrów.

# 2. Zadania projektowe

## 2.1. Analiza transmitancji obiektu

Obiektem badań był układ inercyjny drugiego rzędu z opóźnieniem, opisany transmitancją ciągłą:

$$G(s) = \frac{K_o e^{-T_o s}}{(T_1 s + 1)(T_2 s + 1)} \tag{2.1}$$

gdzie:

— $K_o = 4,7$ - współczynnik wzmocnienia statycznego
— $T_o = 5$ s - opóźnienie transportowe
— $T_1 = 1,92$ s - pierwsza stała czasowa
— $T_2 = 4,96$ s - druga stała czasowa

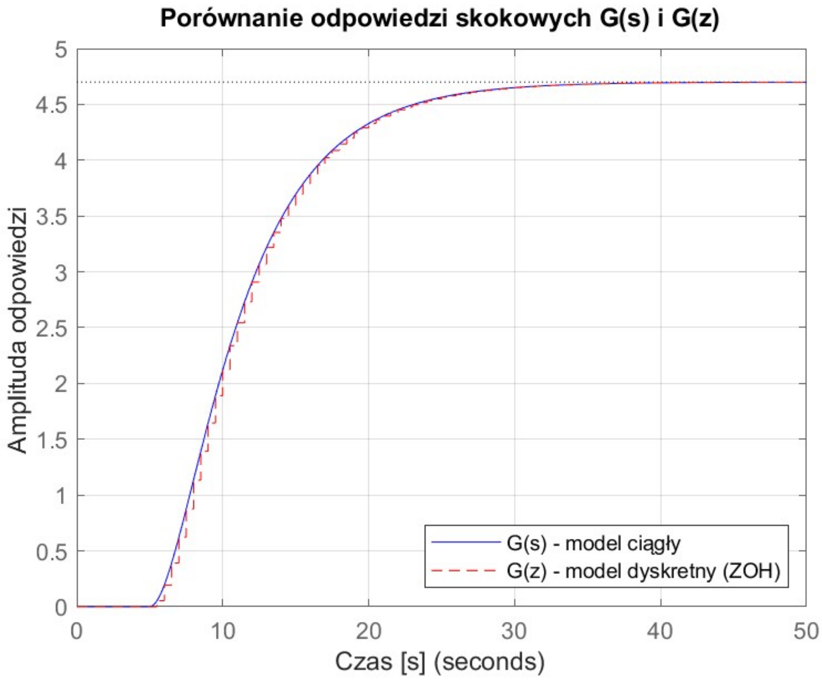
Po podstawieniu wartości parametrów uzyskano transmitancję ciągłą w postaci:

$$G(s) = e^{-5s} \cdot \frac{4,7}{9,523s^2 + 6,88s + 1} \tag{2.2}$$

Transmitancja dyskretna została wyznaczona za pomocą funkcji `c2d(G,Tp,'zoh')` w środowisku MATLAB, z okresem próbkowania $T_p = 0,5$ s. Uzyskano następującą postać:

$$G(z) = z^{-10} \cdot \frac{0,05477z + 0,04856}{z^2 - 1,675z + 0,6968} \tag{2.3}$$

Współczynniki wzmocnienia statycznego dla obu transmitancji są równe $K_s = K_z = 4,7$, co potwierdza poprawność przekształcenia z postaci ciągłej na dyskretną.




Rys. 2.1. Porównanie odpowiedzi skokowych obiektu ciągłego $G(s)$ i dyskretnego $G(z)$ dla okresu próbkowania $T_p = 0,5$ s

Na rysunku 2.1 widać, że odpowiedzi skokowe modelu ciągłego i dyskretnego są zbliżone, co potwierdza poprawność dyskretyzacji.

## 2.2. Wyprowadzenie równania różnicowego

Na podstawie transmitancji dyskretnej $G(z)$ wyprowadzono równanie różnicowe opisujące obiekt. Transmitancja dyskretna ma postać:

$$G(z) = z^{-10} \cdot \frac{0,05477z + 0,04856}{z^2 - 1,675z + 0,6968} \tag{2.4}$$

Równanie różnicowe wyprowadzone z tej transmitancji:

$$y(k) = 1,6748 \cdot y(k-1) - 0,69682 \cdot y(k-2) + 0,054771 \cdot u(k-11) + 0,048559 \cdot u(k-12) \tag{2.5}$$

Ta postać pozwala na rekurencyjne obliczanie wyjścia procesu na podstawie poprzednich wartości wyjścia i sterowania. Opóźnienie transportowe jest reprezentowane przez przesunięcie sygnału wejściowego o 10 okresów próbkowania ($u(k-10)$ i później).

## 2.3. Strojenie regulatora PID metodą Zieglera-Nicholsa

Regulator PID został dostrojony metodą Zieglera-Nicholsa z wykorzystaniem parametrów krytycznych. Na podstawie analizy charakterystyki częstotliwościowej obiektu wyznaczono następujące parametry krytyczne:

— Wzmocnienie krytyczne $K_k = 0,46514$
— Okres oscylacji krytycznych $T_k = 10,6074$ s
— Pulsacja krytyczna $\omega_k = 0,59234$ rad/s

Zgodnie z zasadami Zieglera-Nicholsa, obliczono parametry ciągłego regulatora PID:

— $K_r = 0,19536$ (współczynnik wzmocnienia)
— $T_i = 7,5767$ s (stała czasowa całkowania)
— $T_d = 0,89102$ s (stała czasowa różniczkowania)

Co odpowiada następującym współczynnikom $K_p$, $K_i$ i $K_d$:

— $K_p = 0,19536$ (wzmocnienie członu proporcjonalnego)
— $K_i = 0,025784$ (wzmocnienie członu całkującego)
— $K_d = 0,17407$ (wzmocnienie członu różniczkującego)

Na podstawie tych wartości wyznaczono parametry dyskretnego regulatora PID w postaci równania różnicowego:

$$u(k) = u(k-1) + r_0 e(k) + r_1 e(k-1) + r_2 e(k-2) \tag{2.6}$$

gdzie:

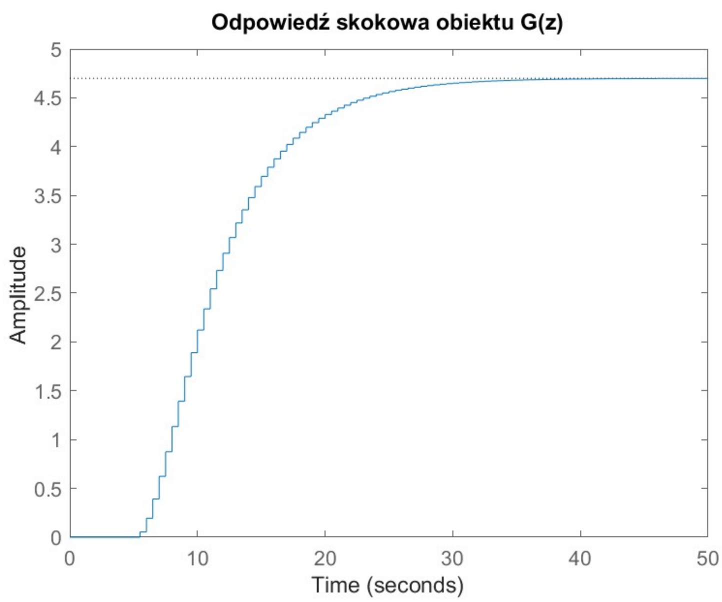
— $r_0 = 0,55639$
— $r_1 = -0,89163$
— $r_2 = 0,34814$

System wyświetlił ostrzeżenie, że układ zamknięty z otrzymanymi parametrami może być niestabilny, co wskazuje na trudności w regulacji tego obiektu przy użyciu standardowego regulatora PID. Jest to typowe dla obiektów z dużym opóźnieniem transportowym.

