

Wydział Elektroniki i Technik Informacyjnych  
Politechnika Warszawska

Projektowanie układów sterowania  
(projekt grupowy)

Sprawozdanie z projektu i ćwiczenia laboratoryjnego  
nr 1, zadanie nr 12

Paulina Dąbrowska, Miłosz Kowalewski,  
Adam Rybojad, Mikołaj Wewiór

Warszawa, 2023

# Spis treści

<b>1. CZĘŚĆ PROJEKTOWA</b>	<b>2</b>
1.1. Sprawdzenie poprawności punktów pracy	2
1.2. Wyznaczenie symulacyjne odpowiedzi skokowej procesu	3
1.2.1. Odpowiedzi skokowe	3
1.2.2. Charakterystyka statyczna $y(u)$	3
1.2.3. Wzmocnienie statyczne	5
1.3. Przekształcenie otrzymanej odpowiedzi na odpowiedź skokową dla algorytmu DMC	5
1.4. Program do symulacji cyfrowego algorytmu PID oraz algorytmu DMC	6
1.4.1. PID	6
1.4.2. DMC	8
1.5. Dobór nastaw algorytmów regulacji metodą eksperymentalną	10
1.5.1. PID	10
1.5.2. DMC	12
1.6. Dobór nastaw algorytmów regulacji w wyniku optymalizacji	15
1.6.1. PID	15
1.6.2. DMC	16
<b>2. CZĘŚĆ LABORATORYJNA</b>	<b>18</b>
2.1. Sprawdzenie możliwości sterowania i pomiaru stanowiska	18
2.2. Wyznaczenie odpowiedzi skokowej procesu	18
2.3. Przekształcenie odpowiedzi skokowej	18
2.3.1. Odpowiedź skokowe wykorzystywana w algorytmie DMC	18
2.3.2. Aproksymacja odpowiedzi skokowej	18
2.3.3. Optymalizacja parametrów modelu	18
2.3.4. Porównanie aproksymowanego modelu z obiektem	18
2.4. Regulacja PID oraz DMC	18
2.5. Dobór nastawów regulatorów PID oraz DMC metodą eksperymentalną	18

