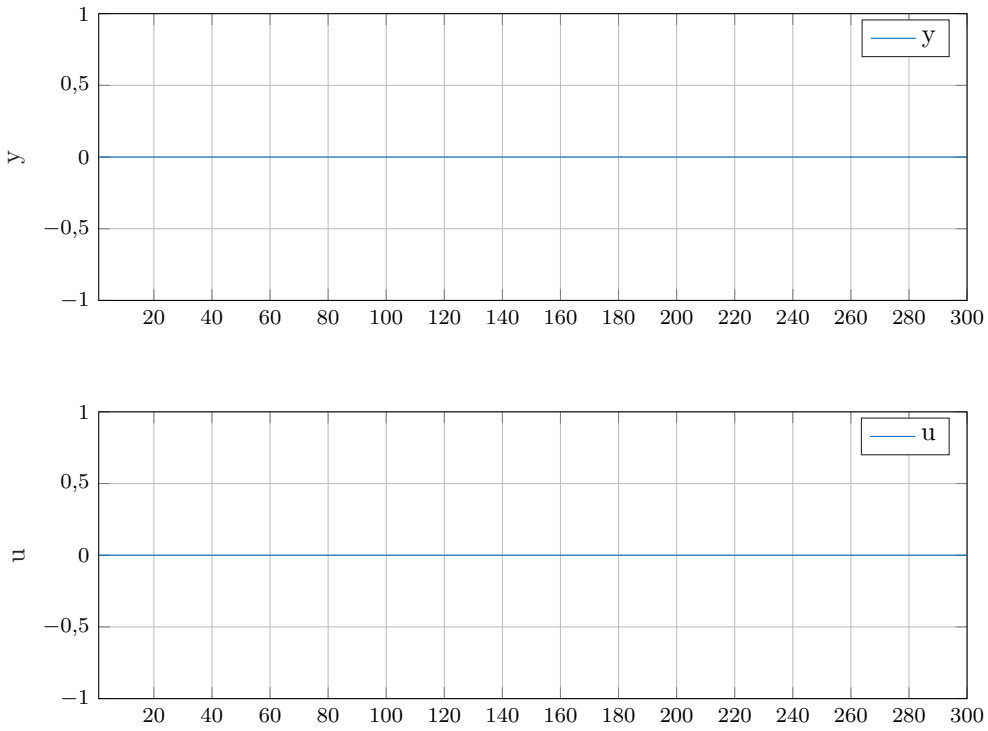
Spis treści

1. Projekt

1.1. Poprawność podanego punktu pracy

Przeprowadzono symulację odpowiedzi procesu dla punktu pracy. Ustalone zostało stałe sterowanie $U_{pp} = 0$.



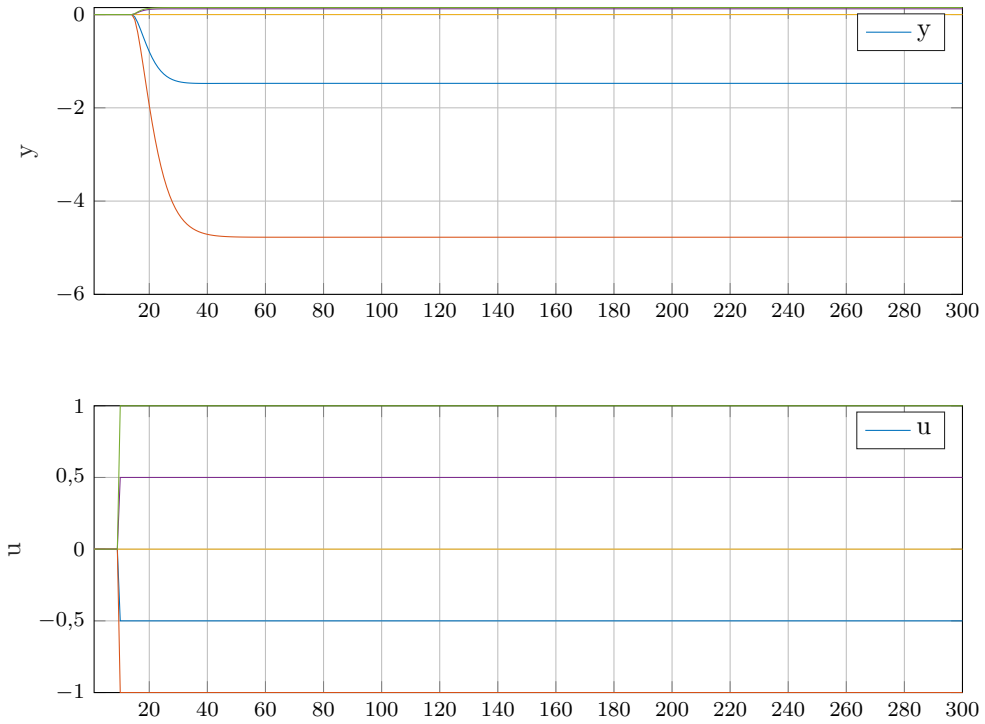
Rys. 1.1. Punkt pracy

Wynik:
Uzyskany punkt pracy wyjścia $Y_{pp} = 0$.
Wniosek:
Stała wartość wyjścia oznacza poprawność danych punktu pracy.

1.2. Symulacyjne wyznaczenie odpowiedzi skokowych

Odpowiedzi skokowe zostały wyznaczone symulacyjnie dla pięciu zmian sygnału sterującego.

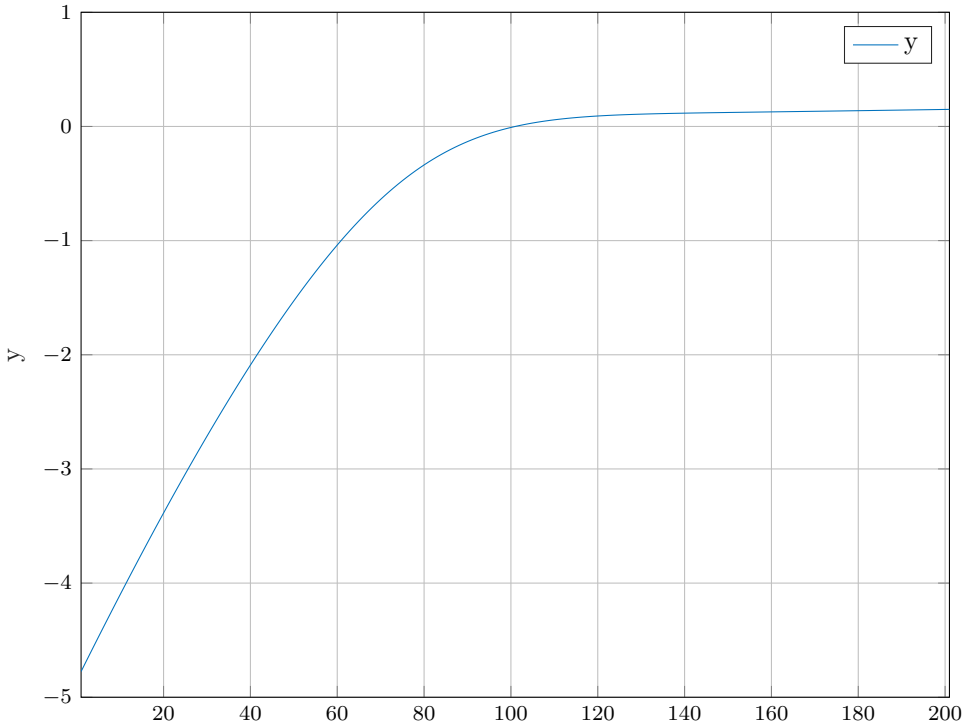
1.2.1. Odpowiedzi skokowe



Rys. 1.2. Odpowiedzi skokowe obiektu

1.2.2. Charakterystyka statyczna

Na podstawie serii odpowiedzi skokowych wyznaczono charakterystykę statyczną



Rys. 1.3. Charakterystyka statyczna obiektu

Na podstawie wykresu charakterystyki statycznej można ustalić, że właściwości statyczne procesu są nieliniowe.