## 2.4. Synteza i badanie regulatora DMC

### 2.4.1. Wyznaczenie horyzontu dynamiki

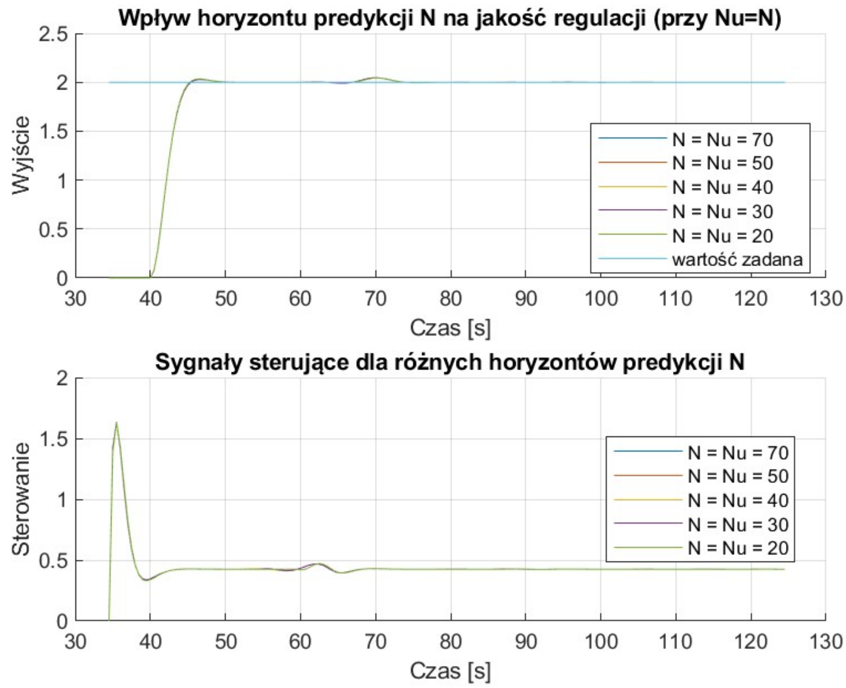
Pierwszym krokiem w syntezie regulatora DMC było wyznaczenie horyzontu dynamiki $D$, czyli liczby współczynników odpowiedzi skokowej koniecznych do poprawnego zamodelowania dynamiki obiektu. Na podstawie odpowiedzi skokowej obiektu (rysunek 2.2) przyjęto $D = 60$, co odpowiada czasowi 30 sekund (przy $T_p = 0,5$ s), po którym odpowiedź skokowa praktycznie osiąga stan ustalony.




Rys. 2.2. Odpowiedź skokowa obiektu dyskretnego wykorzystana do wyznaczenia horyzontu dynamiki $D$

### 2.4.2. Badanie wpływu horyzontu predykcji $N$

W regulatorze DMC horyzont predykcji $N$ określa, na ile kroków w przyszłość prognozowane jest zachowanie obiektu. Aby zbadać wpływ tego parametru na jakość regulacji, przeprowadzono symulację dla różnych wartości $N$ przy ustalonych pozostałych parametrach. Na rysunku 2.3 przedstawiono porównanie odpowiedzi układu dla wartości $N \in \{20, 30, 40, 50, 60\}$, przy założeniu $N_u = N$ i $\lambda = 1$.

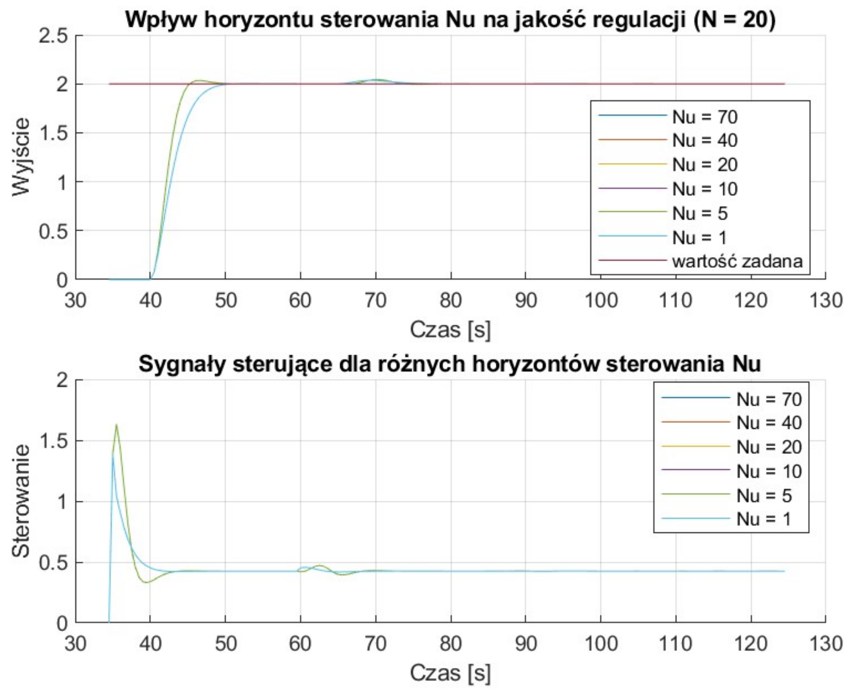



Rys. 2.3. Wpływ horyzontu predykcji $N$ na jakość regulacji (przy $N_u = N$)

Z analizy wynika, że dla $N < 30$ układ wykazuje słabsze tłumienie oscylacji. Przy większych wartościach odpowiedzi stają się bardziej stabilne, ale przy $N > 40$ dalsze zwiększanie horyzontu predykcji nie przynosi już znaczącej poprawy. Jako optymalną wybrano wartość $N = 20$.

### 2.4.3. Badanie wpływu horyzontu sterowania $N_u$

Horyzont sterowania $N_u$ określa, na ile kroków w przyszłość obliczane są przyrosty sterowania. Aby zbadać wpływ tego parametru, przeprowadzono symulacje dla stałego horyzontu predykcji $N = 20$ i różnych wartości $N_u \in \{1, 5, 10, 20, 40, 60\}$.

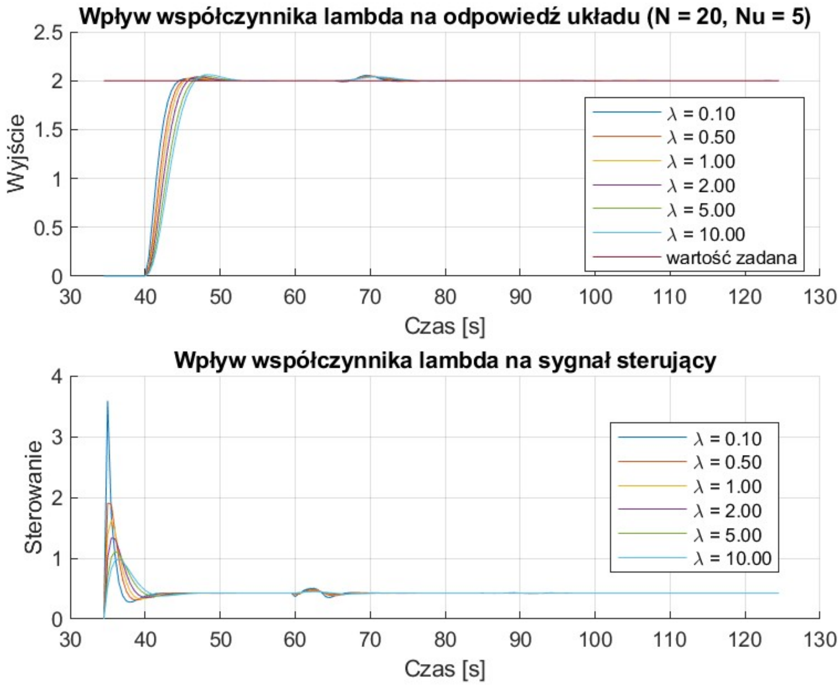



Rys. 2.4. Wpływ horyzontu sterowania $N_u$ na jakość regulacji (przy $N = 20$)

Z rysunku 2.4 wynika, że dla małych wartości $N_u$ (szczególnie $N_u = 1$) odpowiedź układu jest wolniejsza, ale za to łagodniejsza. Wraz ze wzrostem $N_u$ odpowiedź staje się szybsza, ale pojawia się przesterowanie i większe oscylacje sterowania. Jako kompromis między szybkością a jakością wybrano $N_u = 5$.