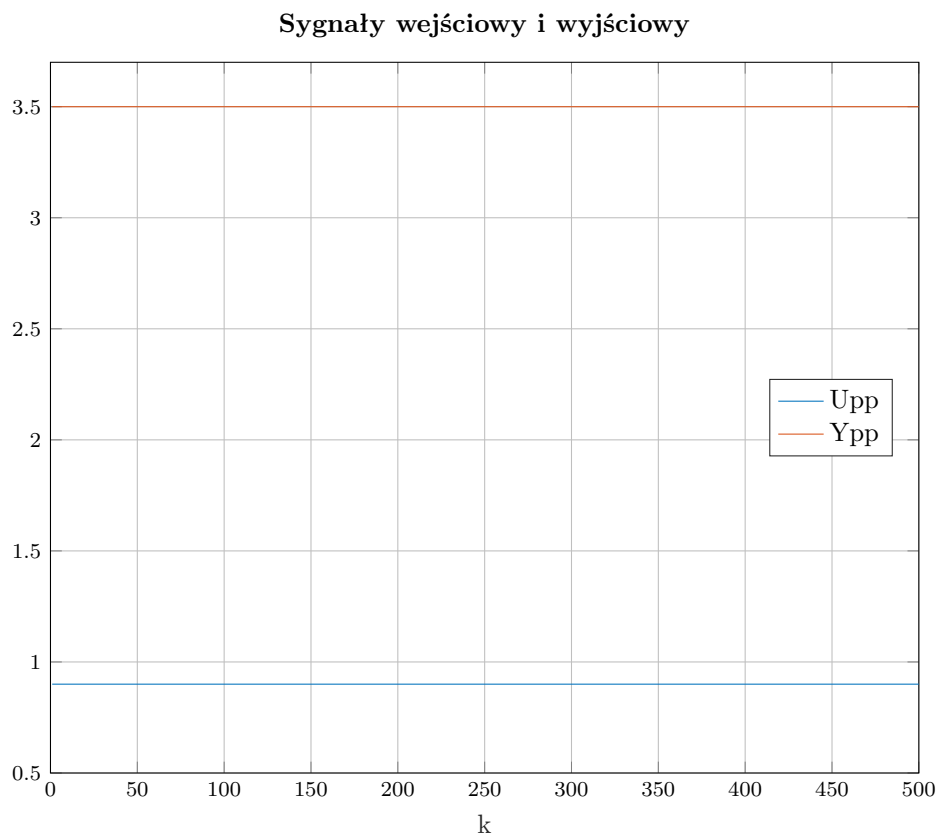
# 1. CZĘŚĆ PROJEKTOWA

## 1.1. Sprawdzenie poprawności punktów pracy

Sprawdzenie poprawności punktów pracy polegało na obserwacji wyjścia symulowanego obiektu projektowego *12y\_p1* przy niezmiennym sterowaniu *U<sub>pp</sub>*. Punkty pracy *U<sub>pp</sub>* oraz *Y<sub>pp</sub>* odczytaliśmy po wywołaniu w terminalu Matlab: *symulacja\_obiektu12y\_p1*. W tym celu skorzystano z funkcji:

```
n = 500;
Upp = 0.9; u(1:n) = Upp;
Ypp = 3.5; y(1:12) = Ypp;

for k = 13:n
    y(k) = symulacja_obiektu12y_p1(u(k-10), u(k-11), ...
        y(k-1), y(k-2));
end
```



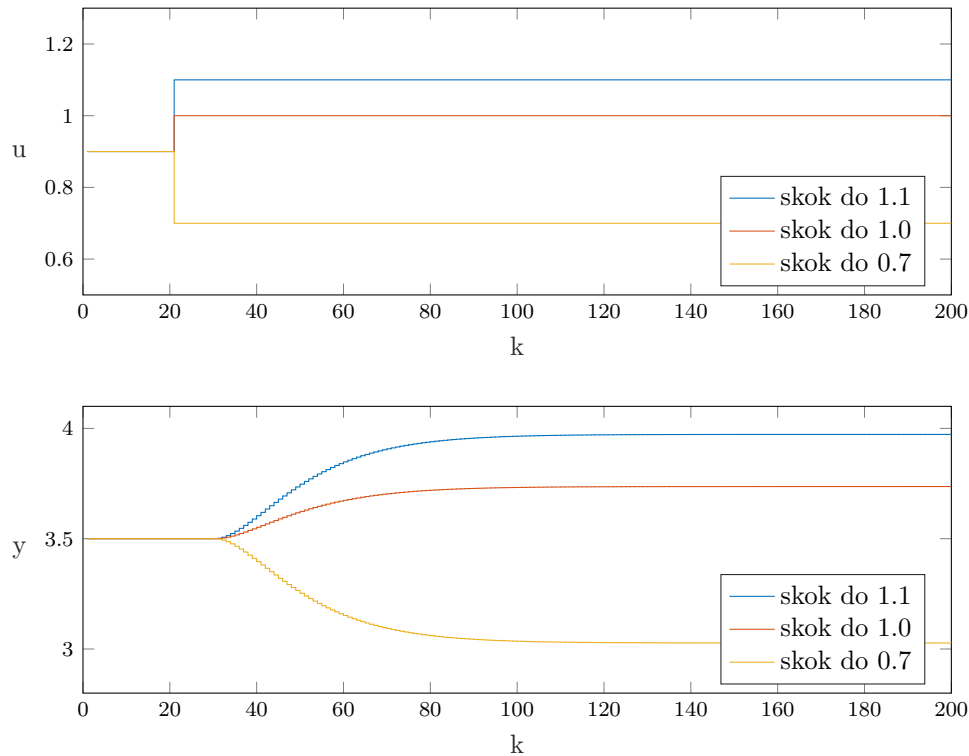
Rys. 1. Punkty pracy *U<sub>pp</sub>* (sygnał wejściowy) i *Y<sub>pp</sub>* (sygnał wyjściowy)

Jak widać, przy niezmiennym sterowaniu *U<sub>pp</sub>* równym 0.9, wyjście obiektu nie zmienia się, pozostając równe 3.5 potwierdzając tym samym, że punkt pracy jest stabilny.

## 1.2. Wyznaczenie symulacyjne odpowiedzi skokowej procesu

### 1.2.1. Odpowiedzi skokowe

Należało symulacyjnie wyznaczyć odpowiedzi skokowe procesu dla kilku zmian sygnału sterującego. W przypadku tego projektu dokonano zmian z punktu pracy  $U_{pp} = 0.9$  do kolejnych wartości sterowania: 1.1, 1.0 oraz 0.7. Następnie obserwowano zmianę na wyjściu  $y$  obiektu.