1.3. Program do symulacji algorytmów PID i DMC

1.3.1. Klasyczny algorytm PID

W środowisku MATLAB napisano skrypt implementujący algorytm klasycznego regulatora PID

Inicjacja symulacji

Listing 1.1. "Inicjacja parametrów symulacji"

```
% wybor symulacja czy obiekt rzeczywisty
% 'simulation' lub 'real'
mode = 'simulation';


if( isequal(regulator, 'real') )
    addpath ('F:\SerialCommunication') ; % add a path
    initSerialControl COM5 % initialise com port
end
% wybor trybu pracy
% setpoint, stepsU, stepsYzad
action = 'stepsYzad';
% wybor regulatora
% none, PID_linear, DMC_linear, PID_fuzzy, DMC_fuzzy
regulator = 'PID_fuzzy';

% parametry skryptu
kk = 2500;
Tp = 1; % okres probkowania
T1 = 39.4; % temperatura punktu pracy
G1 = 32;   % sterowanie punktu pracy
y = ones(1,kk)*T1; % wektor wyjsc obiektu
yzad = ones(1,kk)*T1; % wektor wartosc zadanych
e = zeros(1,kk); % wektor uchybow regulacji
u = ones(1,kk)*G1; % wektor wejsc (sterowan) obiektu
offset = 400;

%ograniczenia wartosci sterowania
if( isequal(regulator, 'real') )
    umax = 100;
    umin = 0;
else
    umax = 1;
    umin = -1;
end
```

Listing 1.2. "Wyznaczanie nastaw klasycznego regulatora PID"

```
% regulacja z uzyciem regulatora PID linear
if( isequal(regulator, 'PID_linear') )
    %nastawy klasycznego regulatora PID
    Kr = 3.7;
    Ti = 60.5;
    Td = 0.0;
    settings(1).N = 1;
    settings(1).Tp = Tp;
    settings(1).r0 = Kr*(1+Tp/(2*Ti)+Td/Tp);
    settings(1).r1 = Kr*(Tp/(2*Ti)-2*Td/Tp-1);
    settings(1).r2 = Kr*Td/Tp;
    settings(1).u1 = G1;
```

Pętla główna symulacji

Listing 1.3. "Wyznaczenie bieżącej wartości wyjścia obiektu y"

```
% obtaining measurements
if( isequal(regulator, 'real') )
    measurements = readMeasurements (1:7) ; % read measurements
    y(k)=measurements(1); % powiekszamy wektor y o element Y
else
    y(k)=symulacja_obiektu7y(u(k-5),u(k-6),y(k-1),y(k-2));
end
```

Listing 1.4. "Wyznaczenie wartości zadanej yzad"

```
% podazanie do setpoint
if( isequal(action, 'setpoint') )
    %disp('setpoint')
    yzad(k) = T1;
    u(k) = G1;
% skoki sterowania
elseif( isequal(action, 'stepsU') )
    %disp('stepsU')
    if( k<=300 )        u(k) = 20; yzad(k) = u(k);
    elseif( k<=600 )    u(k) = 30; yzad(k) = u(k);
    elseif( k<=900 )    u(k) = G1; yzad(k) = u(k);
    elseif( k<=1200 )   u(k) = 40; yzad(k) = u(k);
    elseif( k<=1500 )   u(k) = 50; yzad(k) = u(k);
    elseif( k<=1800 )   u(k) = 60; yzad(k) = u(k);
    elseif( k<=2100 )   u(k) = 70; yzad(k) = u(k);
    else                u(k) = 80; yzad(k) = u(k);
    end
% skoki wartosci zadanej z regulatorem
elseif( isequal(action, 'stepsYzad') )
    %disp('stepsYzad')
    if( k<=offset+350 )       yzad(k) = T1;
    elseif( k<=offset+700 )   yzad(k) = T1 + 5;
    elseif( k<=offset+1050 )  yzad(k) = T1 + 15;
    else                      yzad(k) = T1;
    end
else
    error('NO VALID ACTION SELECTED');
end
```

Listing 1.5. "Wyznaczenie wartości sterowania u"

```
% regulacja z uzyciem regulatora PID linear
if( isequal(regulator, 'PID_linear') )
    %disp('pid_linear')
    %wyznaczenie nowej wartosci sterowania
    u(k) = getPIDcontrol(settings(1), e(k), e(k-1), e(k-2), u(k-1));
    %nalozenie ograniczen sterowania
    if( u(k)>umax )
        u(k) = umax;
    elseif( u(k)<umin )
        u(k) = umin;
    end
    %regulator moze uzyc poprzedniej wartosci sterowania
    settings(1).u1 = u(k);
```

1.3.2. Klasyczny algorytm DMC

W środowisku MATLAB napisano skrypt implementujący algorytm klasycznego regulatora DMC

Inicjacja symulacji

Listing 1.6. "Inicjacja parametrów symulacji"

```
% wybor symulacja czy obiekt rzeczywisty
% 'simulation' lub 'real'
mode = 'simulation';


if( isequal(regulator, 'real') )
    addpath ('F:\SerialCommunication') ; % add a path
    initSerialControl COM5 % initialise com port
end
% wybor trybu pracy
% setpoint, stepsU, stepsYzad
action = 'stepsYzad';
% wybor regulatora
% none, PID_linear, DMC_linear, PID_fuzzy, DMC_fuzzy
regulator = 'PID_fuzzy';

% parametry skryptu
kk = 2500;
Tp = 1; % okres probkowania
T1 = 39.4; % temperatura punktu pracy
G1 = 32;   % sterowanie punktu pracy
y = ones(1,kk)*T1; % wektor wyjsc obiektu
yzad = ones(1,kk)*T1; % wektor wartosc zadanych
e = zeros(1,kk); % wektor uchybow regulacji
u = ones(1,kk)*G1; % wektor wejsc (sterowan) obiektu
offset = 400;

%ograniczenia wartosci sterowania
if( isequal(regulator, 'real') )
    umax = 100;
    umin = 0;
else
    umax = 1;
    umin = -1;
end
```

Listing 1.7. "Wyznaczanie nastaw klasycznego regulatora DMC"

```
% regulacja z uzyciem regulatora DMC linear
elseif( isequal(regulator, 'DMC_linear') )
        %nastawy klasyczengo regulatora DMC
        load('s.mat')
        D = length(s); % horyzont dynamiki
        N=200;
        Nu=10;
        lambda = 1;
        run('DMC_init.m');
        n=1;
        settings(n).lambda = lambda;
        settings(n).N = N;
        settings(n).Nu = Nu;
        settings(n).D = D;
        settings(1).Mp = Mp;
        settings(1).K = K;
        settings(1).deltaUp = deltaUp;
```

Pętla główna symulacji

Listing 1.8. "Wyznaczenie bieżącej wartości wyjścia obiektu y"

```
% obtaining measurements
if( isequal(regulator, 'real') )
    measurements = readMeasurements (1:7) ; % read measurements
    y(k)=measurements(1); % powiekszamy wektor y o element Y
else
    y(k)=symulacja_obiektu7y(u(k-5),u(k-6),y(k-1),y(k-2));
end
```