### 2.4.4. Badanie wpływu współczynnika kary za sterowanie $\lambda$

Współczynnik $\lambda$ odpowiada za wagę członu kary za przyrosty sterowania w funkcji celu regulatora DMC. Im większa wartość $\lambda$, tym większa kara za duże zmiany sterowania, co prowadzi do łagodniejszego działania regulatora.




Rys. 2.5. Wpływ współczynnika kary $\lambda$ na jakość regulacji (przy $N = 20$, $N_u = 5$)

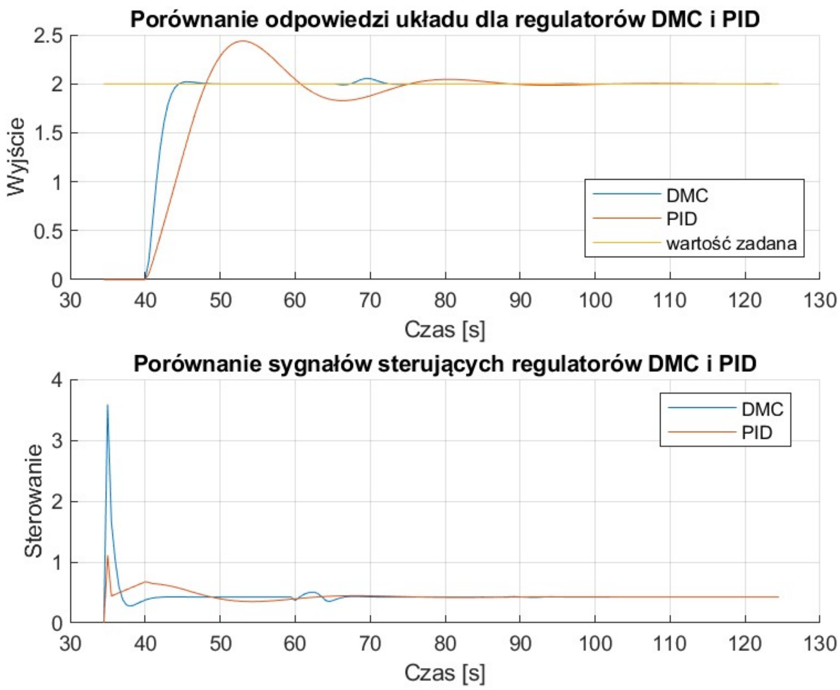
Z rysunku 2.5 wynika, że dla małych wartości $\lambda$ (np. $0,1$) układ reaguje szybko, ale sterowanie może być agresywne. Dla większych wartości $\lambda$ (np. 5 czy 10) odpowiedź jest wolniejsza i łagodniejsza. Jako optymalną wartość wybrano $\lambda = 0,1$, która zapewnia szybką odpowiedź przy akceptowalnym profilu sterowania.

Po przeprowadzeniu powyższych badań ustalono optymalne parametry regulatora DMC:

— Horyzont predykcji $N = 20$
— Horyzont sterowania $N_u = 5$
— Współczynnik kary $\lambda = 0,1$

## 2.5. Porównanie regulatorów DMC i PID

Po określeniu optymalnych parametrów regulatora DMC przeprowadzono porównanie jego działania z regulatorem PID dostrojonym metodą Zieglera-Nicholsa. Rysunek 2.6 przedstawia porównanie odpowiedzi układu oraz sygnałów sterujących dla obu regulatorów.



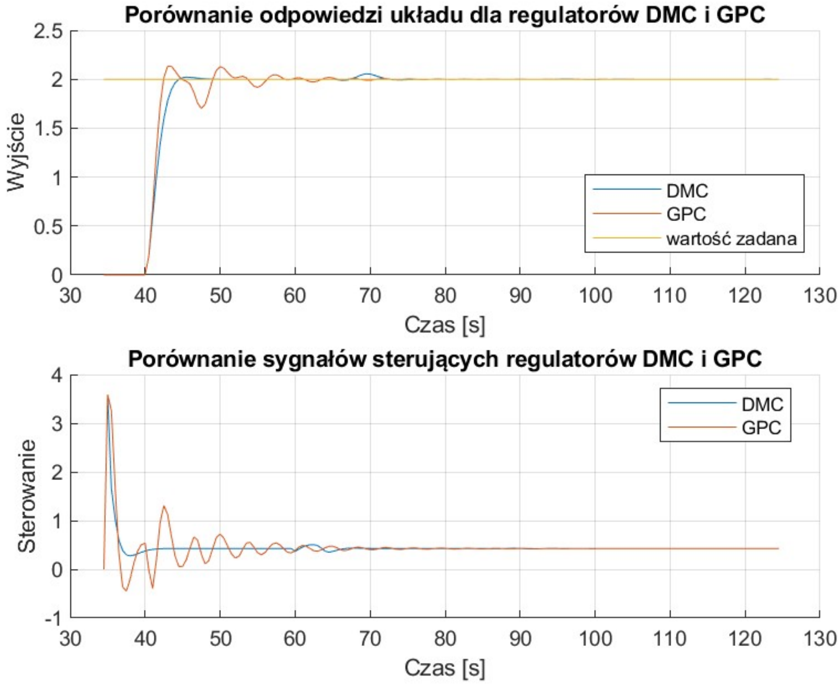

Rys. 2.6. Porównanie odpowiedzi układu z regulatorem DMC i PID

Regulator DMC wykazuje lepszą jakość regulacji niż PID - charakteryzuje się mniejszym przeregulowaniem, szybszym czasem regulacji i lepszym tłumieniem oscylacji. Sygnał sterujący generowany przez DMC ma łagodniejszy przebieg niż w przypadku PID, co jest korzystne z punktu widzenia elementów wykonawczych.

## 2.6. Porównanie regulatorów DMC i GPC

### 2.6.1. Reakcja na zmianę wartości zadanej

Porównano działanie regulatorów DMC i GPC przy jednakowych parametrach ($N = 20$, $N_u = 5$, $\lambda = 0,1$). Na rysunku 2.7 przedstawiono odpowiedź układu na zmianę wartości zadanej.