Rys. 2. Zmiana sygnału sterującego oraz odpowiedź skokowa obiektu

Na pierwszy rzut oka wydaje się, że właściwości statyczne i dynamiczne są liniowe. Poniżej przedstawiono implementację:

```
delay = 20;  
Uk = 1.1      % w kolejnych iteracjach 1.0 oraz 0.7  
u(1:delay-1) = Upp;  
u(delay:n) = Uk;  
y(1:delay-1) = Ypp;  
  
for k = delay:n  
    y(k) = symulacja_obiektu12y_p1(u(k-10), u(k-11), ...  
        y(k-1), y(k-2));  
end
```

### 1.2.2. Charakterystyka statyczna $y(u)$

Do wyznaczenia charakterystyki statycznej zostało zebrane wiele różnych punktów pracy obiektu dla całej dziedziny sterowania. Następnie policzono wzmocnienie dla każdego z nich oraz naniesiono je na wykres.

```
x = 0:0.01:0.4;
```

```

Ustat = zeros(size(x, 2), 1);
Ystat = zeros(size(x, 2), 1);
delay = 20;

for i = 1:size(x, 2)

    Uk = 0.7 + x(i);
    u(1:delay-1) = Upp;
    u(delay:n) = Uk;
    y(1:delay-1) = Ypp;

    for k = delay:n
        y(k)=symulacja_obiektu12y_p1(u(k-10), u(k-11), ...
            y(k-1), y(k-2));
    end

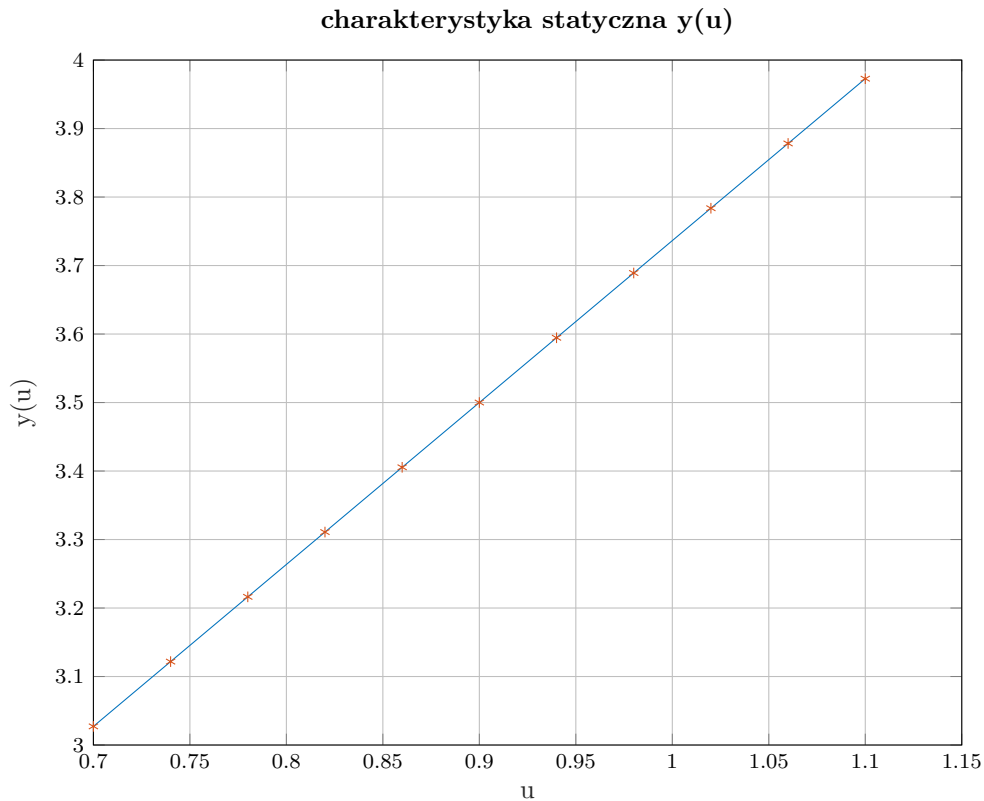
    Ustat(i) = u(n); Ystat(i) = y(n);

end

Kstat = zeros(size(x, 2), 1);

for k = 1:size(x, 2)
    if Ystat(k) ~= Ypp
        Kstat(k) = (Ystat(k) - Ypp) / (Ustat(k) - Upp);
    end
end

```



Rys. 3. Charakterystyka statyczna obiektu

Jak widać, charakterystyka statyczna  $y(u)$  dla obiektu symulacyjnego jest liniowa.