Listing 1.9. "Wyznaczenie wartości zadanej yzad"

```
% podazanie do setpoint
if( isequal(action, 'setpoint') )
    %disp('setpoint')
    yzad(k) = T1;
    u(k) = G1;
% skoki sterowania
elseif( isequal(action, 'stepsU') )
    %disp('stepsU')
    if( k<=300 )        u(k) = 20; yzad(k) = u(k);
    elseif( k<=600 )    u(k) = 30; yzad(k) = u(k);
    elseif( k<=900 )    u(k) = G1; yzad(k) = u(k);
    elseif( k<=1200 )   u(k) = 40; yzad(k) = u(k);
    elseif( k<=1500 )   u(k) = 50; yzad(k) = u(k);
    elseif( k<=1800 )   u(k) = 60; yzad(k) = u(k);
    elseif( k<=2100 )   u(k) = 70; yzad(k) = u(k);
    else                u(k) = 80; yzad(k) = u(k);
    end
% skoki wartosci zadanej z regulatorem
elseif( isequal(action, 'stepsYzad') )
    %disp('stepsYzad')
    if( k<=offset+350 )       yzad(k) = T1;
    elseif( k<=offset+700 )   yzad(k) = T1 + 5;
    elseif( k<=offset+1050 )  yzad(k) = T1 + 15;
    else                      yzad(k) = T1;
    end
else
    error('NO VALID ACTION SELECTED');
end
```

Listing 1.10. "Wyznaczenie wartości sterowania u"

```
% regulacja z uzyciem regulatora DMC linear
elseif( isequal(regulator, 'DMC_linear') )
    %disp('DMC_linear')
            settings(1).u1 = u(k-1);
            u(k) = getDMCcontrol(settings(1), y, k, yzad);
    %nalozenie ograniczen sterowania
    if( u(k)>umax )
        u(k) = umax;
    elseif( u(k)<umin )
        u(k) = umin;
    end
            settings(1).deltaUp(k) = u(k)-u(k-1);
```

1.4. Dobór parametrów klasycznych regulatorów PID i DMC

Ocena jakości regulacji na podstawie rysunków przebiegów sygnałów polega na ocenie jakościowej, analizowane są takie kryteria jak czas regulacji czy przeregulowanie. Na podstawie tych kryteriów oceny dobierane są nastawy regulatora PID.

Ocena jakości regulacji ilościowa polega na wyznaczaniu wskaźnika jakości regulacji, którym jest suma kwadratów uchybów, a k_{konc} oznacza ilość kroków symulacji.

$$E = \sum_{k=1}^{k_{konc}} (y_{zad}(k) - y(k))^2$$

Ocena jakości regulacji jakościowa jest metodą mniej dokładną od oceny ilościowej, nie występują tam obliczenia a jedynie analiza rysunków przebiegów. Podczas analizy rysunku osoba oceniająca jakość regulacji narażona jest na mało precyzyjne wnioski, gdyż należy wtedy zwracać uwagę na skalę rysunku i wiele innych parametrów. Ocena ilościowa nie bierze pod uwagę takich czynników jak np. oscylacje, jedynym kryterium jest suma kwadratów uchybów. Jeżeli do oceny jakości regulacji wymagane są inne kryteria niż podany wskaźnik jakości należy rozpatrzeć metodę oceny jakościowej.

1.4.1. Klasyczny algorytm PID

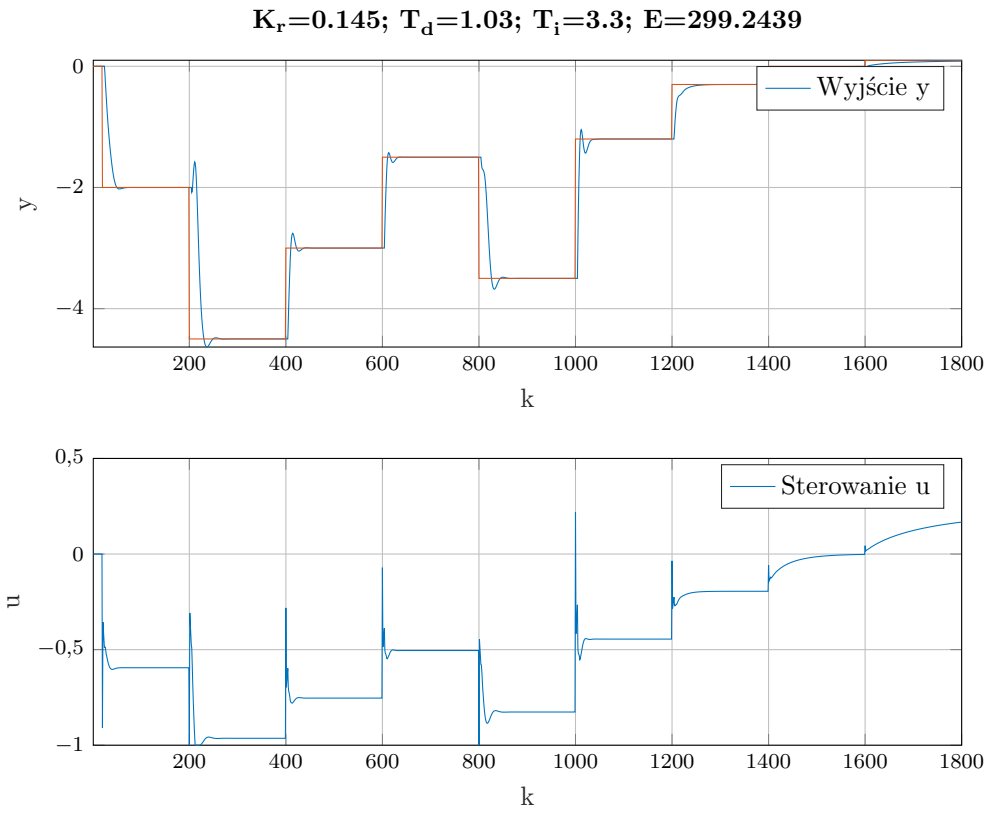
Regulator został nastrojony przy użyciu metody Zieglera-Nicholsa.

