Wydział Elektroniki i Technik Informacyjnych
Politechnika Warszawska

Projektowanie układów sterowania (projekt grupowy)

Sprawozdanie z projektu i ćwiczenia laboratoryjnego nr 3, zadanie nr 12

Paulina Dąbrowska, Miłosz Kowalewski,
Adam Rybojad, Mikołaj Wewiór

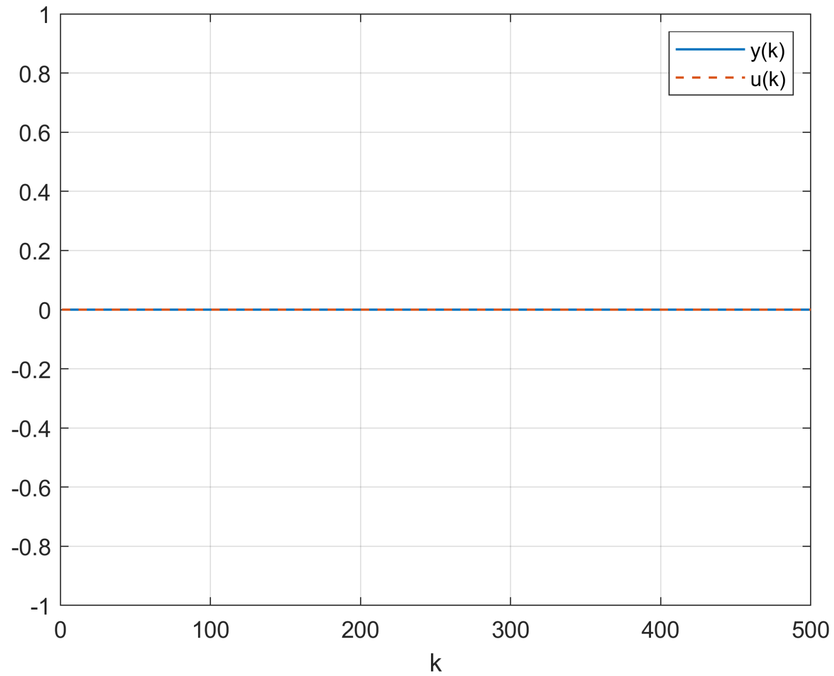
Warszawa, 2023

Spis treści

1. CZĘŚĆ PROJEKTOWA

1.1. Poprawność punktu pracy

Sprawdzenie poprawności podanego punktu pracy ($Upp = 0$, $Ypp = 0$) polegało na obserwacji wyjścia symulowanego obiektu projektowego `12y_p3` przy niezmiennym sterowaniu równym Upp.



Rys. 1. Sprawdzenie poprawności podanego punktu pracy.

Jak można zauważyć, na podstawie Rys. 1, przy niezmiennej wartości sterowania u równej Upp, wartość wyjścia obiektu y nie zmienia się, pozostając równą Ypp, co potwierdza tym samym, że punkt pracy jest stabilny i został poprawnie podany.

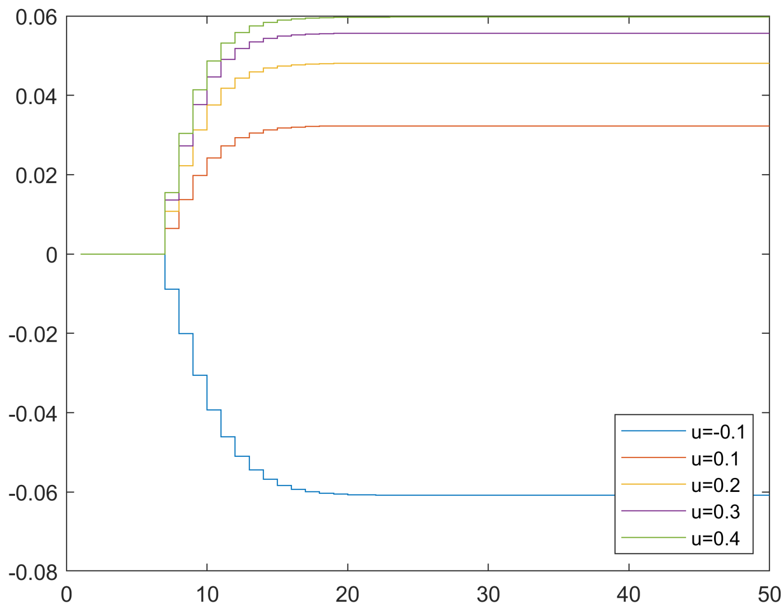
Implementacja w MATLABIE:

```
clear all

% parametry symulacji
kp = 7;
kk = 500;

% punkt pracy
Upp = 0;
Ypp = 0;
```

```
u(1:kp-1) = Upp;    u(kp:kk) = Upp;
y(1:kp-1) = Ypp;    y(kp:kk) = 0;

for k = kp : kk
    y(k) = symulacja_obiektu12y_p3(u(k-5), u(k-6), ...
        y(k-1), y(k-2));
end
```

1.2. Odpowiedzi skokowe

Odpowiedzi skokowe wykonane zostały dla kolejnych skoków sterowania tj. −0,1; 0,1; 0,2; 0,3; 0,4.



Rys. 2. Odpowiedzi skokowe dla różnych wartości skoku sterowania.

Wykonywane one były za pomocą kodu poniżej:

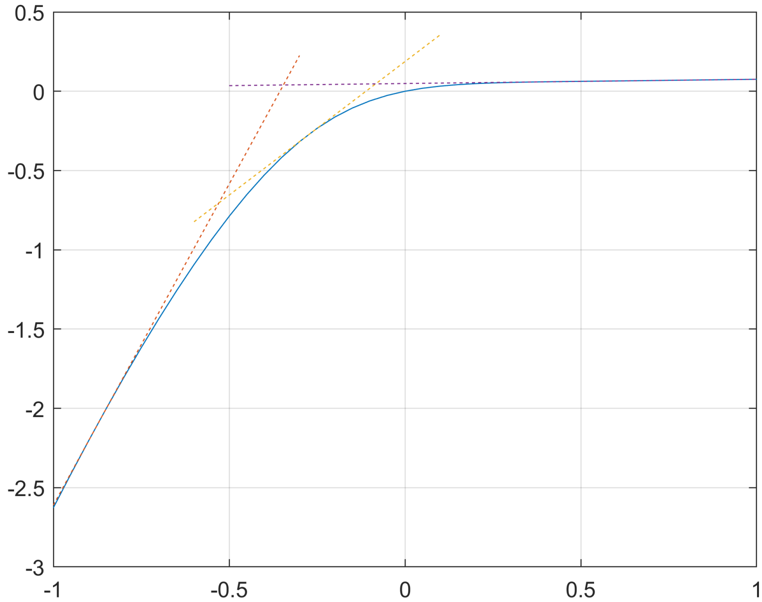
```
n = 6;
duration = 50;
u(1:n, 1:duration) = 0;
y(1:n, 1:duration) = 0;
Kstat(1:n) = 0;
start = 7;

% symulacja
for skok=1:n
    u(skok, 1:duration) = (skok-2)*0.1;
    if skok ~= 2
        for k=start:duration
            y(skok, k) = symulacja_obiektu12y_p3( ...
                                    u(skok, k-5), u(skok, k-6), ...
                                    y(skok, k-1), y(skok, k-2));
        end
        stairs(y(skok, :));
        hold on;
        Kstat(skok) = (y(skok, duration) - y(skok, 1)) /...
                                        u(skok, duration);
    end
end
```

1.2.1. Charakterystyka statyczna

Charakterystykę tą wyznaczaliśmy poniższym kodem. Dodatkowo dokonaliśmy linearyzacji w kliku punktach sterowania.

```
if strcmp(tryb, 'wzmocnienie')
    u(1:n) = 0;
    y(1:n) = 0;
    % wykres i wzmocnienie dla y(u):
    for temp_u = -1:0.05:1

        u(start:n) = temp_u;

        for k = start:n
            y(k) = symulacja_obiektu12y_p3(
                        u(k-5), u(k-6),y(k-1), y(k-2));
        end
        i = round(temp_u*20 + 21);
        Y(i) = y(n);
        U(i) = u(n);

        Ku(i) = (y(n)-y(1))/(u(n));
    end
    Ku(21) = 0;
    K1 = mean(Ku(1:18));
    K2 = mean(Ku(19:end));

    fku = figure;

    a1 = (-2.00291 - -2.20546) / 0.05;
    b1 = -2.20546 - a1*(-0.9);

    a2 =(-0.233832 - -0.31804) / 0.05;
    b2 = -0.233832 - a2*(-0.25);

    a3 = (0.0727849 - 0.0714511) / 0.05;
    b3 = 0.0727849 - a3*0.9;

    x1 = -1:0.01:-0.3;
    x2 = -0.6:0.01:0.1;
    x3 = -0.5:0.01:1;

    y_tan1 = a1*x1 + b1;
    y_tan2 = a2*x2 + b2;
    y_tan3 = a3*x3 + b3;
end
```

Sama charakterystyka wygląda w następujący sposób:



Rys. 3. Charakterystyka statyczna.

Od razu widać, że nie ma ona charakteru liniowego. Ma ona zdecydowane przegięcie w okolicach sterowania $u = -0{,}25$ zaś wartości wzmocnienia statycznego różnią się diametralnie na skrajnych wartościach sterowania. Od $u = -1$ do około $u = -0{,}5$ wzmocnienie ma wartość $K = -2{,}62$ a na końcu charakterystyki tj. od około $u = 0{,}1$ do $u = 1$ wynosi wynosi ono już 0,076. Ze względu na fakt, że w charakterystyka jest fragmentami liniowa, można dojść do wniosku, że regulator rozmyty nawet z niewielką ilością regulatorów lokalnych ma szansę działać poprawnie.

1.3. Implementacja algorytmu PID

Równanie różnicowe regulatora PID ma postać:

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

gdzie parametry r_0, r_1 oraz r_2 dane są poniższymi wzorami:

$$r_o = K_r(1 + \frac{T_p}{2T_i} + \frac{T_d}{T_p}), \;\; r_1 = K_r(\frac{T_p}{2T_i} - 2\frac{T_d}{T_p} - 1), \;\; r2 = \frac{K_r T_d}{T_p}$$

W badanym obiekcie występuje opóźnienie równe sześciu okresom próbkowania, a więc symulacje można rozpocząć od chwili $kp = 7$.

Implementacja w MATLABIE:

```
function E = PID(K_pid, Ti, Td)
    % punkt pracy
    Upp = 0;     Ypp = 0;

    % ograniczenia
    Umin = -1;   Umax = 1;

```

```
% parametry symulacji
kp = 7;          % chwila początkowa symulacji
kk = 500;        % chwila końcowa symulacji
T = 0.5;         % okres próbkowania

u(1:kp-1) = Upp;
y(1:kp-1) = Ypp;
e(1:kp-1) = 0;            e(kp:kk) = 0;
yzad(1:kp-1) = Ypp;       yzad(kp:kk) = 0.05;

% parametry r regulatora PID
r0 = K_pid*(1+T/(2*Ti)+Td/T);
r1 = K_pid*(T/(2*Ti)-2*Td/T-1);
r2 = K_pid*Td/T;

for k = kp : kk
    % symulacja wyjścia obiektu
    y(k) = symulacja_obiektu12y_p3(u(k-5), u(k-6), ...
        y(k-1), y(k-2));

    % uchyb regulacji
    e(k) = yzad(k)-y(k);

    % sygnał sterujący regulatora PID
    u(k) = r2*e(k-2) + r1*e(k-1) + r0*e(k) + u(k-1);

    % ograniczenia wartości sygnału sterującego
    if u(k) < Umin
        u(k) = Umin;
    elseif u(k) > Umax
        u(k) = Umax;
    end
end

% wskaźnik jakości regulacji E
E = 0;
for k = kp : kk
    E = E + (yzad(k)-y(k))^2;
end
```

1.4. Implementacja algorytmu DMC

W najprostszej (oszczędnej obliczeniowo) wersji algorytmu DMC przyrost sygnału sterującego w chwili k można zapisać jako:

$$\Delta u(k) = \Delta u(k|k) = K_e \cdot e(k) - \sum_{j=1}^{D-1} k_j^u \cdot \Delta u(k-j)$$

gdzie

$$K_e = \sum_{i=1}^{N} K_{1,i}$$

$$k_j^u = \bar{K}_1 \cdot M_j^P$$

dla $j = 1, ..., D$, przy czym: $\bar{K}_1$ - wiersz 1 macierzy K, M_j^P - kolumna j macierzy M^P.