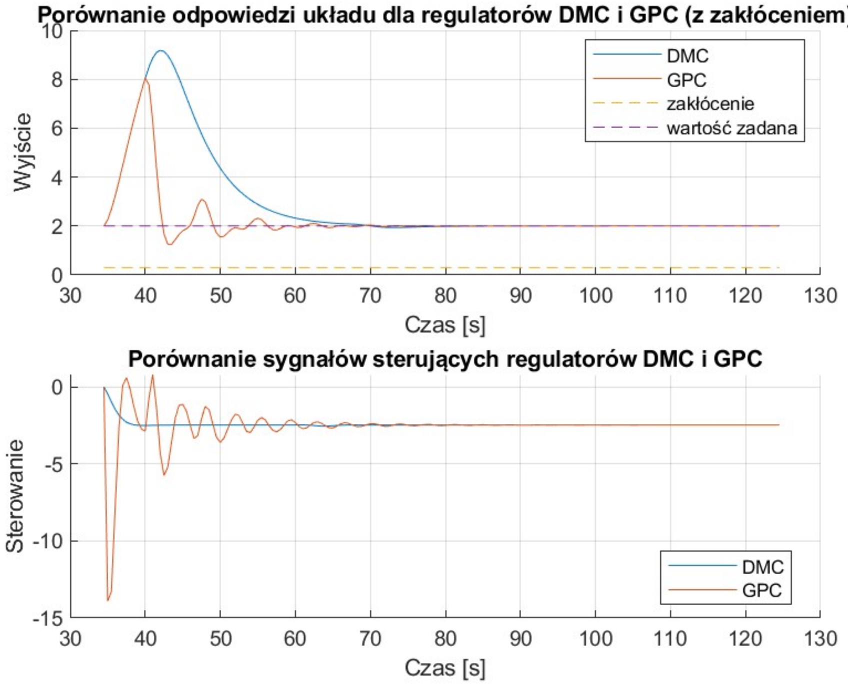



Rys. 2.7. Porównanie odpowiedzi układu z regulatorami DMC i GPC na zmianę wartości zadanej

Zarówno regulator DMC, jak i GPC zapewniają podobną jakość regulacji, jednak GPC charakteryzuje się nieco szybszą odpowiedzią i mniejszym przeregulowaniem. Wynika to z różnego sposobu modelowania obiektu w tych algorytmach - DMC wykorzystuje model odpowiedzi skokowej, a GPC - model w postaci równania różnicowego.

### 2.6.2. Reakcja na zakłócenie

Przeprowadzono również test odporności na zakłócenie działające na wyjściu obiektu. Na rysunku 2.8 przedstawiono odpowiedź układu na zakłócenie o wartości $0,3$.




Rys. 2.8. Porównanie odpowiedzi układu z regulatorami DMC i GPC na zakłócenie

W przypadku zakłócenia regulator GPC wykazuje szybszą reakcję i lepsze tłumienie jego wpływu. Regulator DMC również skutecznie kompensuje zakłócenie, ale potrzebuje nieco więcej czasu. Wynika to z różnych modeli procesu i algorytmu predykcji zakłóceń.

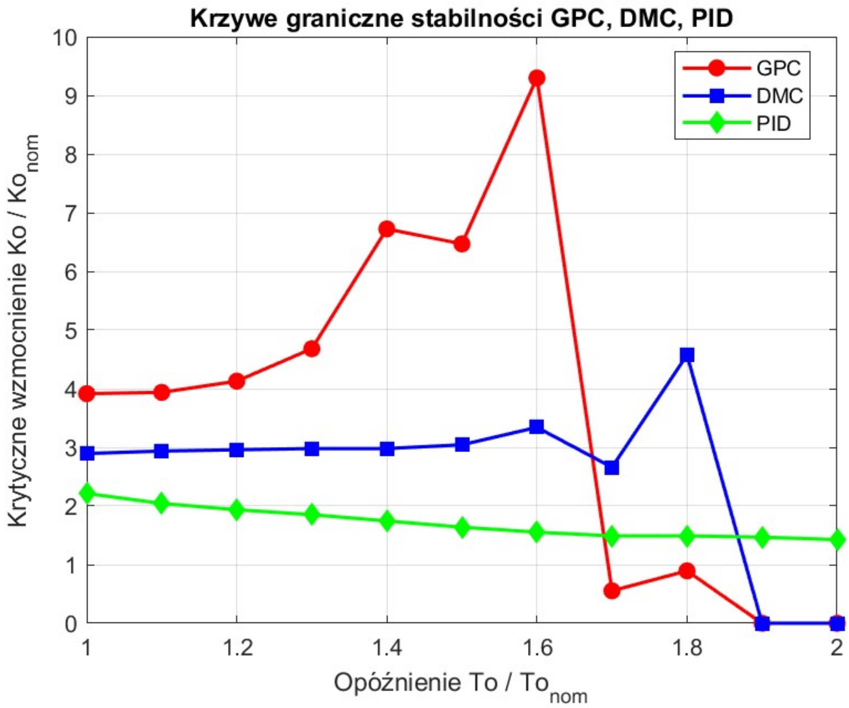
## 2.7. Badanie obszarów stabilności

W ostatnim etapie projektu przeprowadzono badanie obszarów stabilności regulatorów PID, DMC i GPC w funkcji zmiany parametrów obiektu: wzmocnienia $K_o$ i opóźnienia transportowego $T_o$. Zbadano wartości krytyczne wzmocnienia $K_o$ dla różnych wartości opóźnienia $T_o$, przy których układ regulacji traci stabilność.

Wyniki symulacji:

— To=5,00, Ko_kryt: GPC=18,40, DMC=13,60, PID=10,40
— To=5,50, Ko_kryt: GPC=18,50, DMC=13,80, PID=9,60
— To=6,00, Ko_kryt: GPC=19,40, DMC=13,90, PID=9,10
— To=6,50, Ko_kryt: GPC=22,00, DMC=14,00, PID=8,70
— To=7,00, Ko_kryt: GPC=31,60, DMC=14,00, PID=8,20
— To=7,50, Ko_kryt: GPC=30,40, DMC=14,30, PID=7,70
— To=8,00, Ko_kryt: GPC=43,70, DMC=15,70, PID=7,30
— To=8,50, Ko_kryt: GPC=2,60, DMC=12,50, PID=7,00
— To=9,00, Ko_kryt: GPC=4,20, DMC=21,50, PID=7,00
— To=9,50, Ko_kryt: GPC=0,00, DMC=0,00, PID=6,90
— To=10,00, Ko_kryt: GPC=0,00, DMC=0,00, PID=6,70

Na rysunku 2.9 przedstawiono otrzymane krzywe graniczne stabilności.




Rys. 2.9. Krzywe graniczne stabilności dla regulatorów GPC, DMC i PID w funkcji opóźnienia $T_o$ i wzmocnienia $K_o$

Na osiach wykresu przedstawiono względne zmiany parametrów $T_o/T_{o,nom}$ oraz $K_o/K_{o,nom}$. Obszar stabilności leży poniżej krzywych dla każdego z regulatorów - dla danego opóźnienia $T_o/T_{o,nom}$, system jest stabilny jeśli $K_o/K_{o,nom} < K_{o,kryt}/K_{o,nom}$.

Widoczne jest, że regulatory predykcyjne (DMC i GPC) mają znacznie szerszy obszar stabilności niż regulator PID, szczególnie dla większych opóźnień. Regulator GPC wykazuje największą odporność na wzrost opóźnienia transportowego, co jest jego istotną zaletą w zastosowaniach przemysłowych.

Warto odnotować, że dla bardzo dużych opóźnień (To $>$ 9,0) wszystkie regulatory tracą stabilność nawet przy niewielkim wzroście wzmocnienia obiektu. Jest to zgodne z intuicją, ponieważ duże opóźnienie transportowe jest jednym z najtrudniejszych wyzwań w regulacji.