Metoda Zieglera-Nicholsa

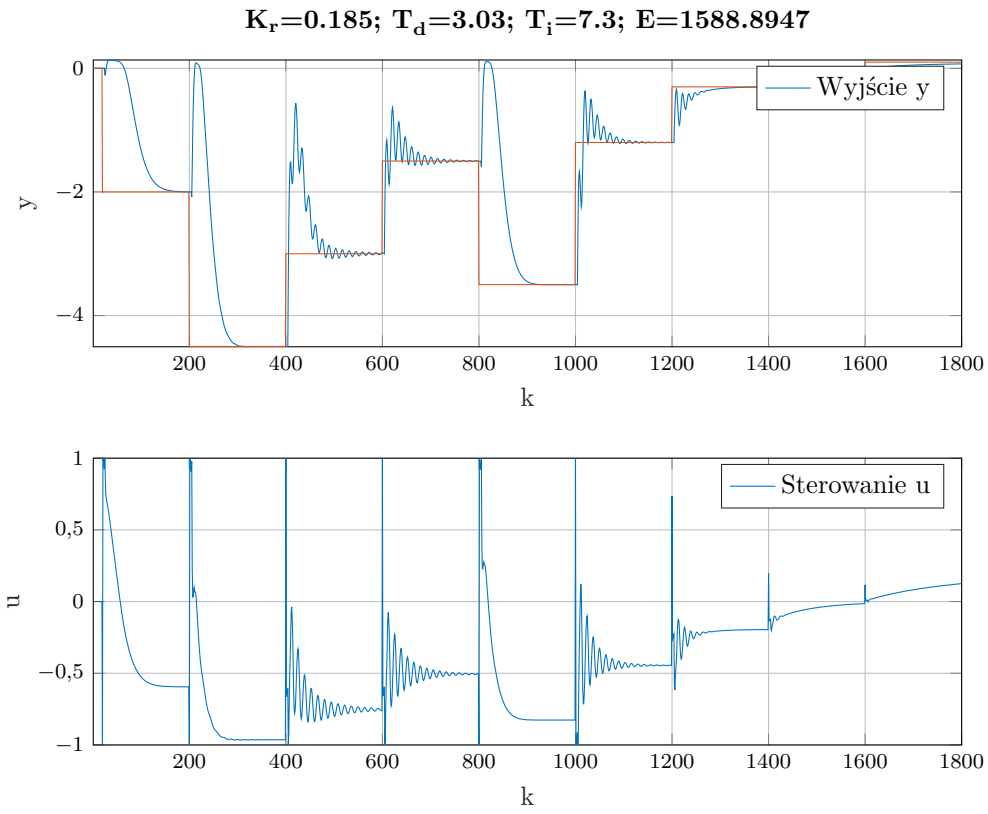
Metoda Zieglera-Nicholsa polega na nastawieniu regulatora na działanie proporcjonalne i zwiększaniu wzmocnienia doprowadzając układ do granicy stabilności. W stanie na granicy stabilności (oscylacje niegasnące) należy odczytać współczynnik wzmocnienia krytycznego układu K_k oraz okres oscylacji T_k. Na podstawie tych wartości obliczamy nastawy regulatora PID za pomocą poniższych równań:

$$K_r = 0.6K_k \qquad T_i = 0.65T_k \qquad T_d = 0.12T_k$$

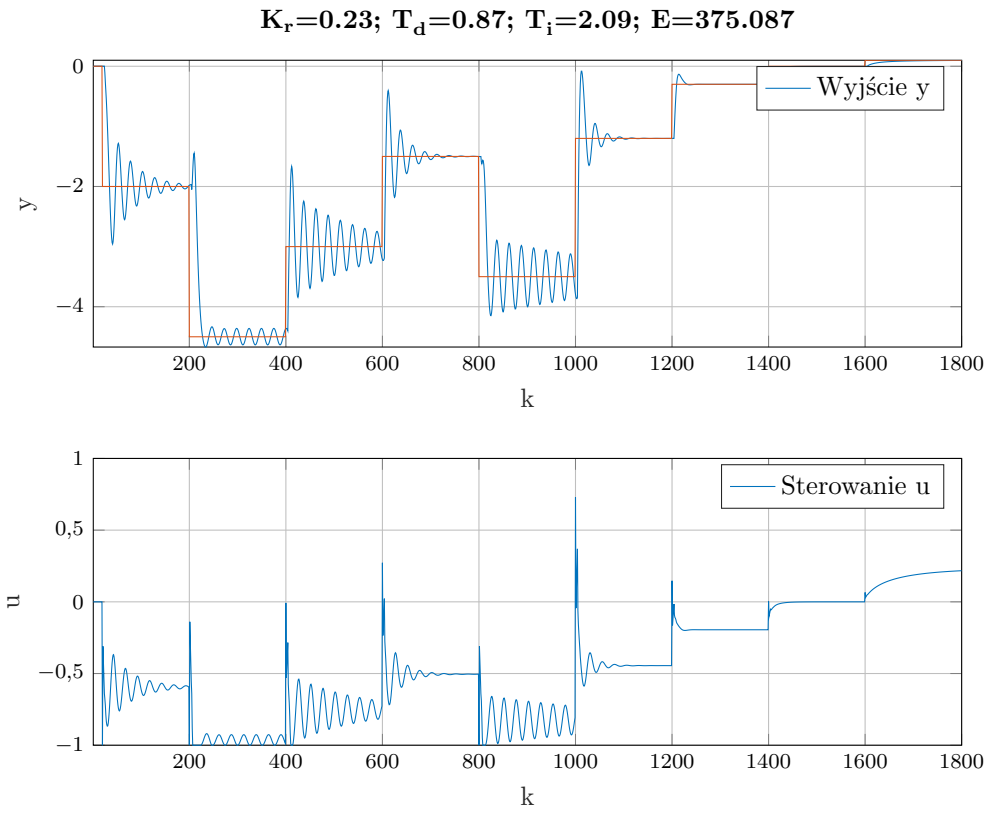
Dobrane regulatory klasyczne PID



Rys. 1.4. Regulator klasyczny PID pierwszy



Rys. 1.5. Regulator klasyczny PID drugi



Rys. 1.6. Regulator klasyczny PID trzeci

1.4.2. Klasyczny algorytm DMC

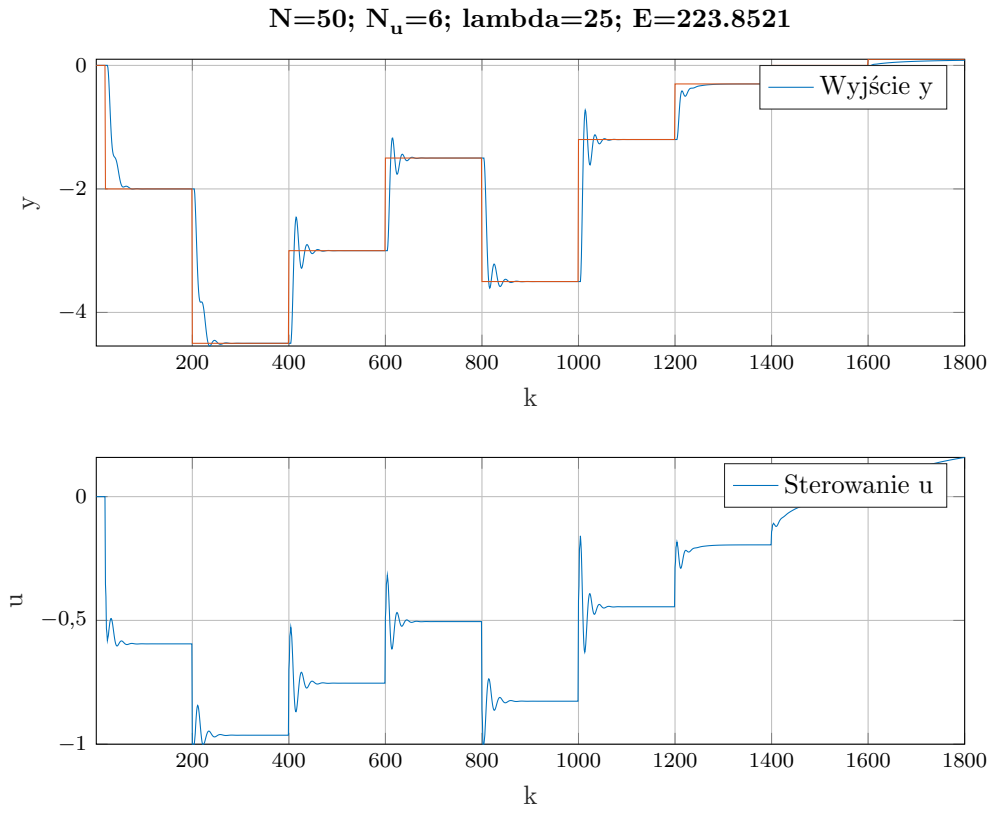
Do doboru nastaw algorytm DMC zastosowano metodę inżynierską, która polega na przeprowadzeniu doświadczeń i analizy uzyskanych wyników. Na podstawie wyciągniętych wniosków modyfikowane są nastawy regulatora. Parametry dobierane są metodą prób i błędów, aż do osiągnięcia oczekiwanych wyników.