Implementacja w MATLABIE:

```
function E = DMC(D, N, Nu, lambda)
    % punkt pracy
    ...

    % ograniczenia
    ...

    % parametry symulacji
    ...

    % odpowiedź skokowa
    s = odp_skokowa(Ypp, Upp, Uk);

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

    % macierz Mp
    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

    % obliczam K
    I = eye(Nu);
    K = ((M'*M + lambda*I)^(-1))*M';
```

```
% oszczędna wersja DMC - parametry
Ke = 0;
for i = 1 : N
    Ke = Ke + K(1, i);
end
ku = K(1, :) * Mp;

% symulacja
u(1:kp-1) = Upp;    u(kp:kk) = 0;
y(1:kp-1) = Ypp;    y(kp:kk) = 0;

for k = kp : kk
    % obiekt
    y(k) = symulacja_obiektu12y_p3(u(k-5), u(k-6), ...
        y(k-1), y(k-2));

    % oszczędna wersja DMC - algorytm
    e = yzad(k) - y(k);
    elem = 0;
    for j = 1 : D-1
        if k-j <= 1
            du = 0;
        else
            du = u(k-j) - u(k-j-1);
        end
        elem = elem + ku(j)*du;
    end

    % optymalny przyrost sygnału sterującego du(k|k)
    dukk = Ke * e - elem;

    % prawo regulacji
    u(k) = dukk + u(k-1);

    if u(k) > Umax
        u(k) = Umax;
    elseif u(k) < Umin
        u(k) = Umin;
    end
end

% wskaźnik jakości regulacji E
E = 0;
for k = kp : kk
    E = E + (yzad(k)-y(k))^2;
end
```

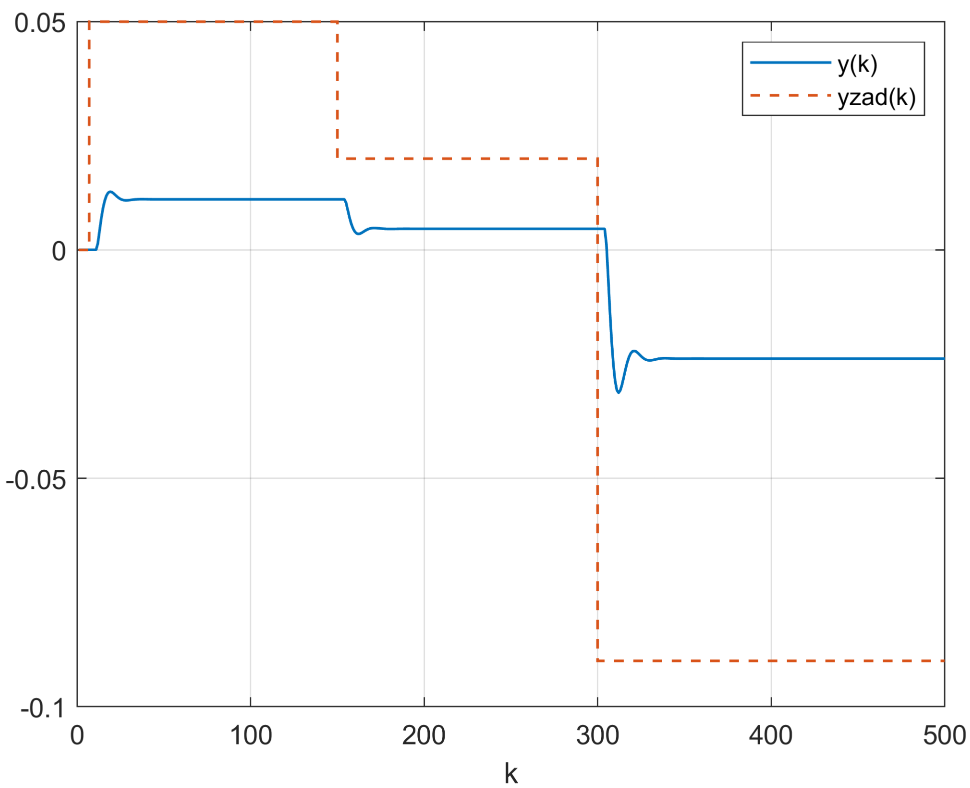
1.5. Dobór nastaw dla PID

Nastawy regulatora zostały dobrane metodą eksperymentalną, ponieważ opiera się ona bezpośrednio na obserwacjach działania modelu, a z tego względu, że możemy bezpośrednio oglądać sygnał wyjściowy obiektu w MATLABIE uznaliśmy ją za wygodniejszą. Co więcej, eksperymentalne podejście doboru nastaw umożliwia zbadanie wpływu różnych wartości parametrów na jakość regulacji i ich optymalne dopasowanie dla konkretnego problemu.

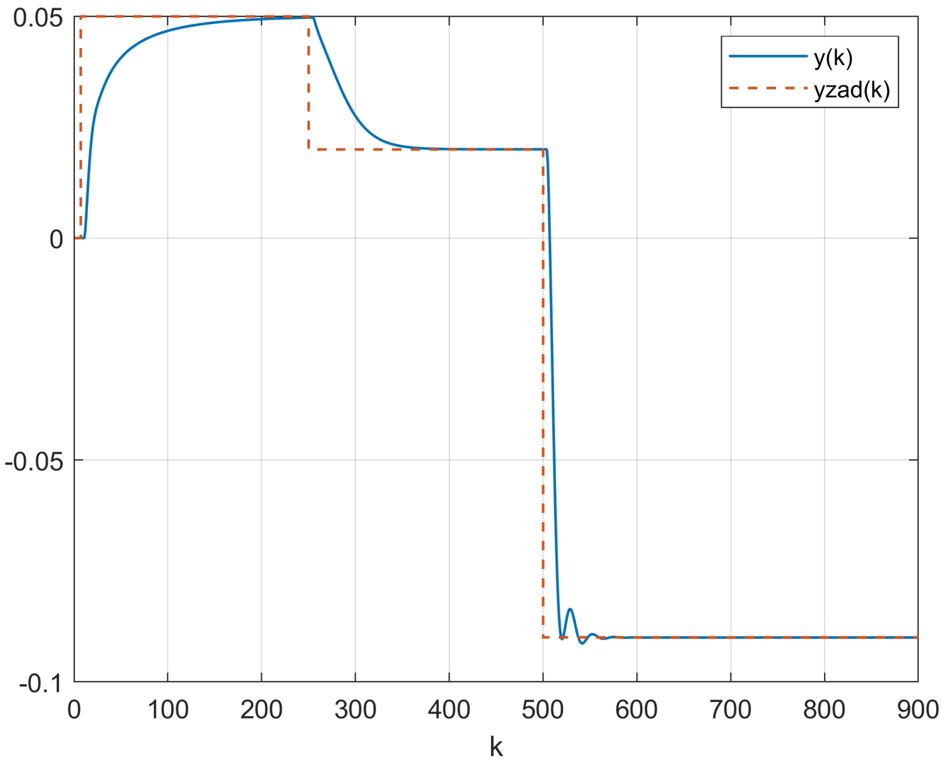
Kolejne kroki doboru nastaw algorytmu PID metodą eksperymentalną:
1. Analiza regulatora P - wyzerowanie członu całkującego (tzn. ustawienie parametru T_i na bardzo dużą wartość) i różniczkującego (tzn. ustawienie parametru T_d na zero). Zwiększanie parametru wzmocnienia K do momentu jak najmniejszego uchybu ustalonego i pojawienia się na wyjściu oscylacji.
2. Dodanie członu całkującego - stopniowe zwiększanie wpływu członu całkującego (zmniejszanie wartości T_i) i ewentualne zwiększenie wzmocnienia.
3. Dodanie członu różniczkującego - zwiększanie wartości parametru T_d. Gdy regulator będzie działał odpowiednio szybko można zwiększyć wpływ członu całkującego lub wzmocnienia.

Nastawy regulatora po takim dostrojeniu wyglądają następująco:
$K = 0{,}9$
$T_i = 2{,}7$
$T_d = 1{,}0$

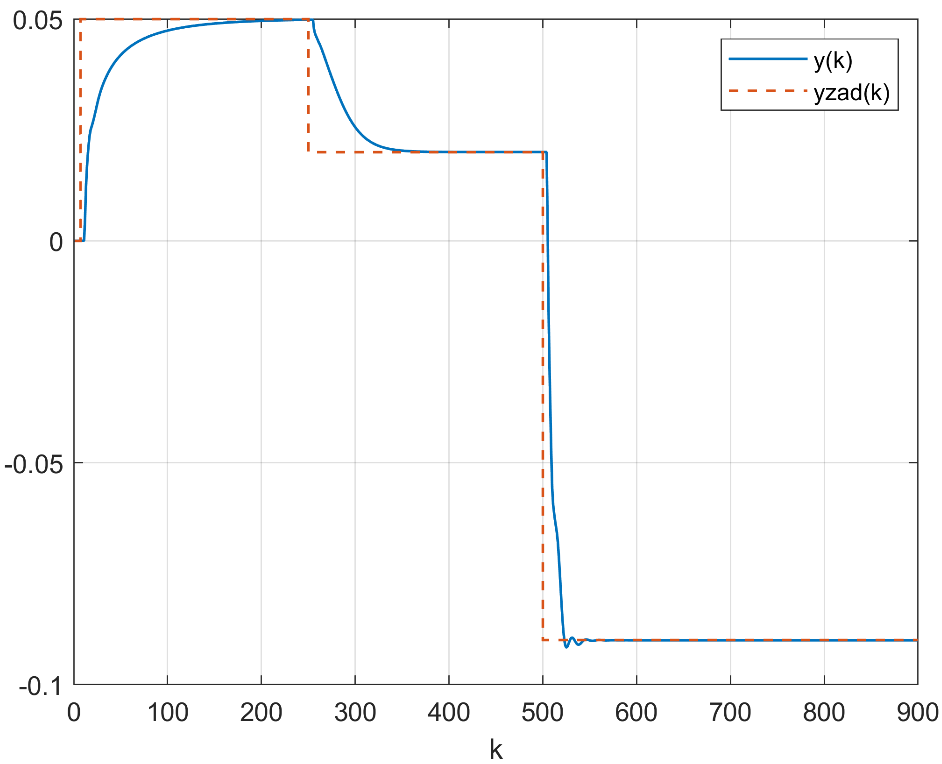
Wskaźnik jakości błędu dla powyższych nastawów: $E = 0{,}1377$



Rys. 4. Dobranie wzmocnienia K regulatorowi PID.



Rys. 5. Dobranie parametru T_i regulatorowi PID.