# 3. Wnioski

Na podstawie przeprowadzonych badań można sformułować następujące wnioski:

1. Regulatory predykcyjne (DMC i GPC) zapewniają lepszą jakość regulacji niż klasyczny regulator PID, szczególnie dla obiektów z opóźnieniem transportowym. Wykazują mniejsze przeregulowanie, krótszy czas regulacji i lepsze tłumienie oscylacji.
2. Parametry regulatora DMC ($N$, $N_u$, $\lambda$) mają istotny wpływ na jakość regulacji. Horyzont predykcji $N$ powinien być wystarczająco duży, aby uwzględnić pełną odpowiedź obiektu. Horyzont sterowania $N_u$ wpływa na agresywność sterowania - większe wartości prowadzą do szybszej, ale bardziej oscylacyjnej odpowiedzi. Współczynnik $\lambda$ pozwala na kompromis między szybkością regulacji a łagodnością sterowania.
3. Regulator GPC wykazuje lepsze właściwości niż DMC w zakresie kompensacji zakłóceń oraz odporności na zmiany parametrów obiektu, szczególnie opóźnienia transportowego. Wynika to z odmiennego sposobu modelowania obiektu i predykcji.
4. Obszar stabilności regulatorów predykcyjnych jest znacznie szerszy niż regulatora PID, co czyni je bardziej odpornymi na zmiany parametrów obiektu. Jest to szczególnie istotne w przypadku procesów o zmiennych parametrach lub niepewności modelowania.

Podsumowując, regulatory predykcyjne stanowią efektywne narzędzie do sterowania procesami, szczególnie w przypadku obiektów trudnych (z opóźnieniem, niestabilnych, wielowymiarowych). Ich implementacja wymaga jednak dokładniejszego modelowania procesu oraz większej mocy obliczeniowej niż w przypadku klasycznych regulatorów PID.

# 4. Kod źródłowy

Poniżej przedstawiono listę skryptów MATLAB wykorzystanych w projekcie. Szczegółowe treści tych skryptów znajdują się w katalogu `/home/jszubzda/STP_2/kod/`.

## 4.1. Główny skrypt projektu

```
% --- Główny skrypt projektu ---
clearvars;
clc;
close all;

% Parametry obiektu
Ko = 4.7;
To = 5;
T1 = 1.92;
T2 = 4.96;
Tp = 0.5;

% Zadanie 1
disp('--- Zadanie 1 ---');
[Gs, Gz] = zadanie1_analiza_transmitancji(Ko, To, T1, T2, Tp);

% Zadanie 2
disp('--- Zadanie 2 ---');
zadanie2_rownanie_roznicowe(Gz);

% Zadanie 3
disp('--- Zadanie 3 ---');
[param_pid, param_pid_ciagly] = zadanie3_strojenie_pid_ziegler_nichols(Gs, Tp);

% Parametry symulacji
param_sym = struct('len', 250, 'tp', 0.5, 'setpoint', 2);
wart_zad = ones(1, param_sym.len - 69) * 2;

% Zadanie 5
disp('--- Zadanie 5 ---');
param_dmc_opt = zadanie5_optymalizacja_dmc(Gz, param_pid, param_sym, wart_zad);

% Zadanie 6
disp('--- Zadanie 6 ---');
zadanie6_porownanie_optymalne_dmc_pid(Gz, param_pid, param_dmc_opt, param_sym, wart_zad);

% Zadanie 8
disp('--- Zadanie 8 ---');
param_gpc = param_dmc_opt; % Parametry GPC takie same jak DMC
zadanie8_porownanie_dmc_gpc(Gz, param_dmc_opt, param_gpc, param_pid, param_sym, wart_zad);

% Zadanie 9
disp('--- Zadanie 9 ---');
zadanie9_badanie_stabilnosci(param_pid, param_dmc_opt, param_gpc, Ko, To, T1, T2, Tp, param_s
```

```
disp('--- Koniec projektu ---');
```

Listing 4.1. Główny skrypt projektu - stp2.m

## 4.2. Funkcje do analizy transmitancji i wyprowadzenia równania różnicowego

```
function [Gs, Gz] = zadanie1_analiza_transmitancji(Ko, To, T1, T2, Tp)
%% zadanie 1
Gs = tf(Ko, conv([T1, 1], [T2, 1]), 'InputDelay', To);

disp('--- Zadanie 1 ---');
disp('Transmitancja ciągła G(s):');
Gs

Gz = c2d(Gs, Tp, 'zoh');

disp('Transmitancja dyskretna G(z):');
Gz

K_statyczne_s = dcgain(Gs);
K_statyczne_z = dcgain(Gz);

disp(['Współczynnik wzmocnienia statycznego G(s), K_s = G(0): ', num2str(K_statyczne_s)]);
disp(['Współczynnik wzmocnienia statycznego G(z), K_z = G(1): ', num2str(K_statyczne_z)]);

fig = figure;
step(Gs, '-b');
hold on;
step(Gz, '-r');
title('Porównanie odpowiedzi skokowych G(s) i G(z)');
legend('G(s) - ciągła', 'G(z) - dyskretna', 'Location','best');
grid on;
hold off;
saveas(fig, "wykresy/zad1.jpg");
close;
end
```

Listing 4.2. zadanie1_analiza_transmitancji.m

```
function zadanie2_rownanie_roznicowe(Gz)
%% zadanie 2
licz_Gz = Gz.Numerator{1};
mian_Gz = Gz.Denominator{1};
opoz = Gz.InputDelay;

rownanie = 'y(k) = ';
pierwszy_term = true;

stopien_mian = length(mian_Gz) - 1;

disp('Równanie różnicowe:');
for i = 1:stopien_mian
    wsp = -mian_Gz(i+1);
    znak = '';
    if wsp > 0
        if ~pierwszy_term
            znak = ' + ';
        end
    else
        znak = ' - ';
        wsp = -wsp;
```

```
    end

    term = [num2str(wsp, 5), '*y(k-', num2str(i), ')'];
    rownanie = [rownanie, znak, term];
    pierwszy_term = false;
end

stopien_licz = length(licz_Gz) - 1;
max_opoz = opoz + stopien_licz;

for l = 0:stopien_licz
    wsp = licz_Gz(l+1);
    biezace_opoz = opoz + l;
    if biezace_opoz >= 1
        znak = '';
        if wsp > 0
            if ~pierwszy_term
                znak = ' + ';
            end
        else
            znak = ' - ';
            wsp = -wsp;
        end

        term = [num2str(wsp, 5), '*u(k-', num2str(biezace_opoz), ')'];
        rownanie = [rownanie, znak, term];
        pierwszy_term = false;
    end
end

disp(rownanie);
end
```

Listing 4.3. zadanie2_rownanie_roznicowe.m

## 4.3. Funkcje do strojenia regulatora PID

```
function [param_pid, param_pid_ciagly] = zadanie3_strojenie_pid_ziegler_nichols(Gs, Tp)
%% zadanie 3
[Gm_abs, ~, ~, Wcp_rad_s] = margin(Gs);

Kk = Gm_abs;
Tk = 2*pi / Wcp_rad_s;

disp('Parametry krytyczne (Ziegler-Nichols):');
disp(['Wzmocnienie krytyczne Kk = ', num2str(Kk)]);
disp(['Okres oscylacji krytycznych Tk = ', num2str(Tk), ' s']);
disp(['Pulsacja krytyczna omega_k = ', num2str(Wcp_rad_s), ' rad/s']);

detuning_factor = 0.7;
Kr = 0.6 * Kk * detuning_factor;
Ti = 0.5 * Tk / detuning_factor;
Td = 0.12 * Tk * detuning_factor;

disp('Parametry ciągłego regulatora PID (wg Zieglera-Nicholsa):');
disp(['Kr = ', num2str(Kr)]);
disp(['Ti = ', num2str(Ti), ' s']);
disp(['Td = ', num2str(Td), ' s']);

Kp = Kr;
Ki = Kr / Ti;
```

```
Kd = Kr * Td;

disp('Współczynniki Kp, Ki, Kd dla ciągłego regulatora PID:');
disp(['Kp = ', num2str(Kp)]);
disp(['Ki = ', num2str(Ki)]);
disp(['Kd = ', num2str(Kd)]);

r0 = Kp + Ki*Tp + Kd/Tp;
r1 = -(Kp + 2*(Kd/Tp));
r2 = Kd/Tp;

param_pid_ciagly = struct('Kp', Kp, 'Ki', Ki, 'Kd', Kd, 'Kr', Kr, 'Ti', Ti, 'Td', Td);
param_pid = struct('r0', r0, 'r1', r1, 'r2', r2);

disp('Parametry r0, r1, r2 dyskretnego regulatora PID:');
disp(['Przyjęty okres próbkowania Tp = ', num2str(Tp), ' s']);
disp(['r0 = ', num2str(r0)]);
disp(['r1 = ', num2str(r1)]);
disp(['r2 = ', num2str(r2)]);
end
```