Horyzont dynamiki D jest to liczba współczynników odpowiedzi skokowej, tzn. liczbę kroków dyskretyzacji, po której można uznać odpowiedź skokową za stabilną równą K_{stat}. Wyznaczona ona została z odpowiedzi skokowej obiektu poprzez wyznaczenie z niej chwili, w której odpowiedź jest stabilna.

Horyzont predykcji N jest to wartość na podstawie, której prognozuje się zachowanie modelu. Zwiększając ten parametr uzyskaliśmy bardzo dobry czas regulacji oraz praktycznie zerowe przesterowanie.

Horyzont sterowania N_u tak jak horyzont predykcji jest parametrem dostrajania regulatora, zależnymi od szybkości dynamiki procesu, możliwości obliczeniowych oraz dokładności modelu, eksperymentalnie dobrano wartość tero parametru.

Ostatnim krokiem w dostrajaniu naszego regulatora było wyznaczenie współczynnika kary lambda, za pomocą którego można zapewnić kompromis pomiędzy szybkością regulacji, a postacią sygnału sterującego. Ponownie był on wyznaczany metodą testowania. Zwiększenie współczynnika kary pogorszyło wynik działania regulatora, został więc on zmniejszony.



Rys. 1.7. Regulator klasyczny DMC

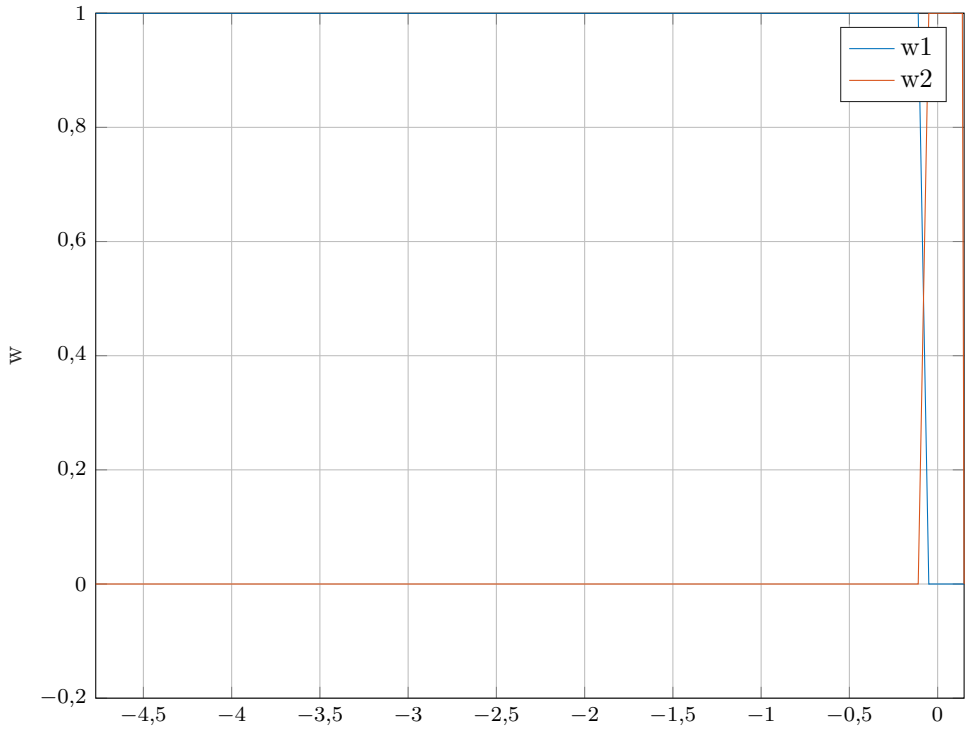
1.5. Implementacja rozmytych algorytmów PID i DMC

Poprzez zastosowanie wielu lokalnych regulatorów dla poszczególnych punktów pracy oraz funkcji przynależności zaimplementowano rozmyte algorytmy PID oraz DMC.

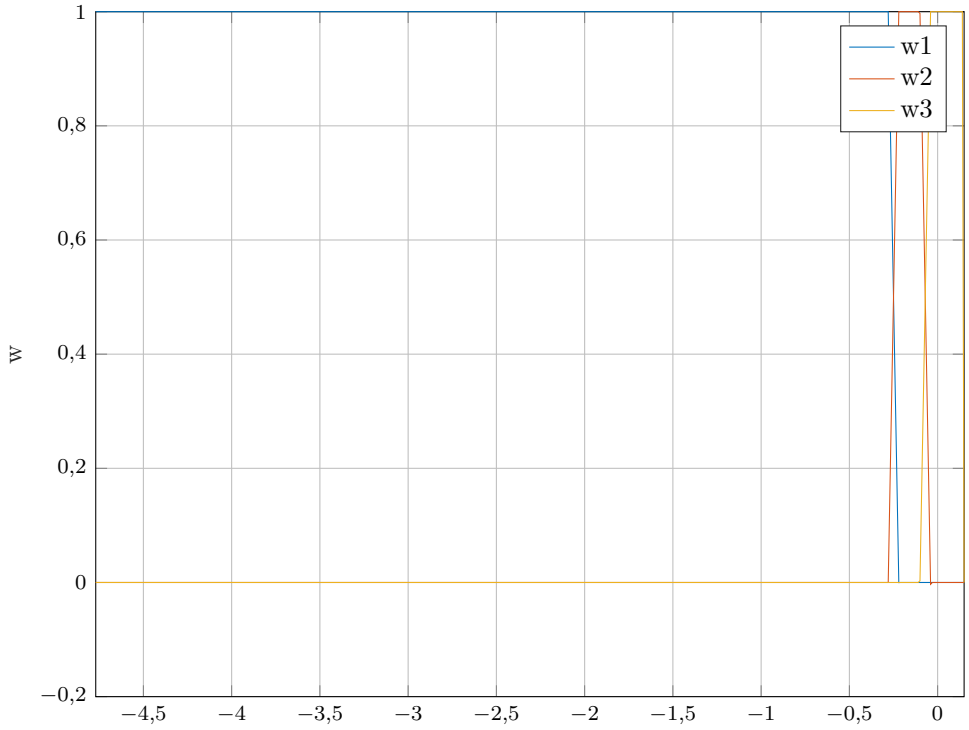
1.5.1. Funkcje przynależności

Wybór regulatora dokonywany jest dzięki funkcji przynależności. Jako funkcja przynależności posłużyła nam funkcja trapezoidalna. Funkcja ta zawiera współczynniki, które definiują jak duży wpływ na dane wyjście ma dany regulator w badanym obszarze. Dobór tych współczynników został przeprowadzony poprzez obserwacje zachowania obiektu. Dla każdego z lokalnych regulatorów PID oraz DMC należało dobrać parametry osobno w jego punkcie pracy, w tym celu zostały przeprowadzone eksperymenty, parametry dobrane zostały metodą inżynierską. Ostatecznie regulatory zostały połączone w funkcji. Dzięki wykorzystaniu rozmytego algorytmu PID oraz DMC możemy zniwelować problem nieliniowości obiektu. Rozmywanie dokonywane jest na podstawie charakterystyki statycznej, dobór parametrów na podstawie aktualnej wartości sygnału wyjściowego y. Kształt trapezoidalny funkcji przynależności został wybrany na podstawie eksperymentów, dla trapezoidalnej funkcji wagi zmieniają się proporcjonalnie. Kształt trapezu pozwala na utrzymanie wartości 1 co w przypadku trójkątnej nie jest możliwe, natomiast na ramionach trapezu można dobrze uzupełniać funkcje przynależności równie dobrze jak w trójkątnej.