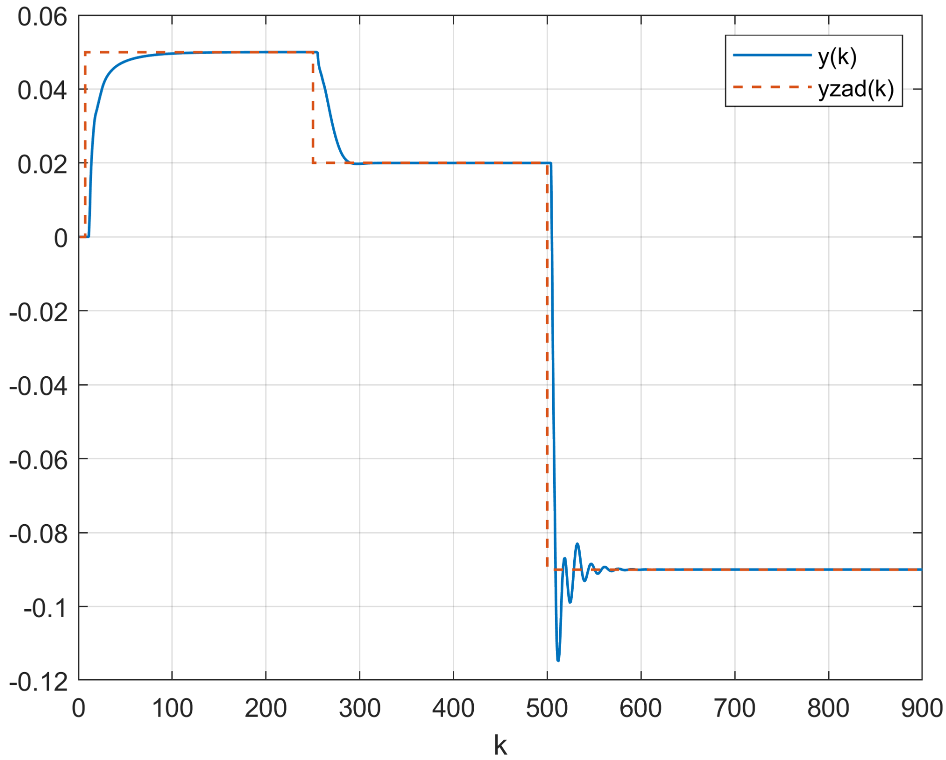


Rys. 6. Dobranie parametru T_d regulatorowi PID.

Po wstępnym dobraniu parametrów regulatora PID, zostały one jeszcze poddane pewnym modyfikacjom w celu minimalizacji wskaźnika jakości regulacji, których efekty można zaobserwować na Rys.7.

Nastawy regulatora po modyfikacji wyglądają następująco:
$K = 1{,}1$
$T_i = 1{,}8$
$T_d = 1{,}2$

Wskaźnik jakości błędu dla powyższych nastawów: $E = 0{,}1083$



Rys. 7. Przebieg po eksperymentalnym zmodyfikowaniu parametrów w celu minimalizacji wskaźnika jakości regulacji.

Jak można zauważyć regulator PID z mniejszą wartością błędu wskaźnika jakości regulacji działa szybciej, ale przy skoku na wartość zadaną 0,09 zwraca niechciane oscylacje. Z tego względu, jakościowo lepiej wypada regulator PID przed zmodyfikowaniem wartości nastaw - nie ma tutaj znacznego przeregulowania i oscylacji.

1.6. Dobór nastaw dla DMC

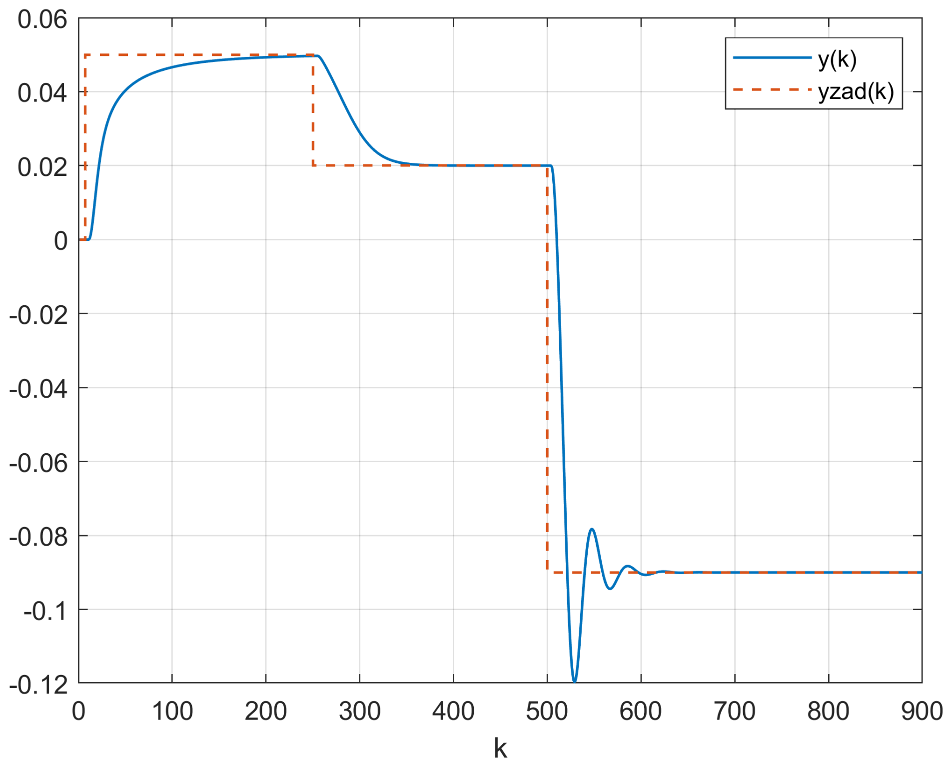
Analogicznie jak w przypadku doboru nastaw regulatora PID, skorzystaliśmy z metody eksperymentalnej.

Kolejne kroki doboru nastaw algorytmu DMC metodą eksperymentalną:
1. Ustalenie długości horyzontu dynamiki z odpowiedzi skokowej obiektu - moment gdy sygnał na wyjściu się nie zmienia lub zmienia się nieznacznie. W przypadku naszego obiektu przyjęliśmy $D = 40$

2. Ustawienie wartości horyzontu predykcji i horyzontu sterowania równych horyzontowi dynamiki. Ustawienie parametru lambda na 1.
3. Stopniowe zmniejszanie wartości horyzontu predykcji i horyzontu sterowania.
4. Zmniejszenie wartości horyzontu sterowania.
5. Zmiana wartości lambdy - zmniejszenie, aby zmniejszyć przeregulowanie dla sygnału sterującego, zwiększenie aby przyspieszyć działanie regulatora. Warto rozważyć zwiększenie horyzontu predykcji.

W wyniku przeprowadzonych eksperymentów dobrano następujące nastawy regulatora:
$D = 40$
$N = 8$
$Nu = 2$
$lambda = 1$

Wskaźnik jakości błędu dla powyższych nastawów: $E = 0{,}2272$



Rys. 8. Wyjście obiektu przy nastawach dobranych metodą eksperymentalną.

Przy dalszej modyfikacji parametrów regulatora, regulacja nie ulegała już poprawie, a wskaźnik jakości regulacji rósł, dlatego dobrane parametry zostały uznane za optymalne. Na podstawie Rys.8 można zauważyć, że regulator działa wolniej niż w przypadku wcześniej badanego regulatora PID.

1.7. Rozmyty algorytm PID

Rozmyty regulator PID można zapisać za pomocą pewnej bazy reguł:
Reguła 1: jeżeli $u(k-1) \in Z^1$

$$u^1(k) = u(k-1) + r_1^1 \cdot e(k) + r_2^1 \cdot e(k-1) + r_3^1 \cdot e(k-2)$$

$$\vdots$$

Reguła r: jeżeli $u(k-1) \in Z^r$

$$u^r(k) = u(k-1) + r_1^r \cdot e(k) + r_2^r \cdot e(k-1) + r_3^r \cdot e(k-2)$$

gdzie $Z^{1,\ldots,r}$ to kolejne zbiory rozmyte.

Sygnał sterujący regulatora rozmytego wyznacza się z następujacej zależności:

$$u(k) = \frac{\sum_{i=1}^{r} u^i(k) \cdot w_i}{\sum_{i=1}^{r} w_i}$$

Implementacja w MATLABIE:

```
function E = PID_rozmyty(K_pid, Ti, Td)
    % ***************inicjalizacja***************
    % liczba reguł
    ...

    % punkt pracy
    ...

    % ograniczenia
    ...

    % parametry symulacji
    ...
    % ***********************************************

    %% parametry r regulatora PID
    for r = 1 : lRegul
        r0(r) = K_pid(r)*(1+T/(2*Ti(r))+Td(r)/T);
        r1(r) = K_pid(r)*(T/(2*Ti(r))-2*Td(r)/T-1);
        r2(r) = K_pid(r)*Td(r)/T;
    end

    %% symulacja
    for k = kp : n
        % wagi
        w = wagi(uResult(k-1));

        % symulacja obiektu
        y(k) = symulacja_obiektu12y_p3(uResult(k-5), ...
            uResult(k-6), y(k-1), y(k-2));
```

```
        % uchyb regulacji
        e(k) = yzad(k)-y(k);

        for r = 1 : lRegul
            % sygnał sterujący regulatora PID
            u{r}(k) = r2(r)*e(k-2) + r1(r)*e(k-1) + ...
                r0(r)*e(k) + uResult(k-1);
            uResult(k) = uResult(k) + w(r)*u{r}(k);
        end
        uResult(k) = uResult(k)/sum(w);

        % ograniczenia wartości sygnału sterującego
        if uResult(k) < Umin
            uResult(k) = Umin;
        elseif uResult(k) > Umax
            uResult(k) = Umax;
        end
    end

    %% wskaźnik jakości regulacji E
    E = 0;
    for k = 1 : n
        E = E + (yzad(k)-y(k))^2;
    end
```

1.8. Rozmyty algorytm DMC

Rozmty regulator DMC w postaci oszczędnej można zapisać za pomocą pewnej bazy reguł:
Reguła 1: jeżeli $u(k-1) \in Z^1$

$$u^1(k) = k_e^1 \cdot e(k) - \sum_{i=1}^{D} k_i^{u^1} \cdot \Delta u(k-i) + u(k-1)$$

$$\vdots$$

Reguła r: jeżeli $u(k-1) \in Z^r$

$$u^r(k) = k_e^r \cdot e(k) - \sum_{i=1}^{D} k_i^{u^r} \cdot \Delta u(k-i) + u(k-1)$$

gdzie $Z^{1,\dots,r}$ to kolejne zbiory rozmyte.