Listing 4.4. zadanie3_strojenie_pid_ziegler_nichols.m

## 4.4. Funkcje do projektowania i badania regulatora DMC

```
function [param_dmc_opt] = zadanie5_optymalizacja_dmc(Gz, param_pid, param_sym, wart_zad)
%% zadanie 5

% a) Odpowiedź skokowa
fig = figure;
step(Gz);
title('Odpowiedź skokowa obiektu G(z)');
saveas(fig, 'wykresy/step.jpg');
close;

% b) Wpływ horyzontu predykcji N
wartosci_N = [70, 50, 40, 30, 20];
uchwyty_wykr = [];
figure;
subplot(2,1,1);
hold on;
grid on;
title('Wpływ horyzontu predykcji N na jakość regulacji (przy Nu=N)');
xlabel('Czas [s]');
ylabel('Wyjście');

subplot(2,1,2);
hold on;
grid on;
title('Sygnały sterujące dla różnych horyzontów predykcji N');
xlabel('Czas [s]');
ylabel('Sterowanie');
legendy = {};

for i = 1:length(wartosci_N)
    N_test = wartosci_N(i);
    param_dmc = struct('N', N_test, 'Nu', N_test, 'lambda', 1, 'D', 70);
    [~, y_dmc, ~, u_dmc, czas_sym, D] = symulacja_dmc_pid(param_dmc, param_sym, param_pid, Gz

    subplot(2,1,1);
    uchwyty_wykr(end+1) = plot(czas_sym(D:end), y_dmc(D:end), '-');
```

```
    subplot(2,1,2);
    plot(czas_sym(D:end), u_dmc(D:end), '-');

    legendy{end+1} = sprintf('N = Nu = %d', N_test);
end

subplot(2,1,1);
uchwyty_wykr(end+1) = plot(czas_sym(D:end), wart_zad, '-');
legendy{end+1} = 'wartość zadana';

subplot(2,1,1);
legend(uchwyty_wykr, legendy, 'Location', 'best');

subplot(2,1,2);
legend(legendy{1:end-1}, 'Location', 'best');

saveas(gcf, 'wykresy/horyzont_predykcji_porownanie.jpg');
close;

N_opt = 20;

% c) Wpływ horyzontu sterowania Nu
wartosci_Nu = [70, 40, 20, 10, 5, 1];

figure;
subplot(2,1,1);
hold on;
grid on;
title(['Wpływ horyzontu sterowania Nu na jakość regulacji (N = ', num2str(N_opt), ')']);
xlabel('Czas [s]');
ylabel('Wyjście');

subplot(2,1,2);
hold on;
grid on;
title('Sygnały sterujące dla różnych horyzontów sterowania Nu');
xlabel('Czas [s]');
ylabel('Sterowanie');

legendy = {};

for i = 1:length(wartosci_Nu)
    Nu_test = wartosci_Nu(i);
    param_dmc = struct('N', N_opt, 'Nu', Nu_test, 'lambda', 1, 'D', 70);

    [~, y_dmc, ~, u_dmc, czas_sym, D] = symulacja_dmc_pid(param_dmc, param_sym, param_pid, Gz

    subplot(2,1,1);
    plot(czas_sym(D:end), y_dmc(D:end), '-');

    subplot(2,1,2);
    plot(czas_sym(D:end), u_dmc(D:end), '-');

    legendy{end+1} = sprintf('Nu = %d', Nu_test);
end

subplot(2,1,1);
plot(czas_sym(D:end), wart_zad, '-');
legendy{end+1} = 'wartość zadana';

subplot(2,1,1);
legend(legendy, 'Location', 'best');
```

```
subplot(2,1,2);
legend(legendy{1:end-1}, 'Location', 'best');

saveas(gcf, 'wykresy/horyzont_sterowania_porownanie.jpg');
close;

Nu_opt = 5;

% d) Wpływ współczynnika lambda
wartosci_lambda = [0.1, 0.5, 1, 2, 5, 10];

figure;
subplot(2,1,1);
hold on;
grid on;
title(['Wpływ współczynnika lambda na odpowiedź układu (N = ', num2str(N_opt), ', Nu = ', num
xlabel('Czas [s]');
ylabel('Wyjście');

subplot(2,1,2);
hold on;
grid on;
title('Wpływ współczynnika lambda na sygnał sterujący');
xlabel('Czas [s]');
ylabel('Sterowanie');

legendy = {};

for i = 1:length(wartosci_lambda)
    lambda_test = wartosci_lambda(i);
    param_dmc = struct('N', N_opt, 'Nu', Nu_opt, 'lambda', lambda_test, 'D', 70);

    [~, y_dmc, ~, u_dmc, czas_sym, D] = symulacja_dmc_pid(param_dmc, param_sym, param_pid, Gz

    subplot(2,1,1);
    plot(czas_sym(D:end), y_dmc(D:end), '-');

    subplot(2,1,2);
    plot(czas_sym(D:end), u_dmc(D:end), '-');

    legendy{end+1} = sprintf('\\lambda = %.2f', lambda_test);
end

subplot(2,1,1);
plot(czas_sym(D:end), wart_zad, '-');
legendy{end+1} = 'wartość zadana';

subplot(2,1,1);
legend(legendy, 'Location', 'best');

subplot(2,1,2);
legend(legendy{1:end-1}, 'Location', 'best');

saveas(gcf, 'wykresy/lambda_porownanie.jpg');
close;

lambda_opt = 0.1;

% Optymalne parametry DMC
param_dmc_opt = struct('N', N_opt, 'Nu', Nu_opt, 'lambda', lambda_opt, 'D', 70);

disp('--- Optymalne parametry regulatora DMC ---');
```

```
disp(['N = ', num2str(N_opt)]);
disp(['Nu = ', num2str(Nu_opt)]);
disp(['lambda = ', num2str(lambda_opt)]);
end
```

Listing 4.5. zadanie5_optymalizacja_dmc.m

```
function zadanie6_porownanie_optymalne_dmc_pid(Gz, param_pid, param_dmc_opt, param_sym, wart_
%% zadanie 6
figure;
subplot(2,1,1);
hold on;
grid on;
title('Porównanie odpowiedzi układu dla regulatorów DMC i PID');
xlabel('Czas [s]');
ylabel('Wyjście');

subplot(2,1,2);
hold on;
grid on;
title('Porównanie sygnałów sterujących regulatorów DMC i PID');
xlabel('Czas [s]');
ylabel('Sterowanie');

[y_pid, y_dmc, u_pid, u_dmc, czas_sym, D] = symulacja_dmc_pid(param_dmc_opt, param_sym, param

subplot(2,1,1);
plot(czas_sym(D:end), y_dmc(D:end), czas_sym(D:end), y_pid(D:end), czas_sym(D:end), wart_zad,

subplot(2,1,2);
plot(czas_sym(D:end), u_dmc(D:end), czas_sym(D:end), u_pid(D:end), '-');

subplot(2,1,1);
legend('DMC', 'PID', 'wartość zadana', 'Location', 'best');

subplot(2,1,2);
legend('DMC', 'PID', 'Location', 'best');

saveas(gcf, 'wykresy/optimal_solution.jpg');
close;
end
```