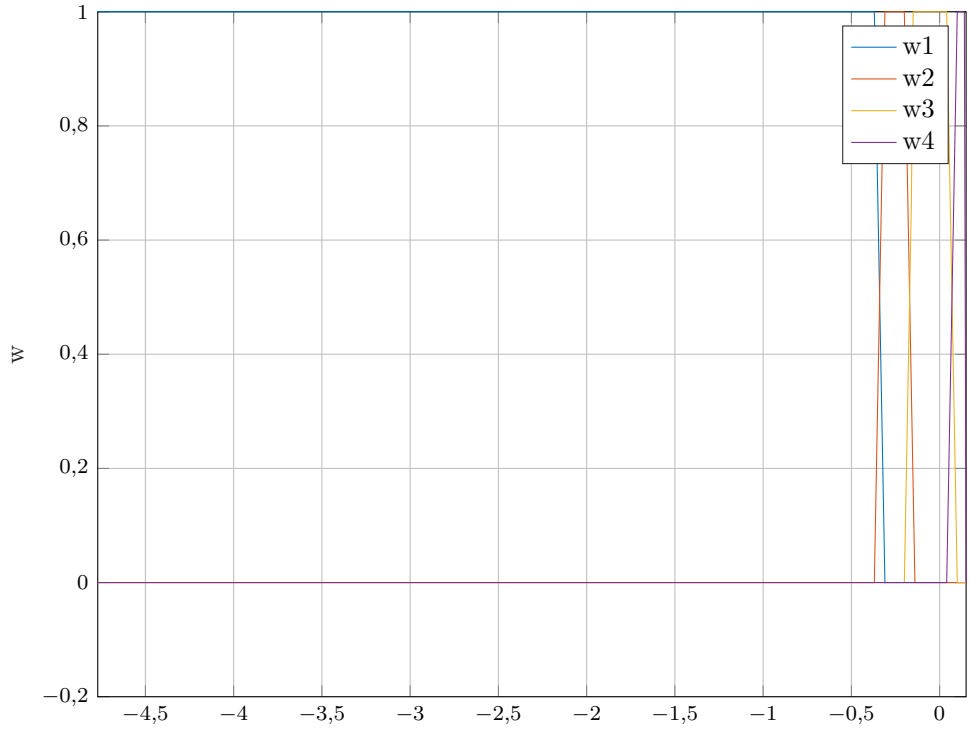
1.5.2. Wykresy funkcji przynależności



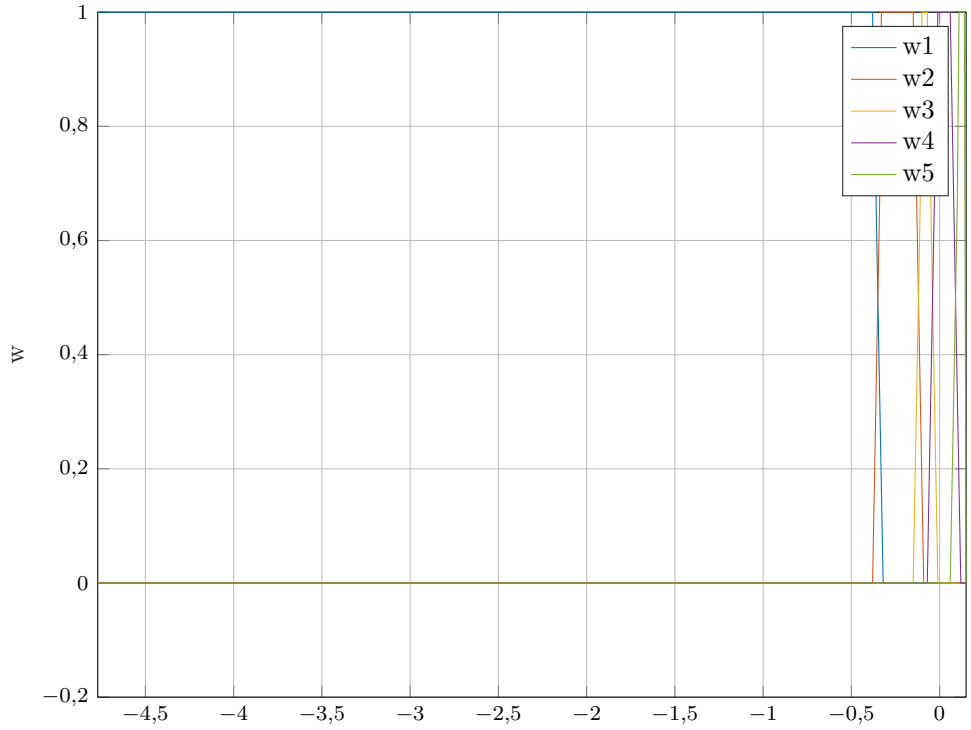
Rys. 1.8. Funkcje rozmycia dla 2 regulatorów lokalnych



Rys. 1.9. Funkcje rozmycia dla 3 regulatorów lokalnych



Rys. 1.10. Funkcje rozmycia dla 4 regulatorów lokalnych



Rys. 1.11. Funkcje rozmycia dla 5 regulatorów lokalnych

1.5.3. Rozmyty algorytm PID

Listing 1.11. "Wyznaczanie nastaw rozmytych regulatorów PID"

```
% regulacja z uzyciem regulatora PID fuzzy
elseif( isequal(regulator, 'PID_fuzzy') )
    %nastawy regulatorow PID
    Kr = [ 4.2  6.5     8.35];  % P
    Ti = [ 67.8 83.7    90.4];  % I
    Td = [ 0.0  0.0     0.3 ];  % D

    fuzzyN = length(Kr);
    settings(fuzzyN).N = fuzzyN;
    for n=1:fuzzyN
        settings(n).Tp = Tp;
        settings(n).r0 = Kr(n)*(1+Tp/(2*Ti(n))+Td(n)/Tp);
        settings(n).r1 = Kr(n)*(Tp/(2*Ti(n))-2*Td(n)/Tp-1);
        settings(n).r2 = Kr(n)*Td(n)/Tp;
        settings(n).u1 = G1;
    end
```

Listing 1.12. "Wyznaczenie wartości sterowania u"

```
    % regulacja z uzyciem regulatora PID fuzzy
    elseif( isequal(regulator, 'PID_fuzzy') )
        u_ = zeros(1, fuzzyN);
        for n=1:fuzzyN
            %wyznaczenie nowej wartosci sterowania od regulatora
            u_(n) = getPIDcontrol(settings(n), e(k), e(k-1), e(k-2), settings(n).u1);
            %nalozenie ograniczen sterowania
            if( u_(n)>umax )
                u_(n) = umax;
            elseif( u_(n)<umin )
                u_(n) = umin;
            end
            %regulator moze uzyc poprzedniej wartosci sterowania
            settings(n).u1 = u_(n);

            %funkcja wagi regulatora
            w = [];
            if( isequal(mode, 'real') )
                w = fun_przyn_lab(y(k));
                w = w(n);
            else



            end
            w
            u_(n) = u_(n) * w;

        end
        u_
        % wyznaczenie sterowania rozmytego
        u(k) = sum(u_);
```