Przy czym k_e^l oraz $k_i^{u^l}$, gdzie $l = 1, ..., r$ obliczamy korzystając z poniższych zależności:

$$k_e^1 = \sum_{p=1}^{N} k_{1,p}^1$$

$$\vdots$$

$$k_e^r = \sum_{p=1}^{N} k_{1,p}^r$$

$$k_i^{u^1} = \bar{K_1^1} \cdot M_i^{P^1}$$

$$\vdots$$

$$k_i^{u^r} = \bar{K_1^r} \cdot M_i^{P^r}$$

Implementacja w MATLABIE:

```
function E = DMC_rozmyty(D, N, Nu, lambda)
    % ***************inicjalizacja***************
    % liczba reguł
    ...

    % punkty pracy
    ...

    % ograniczenia
    ...

    % parametry symulacji
    ...
    % *********************************************

    %% skok
    for r = 1:lRegul
        s{r} = odp_skokowa(Ypp(r), Upp(r), Upp(r)+0.01);
    end

    %% macierz M
    for r = 1:lRegul
        m = zeros(N, Nu);
        for i = 1 : Nu
            m(i : N, i) = s{r}(1 : (N - i + 1))';
        end
        M{r} = m;
    end

    %% macierz Mp
    for r = 1:lRegul
        mp = zeros(N, D-1);
        for i = 1 : N
           for j = 1 : D-1
              if i+j <= D
                 mp(i, j) = s{r}(i+j) - s{r}(j);
              else
                 mp(i, j) = s{r}(D) - s{r}(j);
              end
           end
```

```
        end
        Mp{r} = mp;
    end

    %% obliczam K
    I = eye(Nu);
    for r = 1:lRegul
        K{r} = ((M{r}'*M{r} + lambda*I)^(-1))*M{r}';
    end

    %% oszczędna wersja DMC - parametry
    for r = 1:lRegul
        Ke{r} = sum(K{r}(1,:));
        ku{r} = K{r}(1, :)*Mp{r};
    end

    %% symulacja
    for k = kp:kk
        % wagi
        w = wagi(uResult(k-1));

        % obiekt
        y(k) = symulacja_obiektu12y_p3(uResult(k-5), ...
            uResult(k-6), y(k-1), y(k-2));

        % uchyb
        e = yzad(k) - y(k);

        for r = 1:lRegul
            elem = 0;
            for j = 1 : D-1
                if k-j <= 1
                    du = 0;
                else
                    du = u{r}(k-j) - u{r}(k-j-1);
                end
                elem = elem + ku{r}(j)*du;
            end

            % optymalny przyrost sygnału sterującego du(k|k)
            dukk = Ke{r} * e - elem;

            % prawo regulacji
            u{r}(k) = dukk + u{r}(k-1);

            uResult(k) = uResult(k) + w(r)*u{r}(k);
        end
        uResult(k) = uResult(k) / sum(w);

        % ograniczenia sterowania
        if uResult(k) > Umax
```

```
            uResult(k) = Umax;
        elseif uResult(k) < Umin
            uResult(k) = Umin;
        end
    end
```

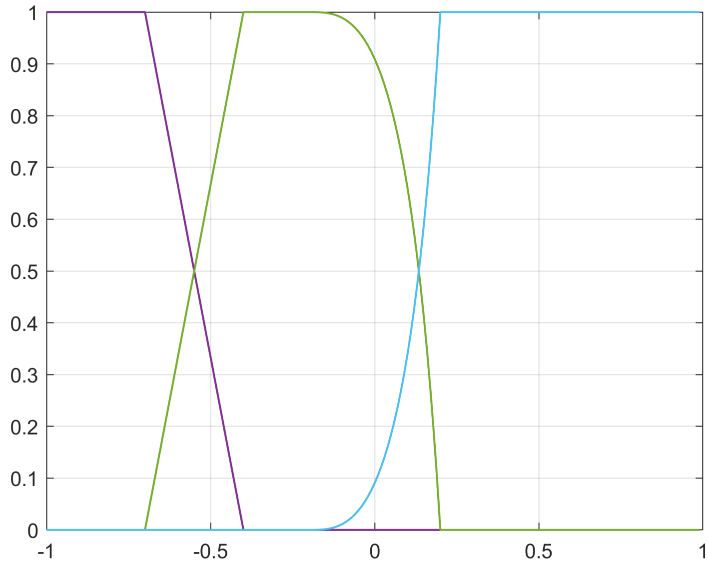
```
function s = odp_skokowa(Ypp, Upp, Uk)

% inicjalizacja parametrów symulacji
kp = 7;
kk = 800;

% sterowanie
y(1:kp-1) = Ypp;
u(1:kp-1) = Upp;
u(kp:kk+kp) = Uk;

% model
for k = kp:kk+kp
    y(k) = symulacja_obiektu12y_p3(u(k-5), u(k-6), y(k-1), y(k-2));
end
s(1:kk) = (y(kp:kk+kp-1)-ones(1, kk)*Ypp)/(Uk-Upp);
```

W celu wyznaczenia odpowiedzi skokowej dla każdej reguły, zostały odczytane z charakterystyki statycznej kolejne punkty pracy i odpowiednio podane jako argumenty wyżej zaimplementowanej funkcji `odp_skokowa`.

1.9. Rozmywanie i funkcja przynależności

Jako zmienną na podstawie, której dokonywane jest rozmywanie przyjęliśmy wartość sterowania z chwili wcześniejszej, czyli $u(k-1)$. Zastosowaliśmy historię sterowania, ponieważ jest ona wykorzystywana do obliczenia bieżącej wartości sygnału sterowania u(k). Z tego względu, rozmywanie $u(k-1)$ może poprawić skuteczność regulacji.

Jako funkcję przynależności wybraliśmy funkcję, która jest w połowie trapezowa, a w połowie dzwonowa. Zdecydowaliśmy się na takie kształty po jakościowej analizie charakterystyki statycznej. Po sprawdzeniu nastaw dla lokalnych regulatorów PID w obszarze ich działania, widzimy diametralne różnice dla dwóch pierwszych obszaró i trzeciego. Dlatego między regulatorem nr 1 i 2 jest funkcja trapezowa, a między regulatorem nr 2 i 3 jest już dzwonowa. Gdy Używaliśmy trapezowej funkcji przynależności, to regulator przechodząc między obszarami nie działał poprawnie.



Rys. 9. Funkcja przynależności w przypadku 3 regulatorów lokalnych.

1.10. Eksperymenty z różną liczbą regulatorów lokalnych

W przypadku 3 regulatorów lokalnych funkcja przynależności wygląda tak, jak na Rys.9. Na jej podstawie obliczane są poziomy aktywacji poszczególnych reguł.

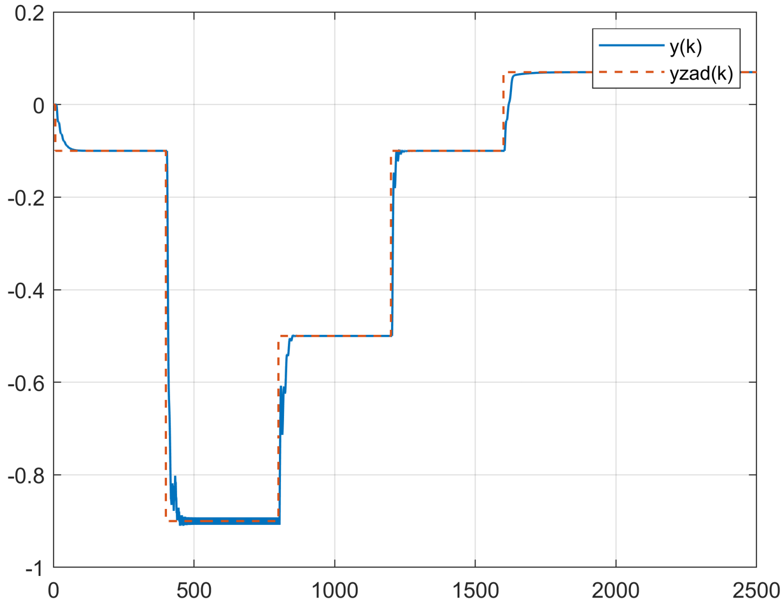
Implementacja w MATLABIE:

```
function w = wagi(u)

    if u <= -0.7
        w1 = 1;
        w2 = 0;
        w3 = 0;

    elseif u > -0.7 && u < -0.4
        a1 = 10/3;  b11 = -4/3; b12 = 7/3;
        w1 = -a1*u + b11;
        w2 = a1*u + b12;
        w3 = 0;

    elseif u >= -0.4 && u <= -0.2
        w1 = 0;
        w2 = 1;
        w3 = 0;

    elseif u > -0.35 && u < 0.2
        X = 10/4 * (u + 0.2);
        bell = 2.^(X.^7) - 1;

        w1 = 0;
        w2 = -bell +1;
        w3 = bell;

    elseif u >= 0.2
        w1 = 0;
        w2 = 0;
        w3 = 1;
    end
    w = [w1, w2, w3];
```

Dla tak ustalonych wag, regulacja dla założonej trajektorii zmian sterowania została przedstawiona poniżej.

Rozmyty regulator DMC:
Parametry regulatora DMC wyznaczone metodą eksperymentalną:
$D = 150$
$N = 39$
$Nu = 9$
$lambda = 1$

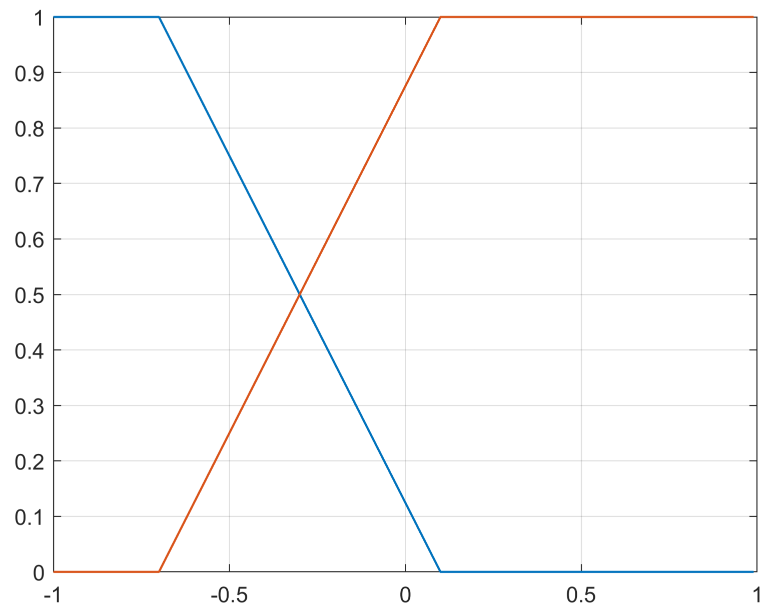
Wskaźnik jakości błędu dla powyższych nastawów: $E = 8{,}1443$.



Rys. 10. Rozmyty regulator DMC, z trzema regulatorami lokalnymi.

Rozmyty regulator PID:

W przypadku 2 regulatorów lokalnych funkcja przynależności wygląda tak, jak na Rys.**??**. Na jej podstawie obliczane są poziomy aktywacji poszczególnych reguł.



Rys. 11. Funkcja przynależności w przypadku dwóch regulatorów lokalnych.

Implementacja w MATLABIE:

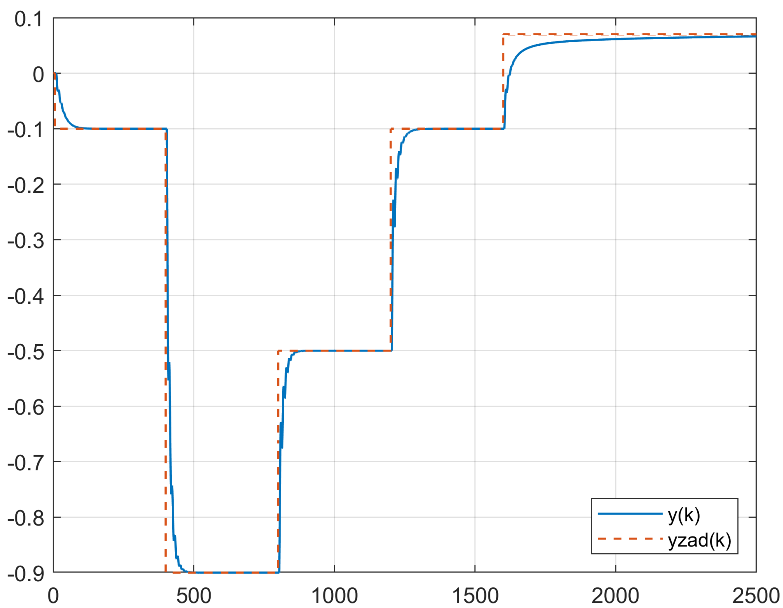
```
function w = wagi2(u)
```

```
    if u <= -0.7
        w1(k) = 1;
        w2(k) = 0;

    elseif(u > -0.7 && u < 0.1)
        a1 = 10/8;  b11 = 0.125; b12 = 0.875;
        w1(k) = -a1*u + b11;
        w2(k) = a1*u + b12;

    elseif (u >= 0.1 && u <= 1)
        w1(k) = 0;
        w2(k) = 1;
    end

w = [w1, w2];
```

Dla tak ustalonych wag, regulacja dla założonej trajektorii zmian sterowania została przedstawiona poniżej.

Rozmyty regulator DMC:
Parametry regulatora DMC wyznaczone metodą eksperymentalną:
$D = 150$
$N = 39$
$Nu = 9$
$lambda = 1$

Wskaźnik jakości błędu dla powyższych nastaw: $E = 9{,}2423$.