Listing 4.6. zadanie6_porownanie_optymalne_dmc_pid.m

## 4.5. Funkcje do porównania regulatorów DMC i GPC

```
function zadanie8_porownanie_dmc_gpc(Gz, param_dmc, param_gpc, param_pid, param_sym, wart_zad
%% zadanie 8
% a) Porównanie dla zmiany wartości zadanej
figure;
subplot(2,1,1);
hold on;
grid on;
title('Porównanie odpowiedzi układu dla regulatorów DMC i GPC');
xlabel('Czas [s]');
ylabel('Wyjście');

subplot(2,1,2);
hold on;
grid on;
title('Porównanie sygnałów sterujących regulatorów DMC i GPC');
```

```
xlabel('Czas [s]');
ylabel('Sterowanie');

[~, y_dmc, ~, u_dmc, ~, ~] = symulacja_dmc_pid(param_dmc, param_sym, param_pid, Gz);
[y_gpc, u_gpc, czas_sym, D] = symulacja_gpc(param_gpc, param_sym, Gz);

subplot(2,1,1);
plot(czas_sym(D:end), y_dmc(D:end), czas_sym(D:end), y_gpc(D:end), czas_sym(D:end), wart_zad,

subplot(2,1,2);
plot(czas_sym(D:end), u_dmc(D:end), czas_sym(D:end), u_gpc(D:end), '-');

subplot(2,1,1);
legend('DMC', 'GPC', 'wartość zadana', 'Location', 'best');

subplot(2,1,2);
legend('DMC', 'GPC', 'Location', 'best');

saveas(gcf, 'wykresy/DMC_vs_GPC_setpoint.jpg');
close;

% b) Porównanie dla zakłócenia
zaklocenie = 0.3;

figure;
subplot(2,1,1);
hold on;
grid on;
title('Porównanie odpowiedzi układu dla regulatorów DMC i GPC (z zakłóceniem)');
xlabel('Czas [s]');
ylabel('Wyjście');

subplot(2,1,2);
hold on;
grid on;
title('Porównanie sygnałów sterujących regulatorów DMC i GPC');
xlabel('Czas [s]');
ylabel('Sterowanie');

[~, y_dmc, ~, u_dmc, ~, ~] = symulacja_dmc_pid(param_dmc, param_sym, param_pid, Gz, zakloceni
[y_gpc, u_gpc, czas_sym, D] = symulacja_gpc(param_gpc, param_sym, Gz, zaklocenie, true);

subplot(2,1,1);
plot(czas_sym(D:end), y_dmc(D:end), czas_sym(D:end), y_gpc(D:end), '-');
plot(czas_sym(D:end), ones(1, length(czas_sym(D:end)))*zaklocenie, '--', czas_sym(D:end), war

subplot(2,1,2);
plot(czas_sym(D:end), u_dmc(D:end), czas_sym(D:end), u_gpc(D:end), '-');

subplot(2,1,1);
legend('DMC', 'GPC', 'zakłócenie', 'wartość zadana', 'Location', 'best');

subplot(2,1,2);
legend('DMC', 'GPC', 'Location', 'best');

saveas(gcf, 'wykresy/DMC_vs_GPC_disturbance.jpg');
close;
end
```

Listing 4.7. zadanie8_porownanie_dmc_gpc.m

### 4.6. Funkcje do badania stabilności

```
function zadanie9_badanie_stabilnosci(param_pid, param_dmc, param_gpc, Ko, To, T1, T2, Tp, pa
%% zadanie 9
mnozniki_To = [1, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2];
Ko_kryt_gpc = zeros(size(mnozniki_To));
Ko_kryt_dmc = zeros(size(mnozniki_To));
Ko_kryt_pid = zeros(size(mnozniki_To));

for i = 1:length(mnozniki_To)
    To_test = mnozniki_To(i) * To;
    Ko_test = 0.1;
    Ko_max = 50;
    znaleziono_gpc = false;
    znaleziono_dmc = false;
    znaleziono_pid = false;
    while Ko_test < Ko_max && (~znaleziono_gpc || ~znaleziono_dmc || ~znaleziono_pid)
        Gs_test = tf(Ko_test, conv([T1, 1], [T2, 1]), 'InputDelay', To_test);
        Gz_test = c2d(Gs_test, Tp, 'zoh');
        [y_pid, y_dmc, ~, ~, ~, ~] = symulacja_dmc_pid(param_dmc, param_sym, param_pid, Gz_te
        [y_gpc, ~, ~, ~] = symulacja_gpc(param_gpc, param_sym, Gz_test);

        if czy_oscyluje(y_gpc, 8, 0.01, 2) && ~znaleziono_gpc
            Ko_kryt_gpc(i) = Ko_test;
            znaleziono_gpc = true;
        end
        if any(y_pid > 7) && ~znaleziono_pid
            Ko_kryt_pid(i) = Ko_test;
            znaleziono_pid = true;
        end
        if czy_oscyluje(y_dmc, 8, 0.01, 1) && ~znaleziono_dmc
            Ko_kryt_dmc(i) = Ko_test;
            znaleziono_dmc = true;
        end
        Ko_test = Ko_test + 0.1;
    end
    if ~znaleziono_gpc, Ko_kryt_gpc(i) = 0; end
    if ~znaleziono_dmc, Ko_kryt_dmc(i) = 0; end
    if ~znaleziono_pid, Ko_kryt_pid(i) = 0; end
    fprintf('To=%.2f, Ko_kryt: GPC=%.2f, DMC=%.2f, PID=%.2f\n', To_test, Ko_kryt_gpc(i), Ko_k
end

figure;
plot(mnozniki_To, Ko_kryt_gpc/Ko, 'r-o', 'LineWidth', 1.5, 'MarkerFaceColor', 'r'); hold on;
plot(mnozniki_To, Ko_kryt_dmc/Ko, 'b-s', 'LineWidth', 1.5, 'MarkerFaceColor', 'b');
plot(mnozniki_To, Ko_kryt_pid/Ko, 'g-d', 'LineWidth', 1.5, 'MarkerFaceColor', 'g');
grid on;
title('Krzywe graniczne stabilności GPC, DMC, PID');
xlabel('Opóźnienie To / To_{nom}');
ylabel('Krytyczne wzmocnienie Ko / Ko_{nom}');
legend('GPC', 'DMC', 'PID', 'Location', 'best');
saveas(gcf, 'wykresy/GPC_DMC_PID_stabilnosc_Ko_vs_To.jpg');
close;
disp(' ');
disp('Obszar stabilności: Dla danego To/To_nom, system jest stabilny dla Ko/Ko_nom < Ko_kryt/
disp('System jest niestabilny dla Ko > Ko_kryt (powyżej krzywej).');
end
```