1.5.4. Rozmyty algorytm DMC

Listing 1.13. "Wyznaczanie nastaw rozmytych regulatorów DMC"

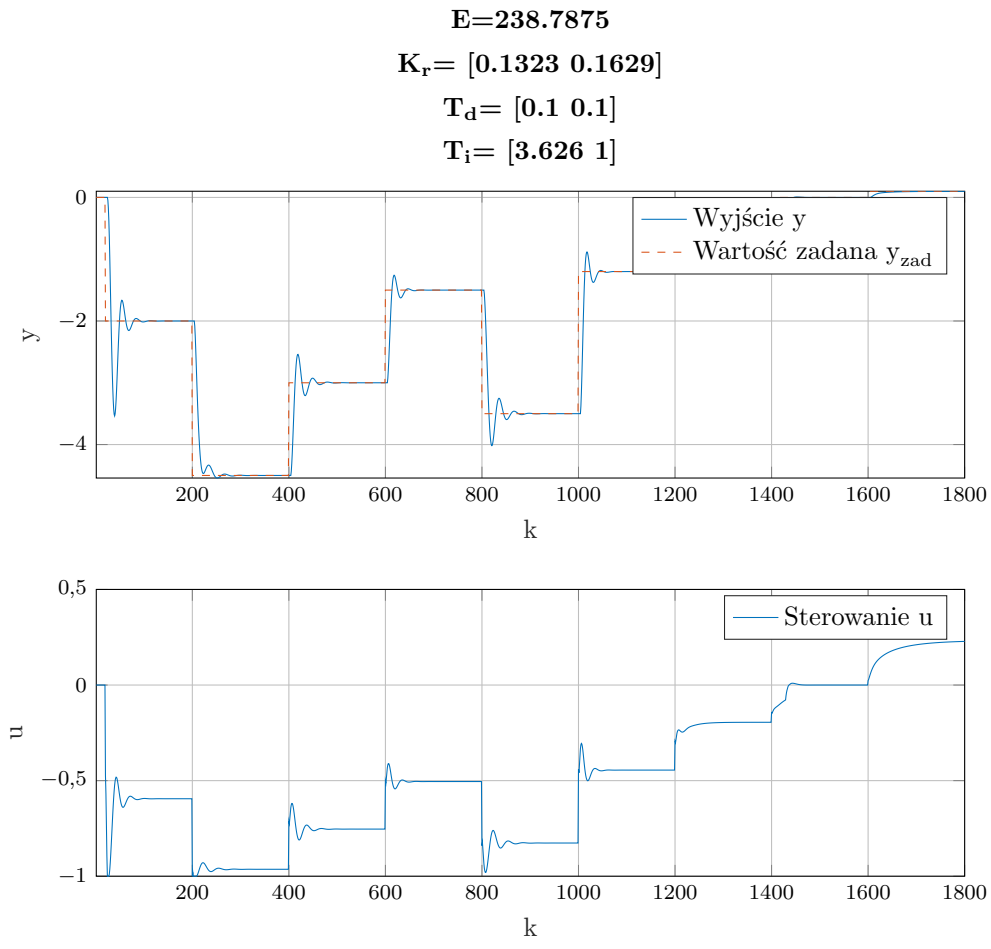
```
% regulacja z uzyciem regulatora DMC fuzzy
elseif( isequal(regulator, 'DMC_fuzzy') )
    fuzzyN = 3;
        %nastawy regulatora DMC
        N=200;
        Nu=10;

        lambdaVect = [1 1 1];
        for n=1:length(lambdaVect)
                lambda = lambdaVect(n);
        %wczytanie wektora s danego regulatora lokalnego
                load(strcat('s',num2str(n),'.mat'));
                D = length(s); % horyzont dynamiki
        N = D;
        Nu = N;
                run('DMC_init.m');

                settings(n).lambda = lambda;
                settings(n).N = N;
                settings(n).Nu = Nu;
                settings(n).D = D;
                settings(n).Mp = Mp;
                settings(n).K = K;
                settings(n).deltaUp = deltaUp;
        settings(n).u1 = G1;
        end
```

Listing 1.14. "Wyznaczenie wartości sterowania u"

```
    % regulacja z uzyciem regulatora DMC fuzzy
    elseif( isequal(regulator, 'DMC_fuzzy') )
        u_ = zeros(1, fuzzyN);
        for n=1:fuzzyN
            %wyznaczenie nowej wartosci sterowania od regulatora
                        u_(n) = getDMCcontrol(settings(1), y, k, yzad);
            %nalozenie ograniczen sterowania
            if( u_(n)>umax )
                u_(n) = umax;
            elseif( u_(n)<umin )
                u_(n) = umin;
            end
            %regulator moze uzyc poprzedniej wartosci sterowania

                        settings(n).deltaUp(k) = u_(n)-settings(n).u1;
            settings(n).u1 = u_(n);

            %funkcja wagi regulatora
            w = [];
            if( isequal(mode, 'real') )
                w = fun_przyn_lab(y(k));
                w = w(n);
            else



            end
            u_(n) = u_(n) * w
        end
        % wyznaczenie sterowania rozmytego
        u(k) = sum(u_);
```

1.6. Dobór parametrów lokalnych regulatorów PID i DMC

1.6.1. Rozmyty regulator PID



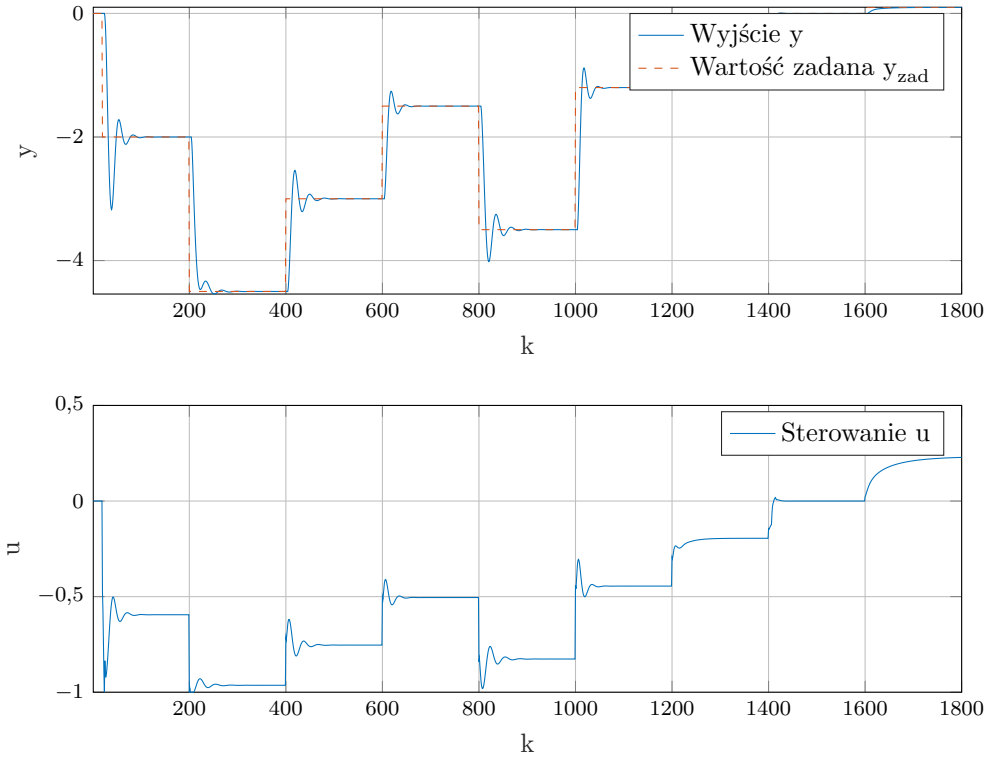
Rys. 1.12. Dwa regulatory lokalne PID

E=229.7839

K_r= [0.1323 0.1433 0.1629]