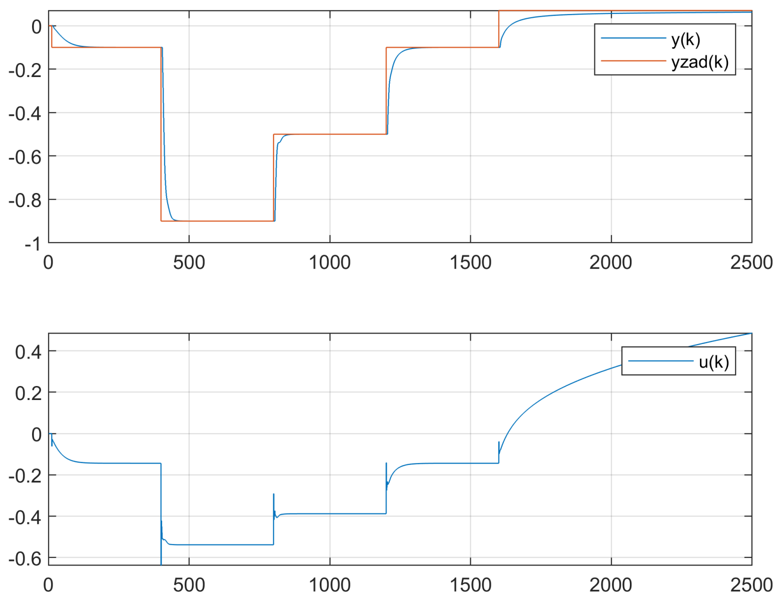
Rys. 12. Rozmyty regulator DMC, z dwoma regulatorami lokalnymi.

Rozmyty regulator PID:

Parametry regulatora PID z dwoma regulatorami lokalnymi wyznaczone metodą eksperymentalną:

Pierwszy regulator:
$K = 0{,}3$
$T_i = 4$
$T_d = 0{,}5$
Drugi regulator:
$K = 0{,}2$
$T_i = 3$
$T_d = 1$

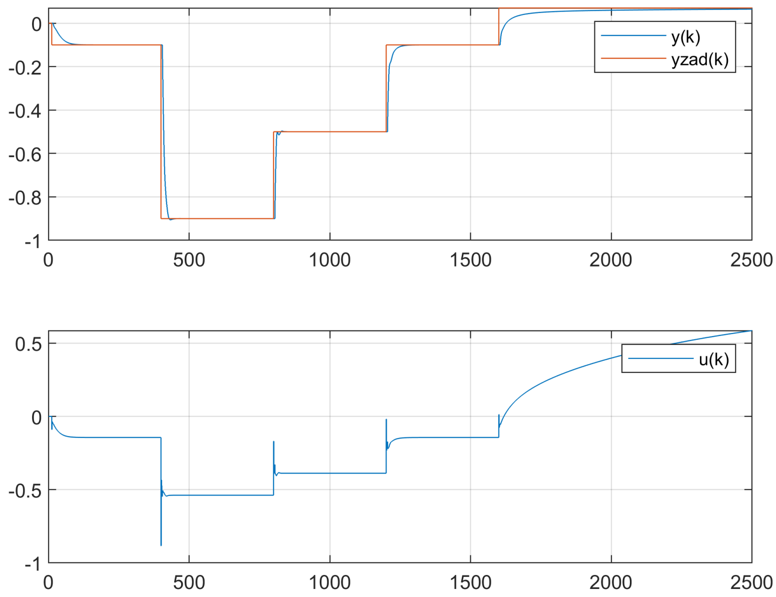
Wskaźnik jakości błędu dla powyższych nastaw: $E = 9{,}6171$.



Rys. 13. Rozmyty regulator PID, z dwoma regulatorami lokalnymi.

Parametry regulatora PID wyznaczone metodą eksperymentalną:
Pierwszy regulator:
$K = 0{,}15$
$T_i = 3{,}5$
$T_d = 0{,}8$
Drugi regulator:
$K = 0{,}4$
$T_i = 3$
$T_d = 0{,}7$
Trzeci regulator:
$K = 14$
$T_i = 3$
$T_d = 0{,}3$

Wskaźnik jakości błędu dla powyższych nastaw: $E = 8{,}2092$.



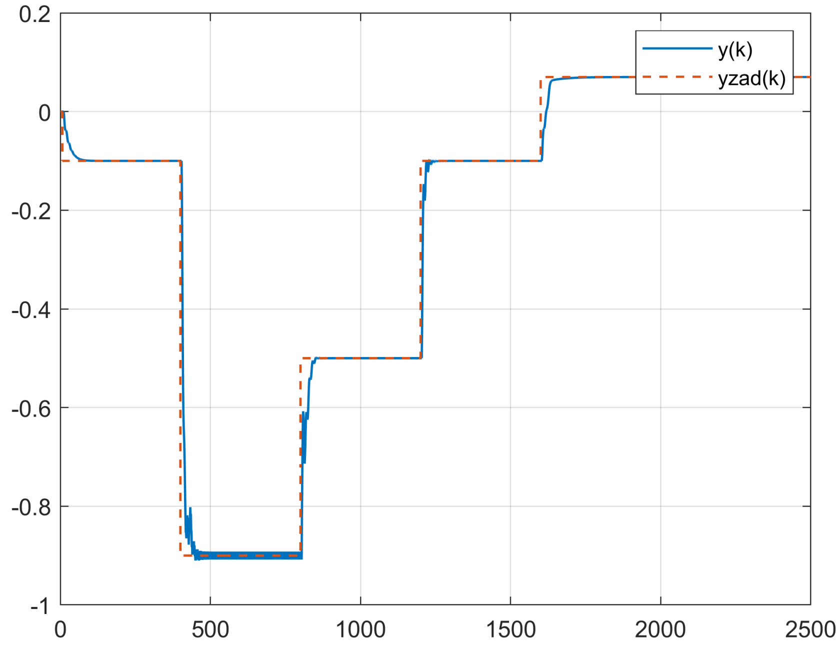
Rys. 14. Rozmyty regulator PID, z trzema regulatorami lokalnymi.

Na podstawie ilościowego wskaźnika jakości a także oceny jakościowej wykresów można stwierdzić, że regulacja działa zauważalnie lepiej, choć nie jest to jakaś ogromna różnica. Regulator rozmyty z dwoma lokalnymi jest już całkiem szybszy. Dodatkowym problemem było poprawne strojenie, które było w istocie nietrywalne. Z tych powodów uznaliśmy, że nie będzimy testowali reuglatora rozmytego z większą ilością regulatorów loklanych niż 3.

1.11. Parametry λ regulatorów lokalnych DMC

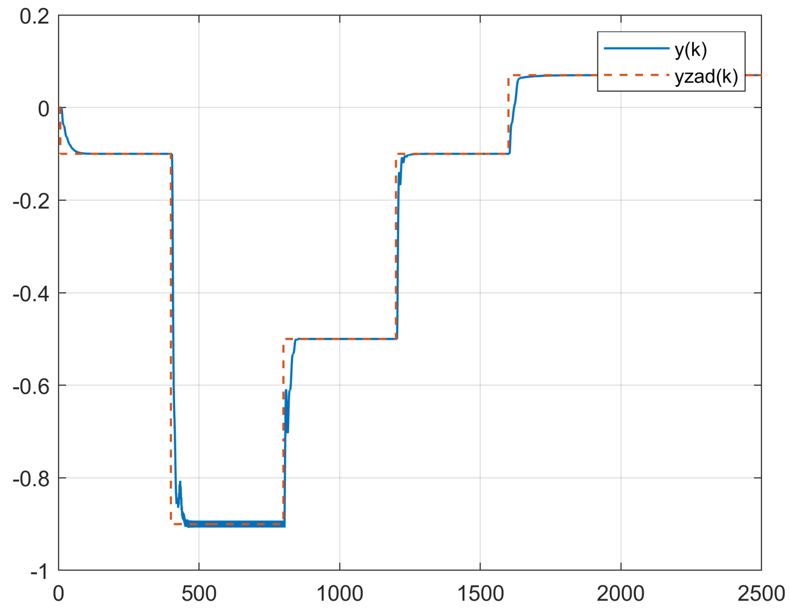
Dla 3 regulatorów lokalnych:

$\lambda_1 = \lambda_2 = \lambda_3 = 1$, $E = 9{,}2423$.



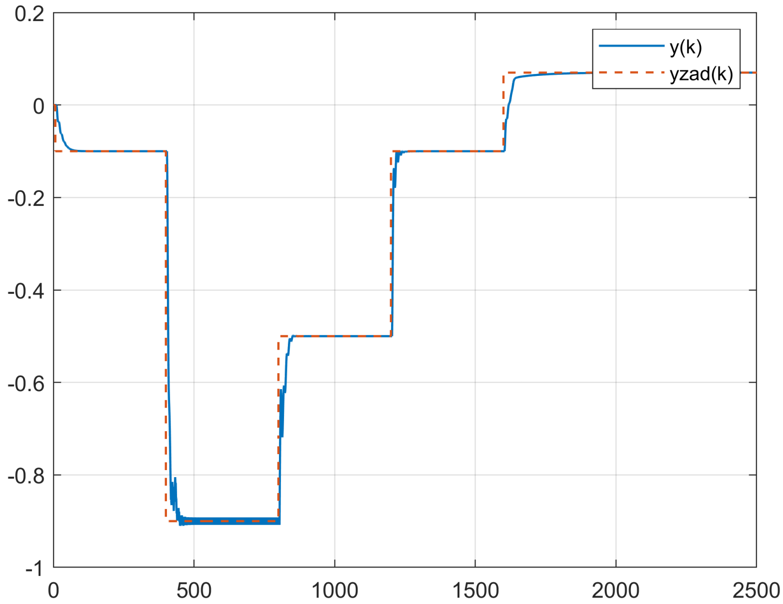
Rys. 15. Regulator rozmyty DMC z trzema regulatorami lokalnymi, gdzie $\lambda_1 = \lambda_2 = \lambda_3 = 1$.

$\lambda_1 = 1$, $\lambda_2 = 2$, $\lambda_3 = 1$, $E = 8{,}3529$.



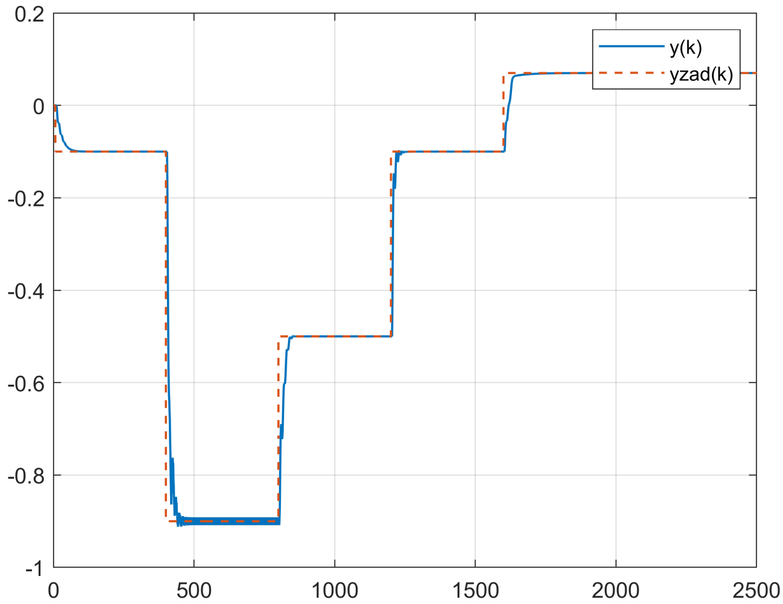
Rys. 16. Regulator rozmyty DMC z trzema regulatorami lokalnymi, gdzie $\lambda_1 = 1$, $\lambda_2 = 2$, $\lambda_3 = 1$.