Listing 4.8. zadanie9_badanie_stabilnosci.m

```
function czy_osc = czy_oscyluje(y, liczba_szczytow, tolerancja, typ)
    czy_osc = false;
    % Analizujemy tylko końcowy fragment sygnału, aby uniknąć stanów przejściowych
```

```
    if length(y) > 100
        y = y(end-100:end);
    end
    [szczyty, ~] = findpeaks(y);
    switch typ
        case 1 % Dla DMC: sprawdzanie trendu wzrostowego ostatnich szczytów
            if length(szczyty) > liczba_szczytow
                % Analizujemy trend ostatnich 'liczba_ostatnich_szczytow' szczytów
                trend = diff(szczyty(end-liczba_szczytow:end));
                % Jeśli wszystkie przyrosty są większe od tolerancji, uznajemy za oscylacje n
                if all(trend > tolerancja)
                    czy_osc = true;
                end
            end
        case 2 % Dla GPC: prostsze sprawdzenie, czy ostatnie szczyty są wyższe
            if length(szczyty) > 5 % Potrzebujemy wystarczającej liczby szczytów do analizy
                % Sprawdzamy, czy ostatni zanotowany szczyt jest znacząco wyższy niż maksimum
                % To uproszczone kryterium, może wymagać dostosowania
                czy_osc = (max(szczyty(1:4)) - szczyty(end) < 0);
            end
    end
end
```

Listing 4.9. czy_oscyluje.m

## 4.7. Funkcje pomocnicze do symulacji

```
function [y_pid, y_dmc, u_pid, u_dmc, czas_sym, D] = symulacja_dmc_pid(param_dmc, param_sym,
%% symulacja dmc i pid

if nargin < 5
    zaklocenie = 0;
end

if nargin < 6
    stan_ustalony = false;
end

dlugosc = param_sym.len;
czas_sym = (0:param_sym.len-1)*param_sym.tp;
wart_zad = param_sym.setpoint;
licz_Gz = Gz.Numerator{1};
mian_Gz = Gz.Denominator{1};
opoz = Gz.InputDelay;

r0 = param_pid.r0;
r1 = param_pid.r1;
r2 = param_pid.r2;

y_pid = zeros(1, dlugosc);
u_pid = zeros(1, dlugosc);
e_pid = zeros(1, dlugosc);

N = param_dmc.N;
Nu = param_dmc.Nu;
lambda = param_dmc.lambda;
D = param_dmc.D;

yzad = wart_zad * ones(N, 1);

y_dmc = zeros(1, dlugosc);
```

```
u_dmc = zeros(1, dlugosc);
du_dmc = zeros(1, dlugosc);

if stan_ustalony
    y_dmc(1:D) = wart_zad;
end

s = step(Gz, (0:D-1)*param_sym.tp);
s = s(1:D);

M = zeros(param_dmc.N, Nu);
for i = 1:N
    for j = 1:min(i, Nu)
        M(i, j) = s(i-j+1);
    end
end

s = s(2:D);

K = inv(M'*eye(N)*M + lambda*eye(Nu)) * M'*eye(N);

for k = max(3, D+1):dlugosc
    for i = 1:length(mian_Gz)-1
        y_pid(k) = y_pid(k) - mian_Gz(i+1) * y_pid(k-i);
        y_dmc(k) = y_dmc(k) - mian_Gz(i+1) * y_dmc(k-i);
    end

    for i = 1:length(licz_Gz)
        idx = k - opoz - i + 1;
        y_pid(k) = y_pid(k) + licz_Gz(i) * u_pid(idx);
        y_dmc(k) = y_dmc(k) + licz_Gz(i) * u_dmc(idx);
    end

    y_dmc(k) = y_dmc(k) + zaklocenie;

    e_pid(k) = wart_zad - y_pid(k);
    u_pid(k) = u_pid(k-1) + r0*e_pid(k) + r1*e_pid(k-1) + r2*e_pid(k-2);

    dUp = zeros(D-1, 1);
    for i = 1:min(D-1, k-1)
        dUp(i) = u_dmc(k-i) - u_dmc(k-i-1);
    end

    y0 = zeros(N, 1);
    for p = 1:N
        y0(p) = y_dmc(k);
        for i = 1:min(D-1, k-1)
            if p+i <= D-1
                y0(p) = y0(p) + (s(p+i) - s(i)) * dUp(i);
            end
        end
    end
    dU = K * (yzad - y0);
    du_dmc(k) = dU(1);

    u_dmc(k) = u_dmc(k-1) + du_dmc(k);
end
end
```

Listing 4.10. symulacja_dmc_pid.m

```
function [y_gpc, u_gpc, czas_sym, D] = symulacja_gpc(param_gpc, param_sym, Gz, zaklocenie, st
%% symulacja gpc
```

```
if nargin < 4
    zaklocenie = 0;
end

if nargin < 5
    stan_ustalony = false;
end

dlugosc = param_sym.len;
czas_sym = (0:param_sym.len-1)*param_sym.tp;
wart_zad = param_sym.setpoint;
licz_Gz = Gz.Numerator{1};
mian_Gz = Gz.Denominator{1};
opoz = Gz.InputDelay;

N = param_gpc.N;
Nu = param_gpc.Nu;
lambda = param_gpc.lambda;
D = param_gpc.D;

yzad = wart_zad * ones(N, 1);

y_gpc = zeros(1, dlugosc);
u_gpc = zeros(1, dlugosc);

if stan_ustalony
    y_gpc(1:D) = wart_zad;
end

s = zeros(D, 1);
for j = 1:D
    if j <= opoz
        s(j) = 0;
    else
        for i = 1:min(length(licz_Gz), j-opoz)
            s(j) = s(j) + licz_Gz(i);
        end
        for i = 1:min(length(mian_Gz) - 1, j-1)
            s(j) = s(j) - mian_Gz(i+1) * s(j-i);
        end
    end
end

M = zeros(param_gpc.N, Nu);
for i = 1:N
    for j = 1:min(i, Nu)
        M(i, j) = s(i-j+1);
    end
end

K = inv(M'*eye(N)*M + lambda*eye(Nu)) * M'*eye(N);

for k = max(3, D+1):dlugosc
    for i = 1:length(mian_Gz)-1
        y_gpc(k) = y_gpc(k) - mian_Gz(i+1) * y_gpc(k-i);
    end

    for i = 1:length(licz_Gz)
        idx = k - opoz - i + 1;
        y_gpc(k) = y_gpc(k) + licz_Gz(i) * u_gpc(idx);
    end

    y_gpc(k) = y_gpc(k) + zaklocenie;
```

```
    y_hat = 0;
    for j = 1:length(licz_Gz)
        idx = k - opoz - j + 1;
        if idx >= 1
            y_hat = y_hat + licz_Gz(j) * u_gpc(idx);
        end
    end
    for j = 1:length(mian_Gz)-1
        y_hat = y_hat - mian_Gz(j+1) * y_gpc(k-j);
    end

    d_k = y_gpc(k) - y_hat;

    y0 = zeros(N, 1);
    for p = 1:N
        y0(p) = 0;
        for i = 1:length(licz_Gz)
            if p+i <= D - 1
                idx = k - opoz - i + p;
                if idx >= k
                    idx = k-1;
                end
                y0(p) = y0(p) + licz_Gz(i) * u_gpc(idx);
            end
        end
        for i = 1:length(mian_Gz)-1
            if p+i <= D - 1
                idx = k - i + p;
                if p > i
                    y0(p) = y0(p) - mian_Gz(i+1) * y0(p-i);
                else
                    y0(p) = y0(p) - mian_Gz(i+1) * y_gpc(idx);
                end
            end
        end
        y0(p) = y0(p) + d_k;
    end

    dU = K * (yzad - y0);
    du_gpc = dU(1);

    u_gpc(k) = u_gpc(k-1) + du_gpc;
end
end
```

Listing 4.11. symulacja_gpc.m