T_d= [0.1 0.1 0.1]

T_i= [3.626 1.333 1]



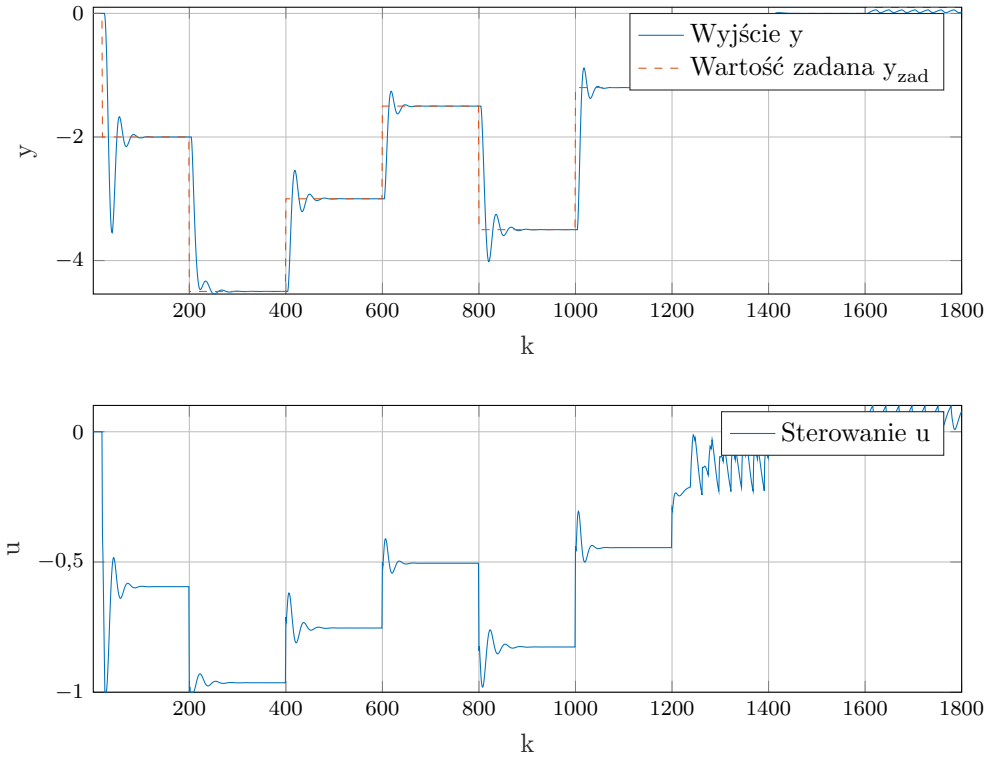
Rys. 1.13. Trzy regulatory lokalne PID

E=244.4525

K_r= [0.1323 0.1485 0.1363 0.2062]

T_d= [0.1 0.1 0.1 0.1]

T_i= [3.626 1.462 1.015 1]



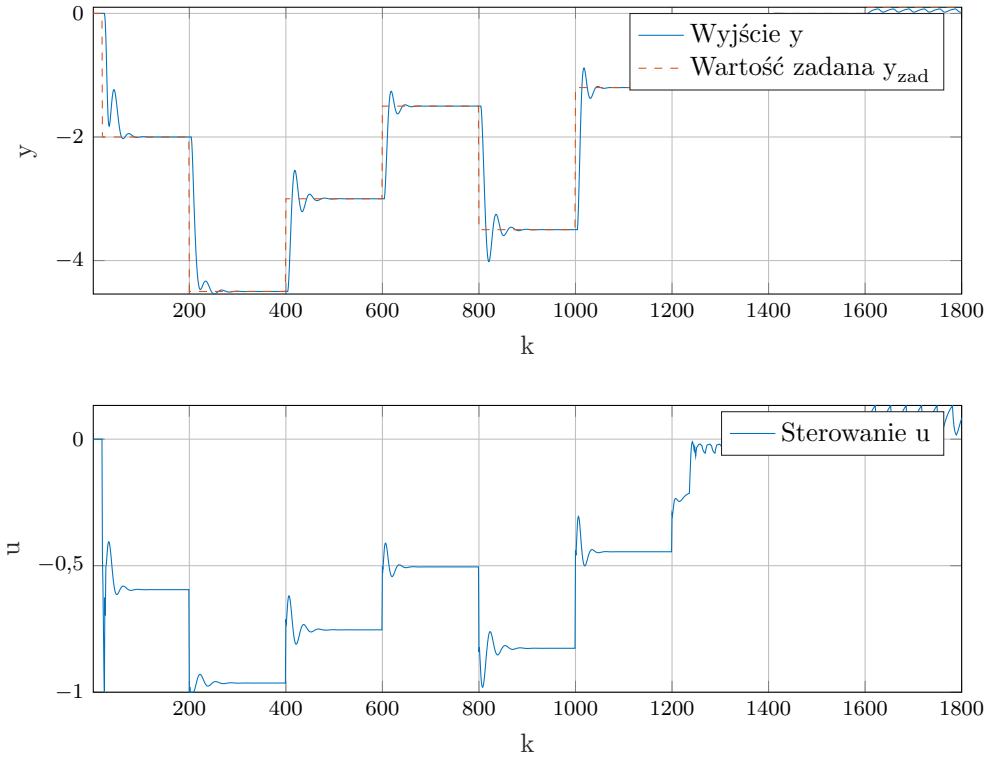
Rys. 1.14. Cztery regulatory lokalne PID

E=235.314

K_r= [0.1323 0.1544 0.1313 0.164 0.2128]

T_d= [0.1 0.1 0.1 0.1 0.1]

T_i= [3.626 1.587 1.016 1 1]



Rys. 1.15. Pięć regulatorów lokalnych PID

1.6.2. Rozmyty regulator DMC - ustalona lambda

1.6.3. Wnioski

Wykorzystanie rozmytego algorytmu regulacji dla obu regulatorów daje lepsze wyniki jakości regulacji w stosunku do pojedynczego regulatora liniowego.

W przypadku regulatorów PID najlepszy okazał się regulator rozmyty składający się z 3 regulatorów lokalnych. Wynika z tego, że większa ilość regulatorów lokalnych nie gwarantuje lepszej jakości regulacji.

W przypadku regulatorów DMC najlepszy okazał się regulator rozmyty składający się z regulatorów lokalnych. Wynika z tego, że większa ilość regulatorów lokalnych nie gwarantuje lepszej jakości regulacji.

1.7. Dobór parametrów lambda lokalnych regulatorów DMC

Dla zaproponowanej trajektorii zmian sygnału zadanego oraz dla róznej liczby regulatorów lokalnych (2, 3, 4, 5, . . .) spróbowac dobrac parametry lambda dla kazdego z lokalnych regulatorów DMC. Zamiescic wyniki symulacji.

2. Laboratorium