$\lambda_1 = 1$, $\lambda_2 = 1$, $\lambda_3 = 2$, $E = 8{,}1515$.



Rys. 17. Regulator rozmyty DMC z trzema regulatorami lokalnymi, gdzie $\lambda_1 = 1$, $\lambda_2 = 2$, $\lambda_3 = 2$.

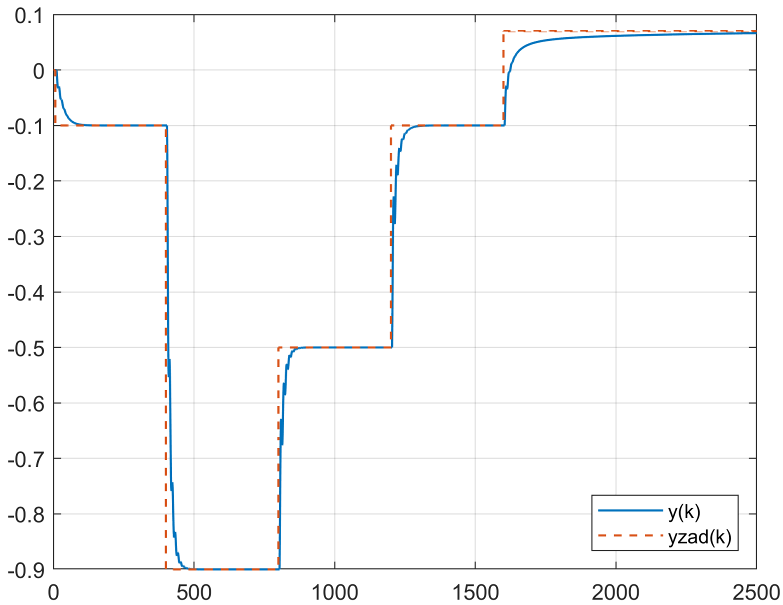
$\lambda_1 = 2$, $\lambda_2 = 1$, $\lambda_3 = 2$, $E = 8{,}4649$.



Rys. 18. Regulator rozmyty DMC z trzema regulatorami lokalnymi, gdzie $\lambda_1 = 2$, $\lambda_2 = 1$, $\lambda_3 = 1$.

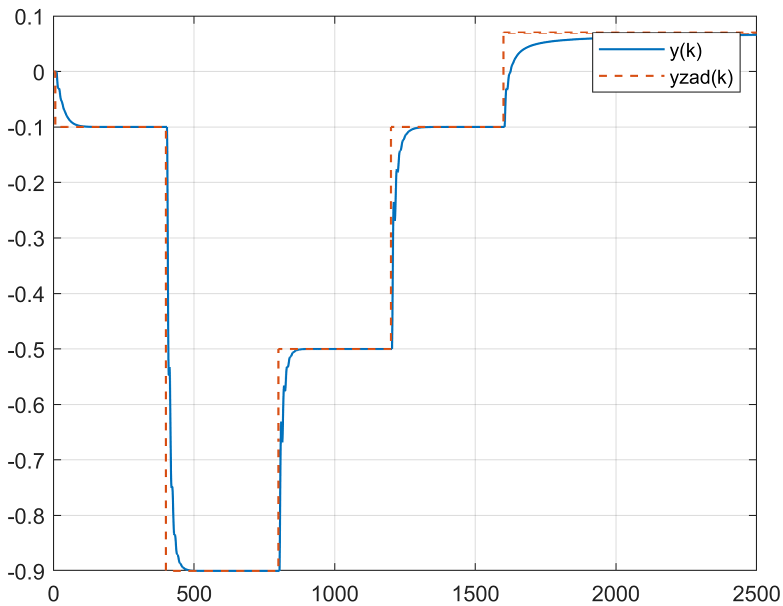
Jak można zauważyć na podstawie wykresów, jak i na podstawie wartości wskaźnika jakości regulacji optymalnymi wartościami parametru λ dla kolejnych regulatorów lokalnych są $\lambda_1 = 1$, $\lambda_2 = 1$,$\lambda_3 = 2$. Dla takich wartości wskaźnik jakości regulacji jest najmniejszy. Dalsze zwiększanie wartości λ_3 powoduje zwiększenie wartości wskaźnika jakości regulacji, dlatego kolejne wykresy zostały pominięte.

Dla 2 regulatorów lokalnych:
$\lambda_1 = 1$, $\lambda_2 = 1$, $E = 9{,}2423$.



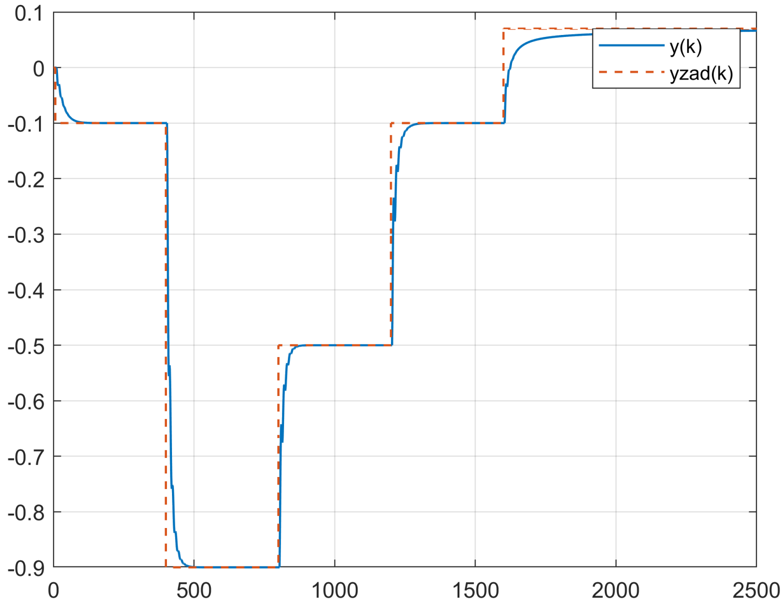
Rys. 19. Regulator rozmyty DMC z dwoma regulatorami lokalnymi, gdzie $\lambda_1 = 1$, $\lambda_2 = 1$.

$\lambda_1 = 1$, $\lambda_2 = 2$, $E = 9{,}4017$.



Rys. 20. Regulator rozmyty DMC z dwoma regulatorami lokalnymi, gdzie $\lambda_1 = 1$, $\lambda_2 = 2$.

$\lambda_1 = 2$, $\lambda_2 = 1$, $E = 9{,}4851$.



Rys. 21. Regulator rozmyty DMC z dwoma regulatorami lokalnymi, gdzie $\lambda_1 = 2$, $\lambda_2 = 1$.

Jak można zauważyć na podstawie wykresów, jak i na podstawie wartości wskaźnika jakości regulacji optymalnymi wartościami parametru λ dla kolejnych regulatorów lokalnych są $\lambda_1 = 1$, $\lambda_2 = 1$. Dla takich wartości wskaźnik jakości regulacji jest najmniejszy.

2. CZĘŚĆ LABORATORYJNA

2.1. Punkt pracy obiektu

Sterowanie stanowiska laboratoryjnego odbywało się za pomocą MATLABA funkcją `obiekt12(u)`, która jest edycją otrzymanego przez nas kodu funkcji `MinimalWorkingExample()`.

Sygnał sterujący wentylatorem W1 ustawiano jako argument funkcji `sendControls()`, której pierwszym argumentem była tablica z ID elementów do ustawienia, a drugim tablica z wartościami sygnałów. ID wentylatora W1 to 1. Do sterowania grzałką i zakłóceniami użyto `sendNonlinearControls(u);`, gdzie `u` przekazywane jest jako argument głównej funkcji `obiekt12()`.

```
 function y = obiekt12(u)
    addpath('D:\SerialCommunication');
    initSerialControl COM7 % initialise com port
    waitForNewIteration();
     sendControls(1, 50);

    sendNonlinearControls(u);
    measurement = readMeasurements(1:1);
    y = measurement
end
```

Punkt pracy grzałki w ciągu zajęć laboratoryjnych bardzo znacznie się wahał. Począwszy od wartości 30 °C kończąc na 35 °C. Niemniej jednak większość pomiarów została przeprowadzona dla ustalonego punktu pracy:

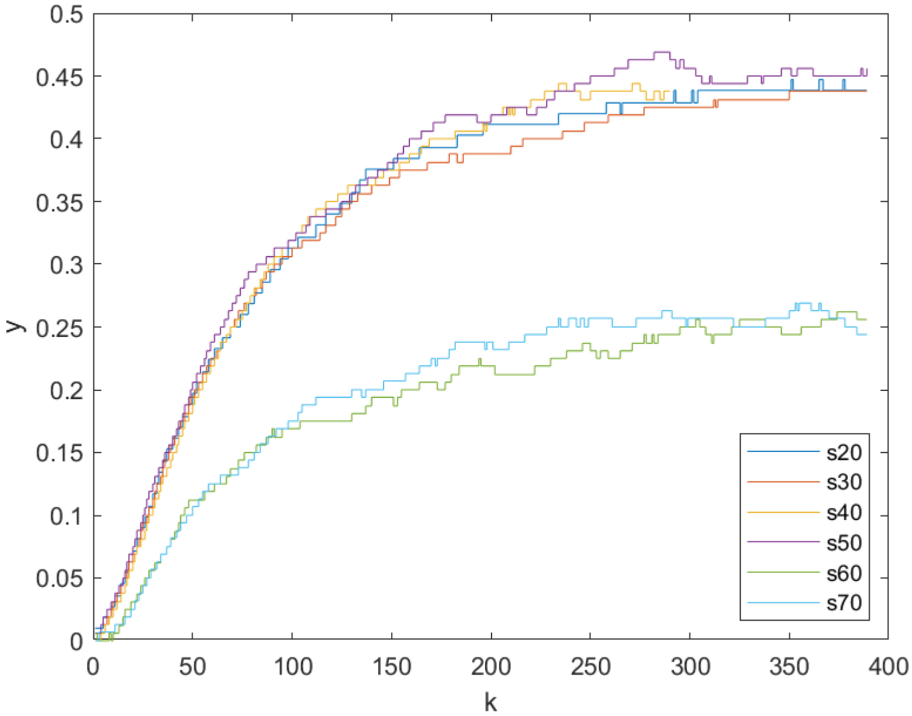
$$G1 = 27, Y = 34{,}3°C$$

2.2. Określenie wzmocnienia w funkcji sterowania

W celu określenia wzmocnienia należało pozyskać odpowiedzi skokowe dla 7 różnych wartości sterowania:

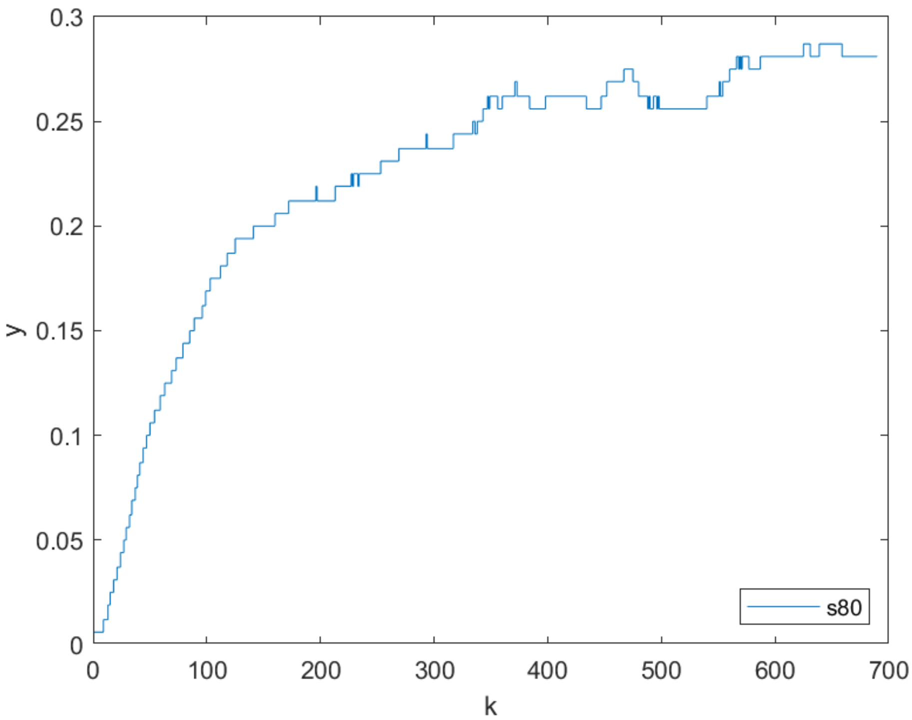
$$u = 20, u = 30, u = 40, u = 50, u = 60, u = 70, u = 80$$

Na rysunku 4. pokazano odpowiedzi skokowe dla pierwszych 6 pomiarów, które zostały już przeliczone pod wartości s używane w regulatorze DMC.



Rys. 22. Odpowiedź skokowa s obiektu dla sterowań $u = 20, 30, 40, 50, 60, 70$

Jak widać, sterowania dla pierwszych 4. wartości sterowań, wzmocnienie jest większe niż dla kolejnych dwóch. Ponieważ dla $u = 80$ odpowiedź obiektu ustalała się o wiele dłużej, została ona pokazana na oddzielnym rysunku:



Rys. 23. Odpowiedź skokowa s obiektu dla sterowania $u = 80$

Z przedstawionych pomiarów można wyznaczyć wzmocnienie statyczne K:

$$K = \frac{y_{ust}}{\Delta u}$$

gdzie Δu to skok sterowania w chwili rozpoczęcia pomiaru. Aby nie tracić czasu na laboratorium pomiary były robione ze skokiem sterowania równym 10.
Dla:

$$u = 20, u = 30, u = 40$$
$$K \approx 0{,}444$$

Dla:

$$u = 50$$
$$K \approx 0{,}456$$

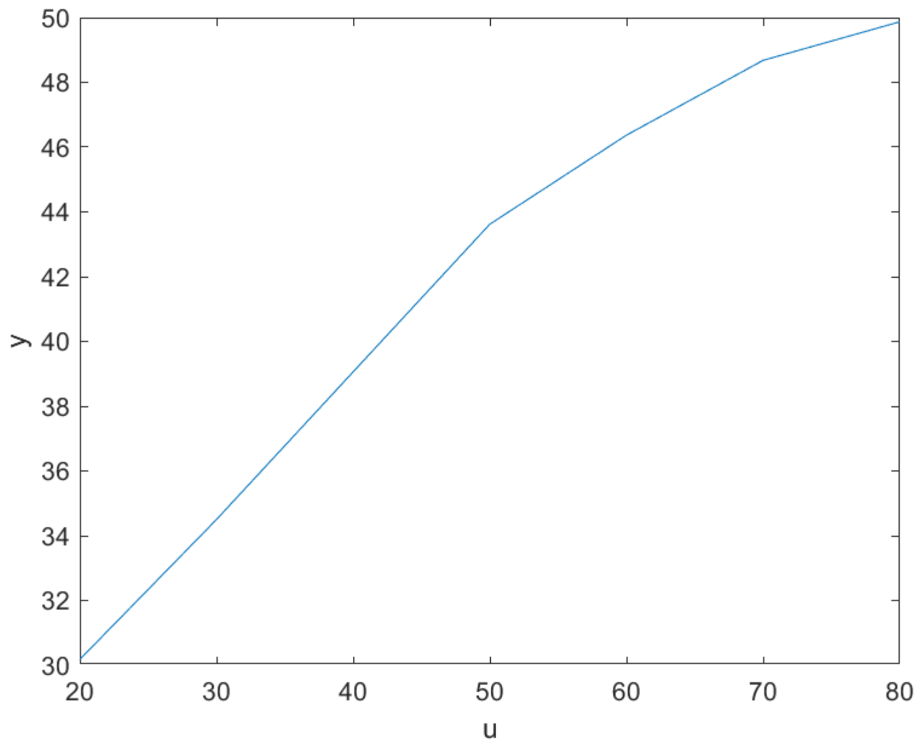
Dla:

$$u = 60, u = 70$$
$$K \approx 0{,}262$$

Dla:

$$u = 80$$
$$K \approx 0{,}281$$

2.2.1. Charakterystyka statyczna



Rys. 24. Charakterystyka statyczna obiektu y(u)

Od razu widać, że charakterystyka ta nie jest liniowa w całym swoim zakresie a jedynie w przybliżeniu odcinkami liniowa. Widać, że dla sterowania równego

$$u = 50, u = 70$$

występują punkty przegięcia. Sugeruje to granice stosowalności poszczególnych regulatorów lokalnych.

2.3. Regulacja PID dla obiektu liniowego

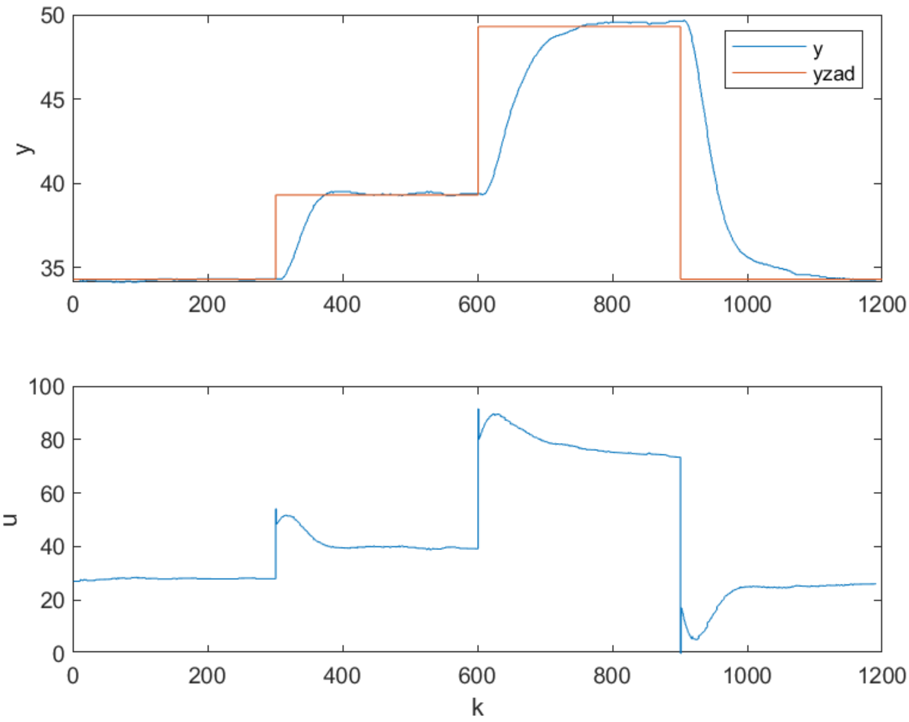
Regulator PID z bloku 1. został zaimplementowany w poniższy sposób:

```
% parametry:
   K_pid = 4;
   Ti = 60;
   Td = 0.3;
   k = 10;

   while(1)  % na rzeczywistym obiekcie
       k = k + 1;
       y(k) = obiekt12(u(k-1)); % rzeczywisty obiekt

       if y(k) > Ymax
           y(k) = Ymax;
       elseif y(k) < Ymin
           y(k) = Ymin;
       end

       u(k) = regulacjaPID(T, k, u, y, yzad, K_pid, Ti, Td);
       if u(k) > Umax
           u(k) = Umax;
       elseif u(k) < Umin
           u(k) = Umin;
       end
```

Gdzie funkcja *regulacjaPID()* wygląda:

```
function uk = regulacjaPID(T, k, u, ypom, yzad, K_pid, Ti, Td)
    r0 = K_pid * (1 + T/(2*Ti) + Td/T);
    r1 = K_pid * (T/(2*Ti) - 2*Td/T - 1);
    r2 = K_pid * Td/T;

    e(k) = yzad(k) - ypom(k);
    e(k-1) = yzad(k-1) - ypom(k-1);
    e(k-2) = yzad(k-2) - ypom(k-2);

    uk = r2*e(k-2) + r1*e(k-1) + r0*e(k) + u(k-1);
end
```

Mając zaimplementowane funkcje, przetestowano działanie algorytmu.



Rys. 25. Wyjście obiektu oraz sterowanie dla nierozmytego algorytmu PID

Na rysunku 7. przedstawiono regulację PID przy założeniu, że obiekt ma liniową charakterystykę statyczną. Tor wartości zadanej:

$$y_{zad1} = 34{,}3, y_{zad2} = 39{,}3, y_{zad3} = 49{,}3, y_{zad4} = 34{,}3$$

Za nastawy przyjęto:

$$K = 4, T_i = 60, T_d = 0{,}3$$

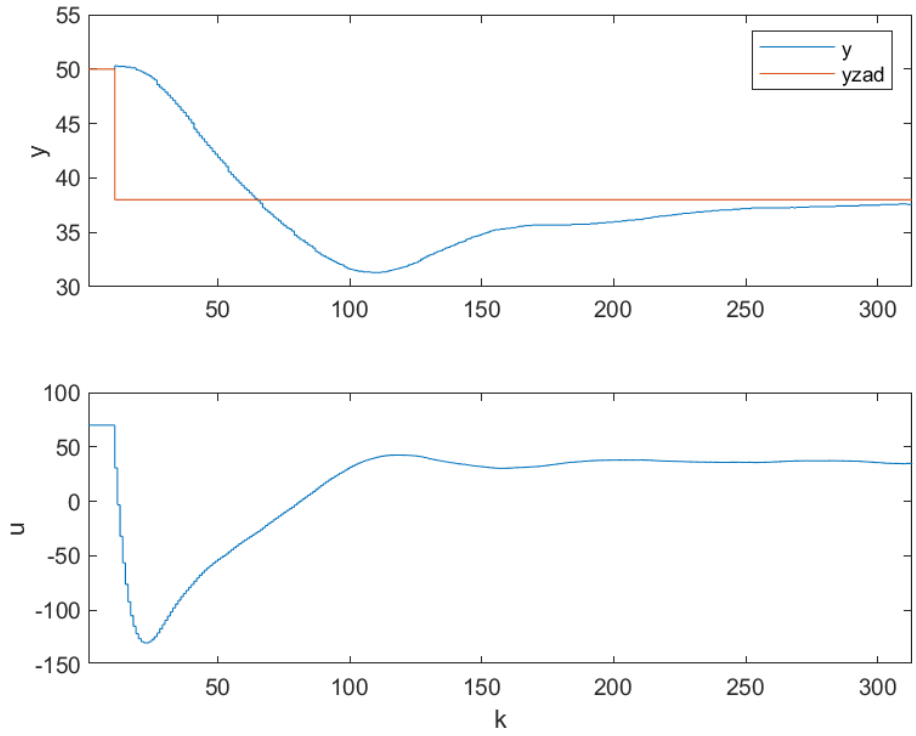
Uzyskany wskaźnik regulacji:

$$E = 12774$$

Oceniając jakość regulacji "na oko" można stwierdzić satysfakcjonujący wynik regulacji. Na wyjściu obiektu nie występują oscylacje, jest małe przesterowanie i odpowiedź ustala się w stosunkowo krótkim czasie (jak na właściwości inercyjne obiektu). Regulator zdecydowanie gorzej radzi sobie ze skokiem wartości zadanej do $y_{zad} = 50$, co może wskazywać na nieliniowość obiektu.

2.4. Regulacja DMC dla obiektu liniowego

Na rysunku 11. przedstawiono regulację dla nierozmytego algorytmu DMC.



Rys. 26. Regulacja nierozmyta DMC, $\lambda = 0{,}2$

Wskaźnik jakości regulacji wynosi $E = 6240$ i "na oko" można stwierdzić bardzo niską jakość regulacji. Na wyjściu pojawia się duże przesterowanie i czas ustalania się wyjścia na wartość zadaną jest stosunkowo długi. Prawdopodobnie wynika to ze źle dobranego parametru λ, który po zmniejszeniu wpłynąłby na szybsze ustalanie się wartości wyjściowej. Ze względu na ograniczony czas na laboratorium, regulację nierozymtą DMC przeprowadzono dla jedynie tej wartości λ.

2.5. Regulacja rozmyta PID

Mając charakterystykę statyczną obiektu, można wyznaczyć rozmyty regulator PID. Poniżej przedstawiono główną pętlę sterowania.

```
while(1)  % na rzeczywistym obiekcie
   k = k + 1;
   y(k) = obiekt12(u(k-1)); % rzeczywisty obiekt

   if y(k) > Ymax
       y(k) = Ymax;
   elseif y(k) < Ymin
       y(k) = Ymin;
   end

   u(k) = PID_rozmytysim(Pid1, Pid2, Pid3, T, k, u, y, yzad);

   if u(k) > Umax
       u(k) = Umax;
   elseif u(k) < Umin
       u(k) = Umin;
```

```
        end
    end
```

Gdzie funkcja $PID\_rozmytysim()$ wygląda:

```
function u_wyj=PID_rozmytysim(Pid1, Pid2, Pid3, T, k, u, y, yzad)

    K_pid1 = Pid1(1); Ti1 = Pid1(2); Td1 = Pid1(3);
    K_pid2 = Pid2(1); Ti2 = Pid2(2); Td2 = Pid2(3);
    K_pid3 = Pid3(1); Ti3 = Pid3(2); Td3 = Pid3(3);

    % parametry r regulatora PID
    r0_1 = K_pid1*(1+T/(2*Ti1)+Td1/T);
    r1_1 = K_pid1*(T/(2*Ti1)-2*Td1/T-1);
    r2_1 = K_pid1*Td1/T;

    r0_2 = K_pid2*(1+T/(2*Ti2)+Td2/T);
    r1_2 = K_pid2*(T/(2*Ti2)-2*Td2/T-1);
    r2_2 = K_pid2*Td2/T;

    r0_3 = K_pid3*(1+T/(2*Ti3)+Td3/T);
    r1_3 = K_pid3*(T/(2*Ti3)-2*Td3/T-1);
    r2_3 = K_pid3*Td3/T;

    % uchyb regulacji
    e(k) = yzad(k)-y(k);
    e(k-1) = yzad(k-1)-y(k-1);
    e(k-2) = yzad(k-2)-y(k-2);

    % sygnał sterujący regulatora PID
    u1 = r2_1*e(k-2) + r1_1*e(k-1) + r0_1*e(k) + u(k-1);
    u2 = r2_2*e(k-2) + r1_2*e(k-1) + r0_2*e(k) + u(k-1);
    u3 = r2_3*e(k-2) + r1_3*e(k-1) + r0_3*e(k) + u(k-1);

    % liczenie wag
    if(y(k-1)<=43)
    % Full pid1
        w1 = 1;
        w2 = 0;
        w3 = 0;

    elseif (y(k-1) >= 43 && y(k-1) <= 49)
    % Full pid2
        w1 = 0;
        w2 = 1;
        w3 = 0;

    elseif(y(k-1)>=49)
    % Full pid3
        w1 = 0;
        w2 = 0;
        w3 = 1;
```

```
    end

    u_wyj = (w1*u1 + w2*u2 + w3*u3) / (w1+w2+w3);
end
```

Przyjęto, że regulator PID składać się będzie z 3 regulatorów lokalnych, których nastawy są przekazywane jako argument funkcji $PID\_rozmytysim()$. Następnie na ich podstawie wyliczane są lokalne parametry r, z których uzyskiwany jest sygnał u dla każdego lokalnego regulatora. Ponieważ charakterystyka statyczna obiektu jest funkcją łamaną, na początku przyjęto dyskretność wpływu wag (czyli, że waga może być równa albo 1 albo 0). Należy podkreślić, że jest to jedynie wstępne założenie, jednakże na laboratorium nie starczyło czasu na wyliczanie wag funkcjami (tak jak zostało zaimplementowane to w projekcie). Punkty "przełączania" wag na wartości 1 i 0 zostały wyznaczone względem odpowiedzi obiektu, zgodnie z charakterystyką statyczną. Na sam koniec wyliczana jest całościowa wartośc sterowania, która zwracana jest przez funkcję.
Na rysunku 8. przedstawiono rozmytego regulatora PID założeniu. Tor wartości zadanej:

$$y_{zad1} = 34{,}3, y_{zad2} = 39{,}3, y_{zad3} = 49{,}3, y_{zad4} = 34{,}3$$

Za nastawy przyjęto:

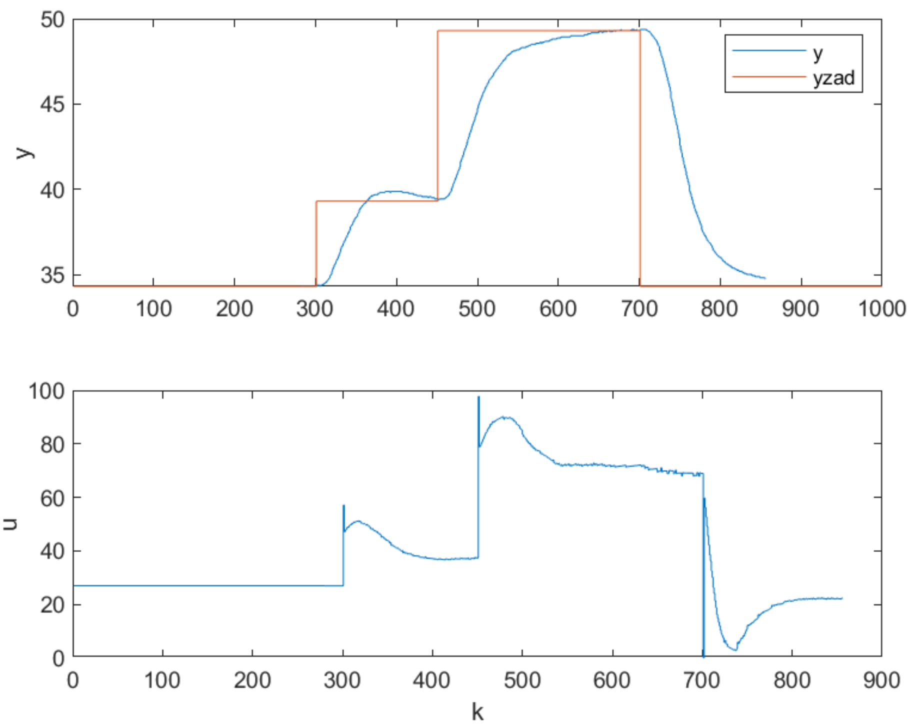
$$K_1 = 4, T_{i1} = 50, T_{d1} = 0{,}5$$

$$K_2 = 8, T_{i2} = 60, T_{d2} = 0{,}3$$

$$K_3 = 14, T_{i3} = 60, T_{d3} = 0{,}3$$

Uzyskany wskaźnik regulacji:

$$E = 14008$$



Rys. 27. Wyjście obiektu oraz sterowanie dla rozmytego algorytmu PID

W porównaniu do regulacji nierozmytego PID, ten regulator radzi sobie o wiele gorzej dla takiego samego toru wartości zadanej. Ma wyraźnie wyższy błąd regulacji E, zdecydowanie później się ustala oraz widoczne jest większe przesterowanie.

Na optymalizację nastaw nie starczyło już czasu na laboratorium.J est to jedna z głównych wad regulatorów rozmytych - dobieranie parametrów jest bardzo czasochłonną i żmudną czynnością i nie zawsze gwarantowane jest, że uzyska się lepszą jakoś regulacji, niż gdyby przyjąć model obiektu za liniowy.

2.6. Regulacja rozmyta DMC

Na wstępie należy zaznaczyć, że ze względu na małą ilość czasu na laboratorium, regulacja rozmytego algorytmu DMC nie została przetestowana.
Główna pętla programu została zaimplementowana w poniższy sposób.

```
    while(1)  % na rzeczywistym obiekcie
        k = k + 1
        y(k)=obiekt12(u(k-1));
        u(k)=dmc_romzyty(D, N, Nu, lambda, k, T, s{1}, s{2}, s{3})
    end
```

Funkcja *dmc_rozmyty*():

```
function uResult = dmc_romzyty(D, N, Nu, lambda, k, T, ...
                                s{1}, s{2}, s{3})

    lRegul = 3;
    %% inicjalizacja sterowania i wyjscia symulacji
    for r = 1:lRegul
        u{r} = ones(1, kk)*Upp;
    end
    y(1:kp-1) = Ypp;
    y(kp:kk) = 0;
    uResult(1:kp-1) = Upp;
    uResult(kp:kk) = 0;

    %% macierz M
    for r = 1:lRegul
        m = zeros(N, Nu);
        for i = 1 : Nu
            m(i : N, i) = s{r}(1 : (N - i + 1))';
        end
        M{r} = m;
    end

    %% macierz Mp
    for r = 1:lRegul
        mp = zeros(N, D-1);
        for i = 1 : N
           for j = 1 : D-1
              if i+j <= D
                 mp(i, j) = s{r}(i+j) - s{r}(j);
              else
                 mp(i, j) = s{r}(D) - s{r}(j);
              end
```

```
            end
        end
        Mp{r} = mp;
    end

    %% obliczam K
    I = eye(Nu);
    for r = 1:lRegul
        K{r} = ((M{r}'*M{r} + lambda*I)^(-1))*M{r}';
    end

    %% oszczędna wersja DMC - parametry
    for r = 1:lRegul
        Ke{r} = sum(K{r}(1,:));
        ku{r} = K{r}(1, :)*Mp{r};
    end

    % wagi
    w = wagi(uResult(k-1));

    % obiekt
    y(k) = obiek12(u(k-1))

    % uchyb
    e = yzad(k) - y(k);

    for r = 1:lRegul
        elem = 0;
        for j = 1 : D-1
            if k-j <= 1
                du = 0;
            else
                du = u{r}(k-j) - u{r}(k-j-1);
            end
            elem = elem + ku{r}(j)*du;
        end

        % optymalny przyrost sterowania w chwili k (du(k|k))
        dukk = Ke{r} * e - elem;

        % prawo regulacji
        u{r}(k) = dukk + u{r}(k-1);

        uResult(k) = uResult(k) + w(r)*u{r}(k);
    end
    uResult(k) = uResult(k) / sum(w);

    % ograniczenia sterowania
    if uResult(k) > Umax
        uResult(k) = Umax;
    elseif uResult(k) < Umin
```

```
        uResult(k) = Umin;
    end
end
```

W powyższej implementacji wywoływana jest funkcja *wagi*():

```
function w = wagi(u)
    % liczenie wag
    if(y(k-1)<=43)
    % Full pid1
        w1 = 1;
        w2 = 0;
        w3 = 0;

    elseif (y(k-1) >= 43 && y(k-1) <= 49)
    % Full pid2
        w1 = 0;
        w2 = 1;
        w3 = 0;

    elseif(y(k-1)>=49)
    % Full pid3
        w1 = 0;
        w2 = 0;
        w3 = 1;
    end
    w =[w1, w2, w3];
end
```

Ze względu na ograniczoną ilość czasu, ponownie tak jak w przypadku rozmytego regulatora PID, wagi wyliczane były w sposób zero-jedynkowy. Punkty ”przełączania” wartości wag na 0 lub 1 również zostały wyznaczone względem odpowiedzi obiektu, zgodnie z charakterystyką statyczną.

Tak jak zostało podkreślone to na wstępie tej sekcji, czas na zajęciach był ograniczony i nie wystarczyło go na przetestowanie działania rozmytego regulatora DMC oraz na dobranie odpowiedniego parametru λ.