### 1.2.3. Wzmocnienie statyczne

Z charakterystyki statycznej (Rys. 3.) można zauważyć, że właściwości obiektu są liniowe. Wzmocnieniem statycznym jest współczynnik kierunkowy prostej, na którym znajdują się punkty pracy obiektu. Dla powyższego obiektu wzmocnienie  $K$  wynosi 2.3639.

```
wzmocnienie_statyczne = median(Kstat);
```

## 1.3. Przekształcenie otrzymanej odpowiedzi na odpowiedź skokową dla algorytmu DMC

Aby z otrzymanej odpowiedzi uzyskać odpowiedź skokową dla algorytmu DMC należy pozbyć się offsetu. Można to zrobić w następujący sposób:

$$s_k = \frac{y(k) - Y_{pp}}{U_k - U_{pp}}$$

gdzie  $k = 1, \dots, D$  oraz

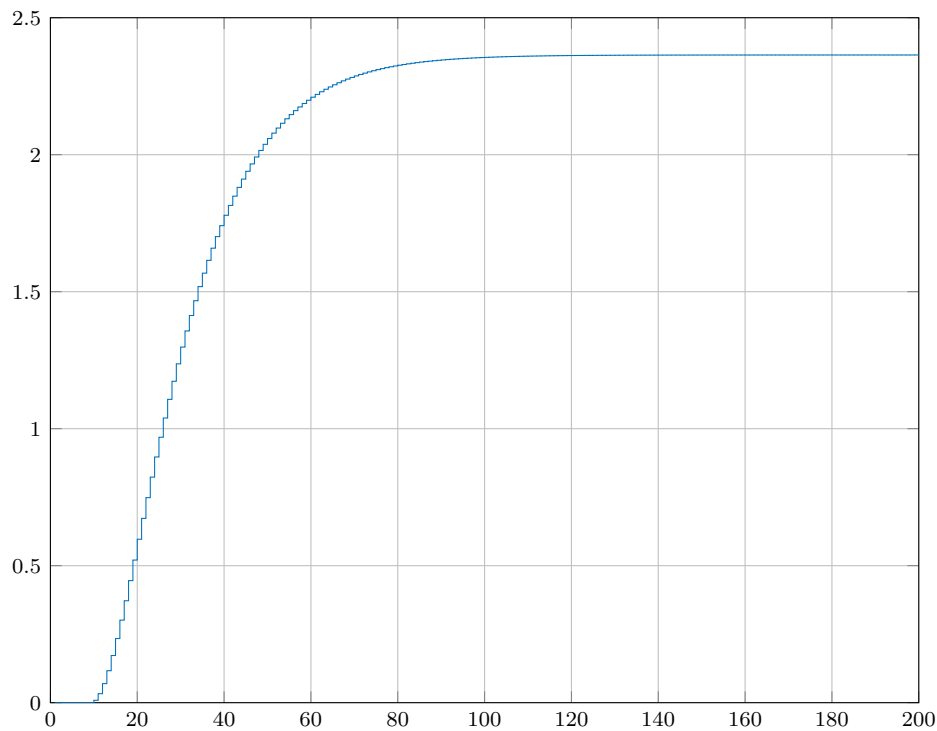
$$u(k) = \begin{cases} U_{pp} & \text{dla } k < 0 \\ U_k & \text{dla } k \geq 0 \end{cases}$$

Implementacja w MATLABIE:

```

for k = kp : kk + kp
    y(k) = symulacja_obiektu12y_p1(u(k-10), u(k-11), ...
        y(k-1), y(k-2));
end
s(1:kk) = (y(kp+1:end)-ones(1, kk)*Ypp)/(Uk-Upp);

```



Rys. 4. odpowiedź obiektu na skok jednostkowy

## 1.4. Program do symulacji cyfrowego algorytmu PID oraz algorytmu DMC

### 1.4.1. 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:

$$r_0 = K_r \left(1 + \frac{T_p}{2T_i} + \frac{T_d}{T_p}\right), \quad r_1 = K_r \left(\frac{T_p}{2T_i} - 2\frac{T_d}{T_p} - 1\right), \quad r_2 = \frac{K_r T_d}{T_p}$$

W badanym obiekcie występuje opóźnienie równe jedenastu okresom próbkowania, a więc symulacje można rozpocząć od chwili  $kp = 12$ .

Implementacja w MATLABIE:

```

function E = PID(K_pid, Ti, Td)
    % inicjalizacja
    Upp = 0.9;      Ypp = 3.5;
    Umin = 0.7;     Umax = 1.1;
    dumax = 0.05;   % maksymalny przyrost sygnału sterowania
    kp = 12;        % chwila początkowa symulacji
    n = 400;        % czas trwania symulacji
    T = 0.5;        % okres próbkowania

    u(1:kp-1) = Upp;      y(1:kp-1) = Ypp;
    yzad(1:kp-1) = Ypp;   yzad(kp:n) = Ypp + 0.2;
    e(1:kp-1) = 0;

    % 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:n
        % symulacja wyjścia obiektu
        y(k) = symulacja_obiektu12y_p1(u(k-10), u(k-11), ...
                                         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);

        du = u(k) - u(k-1);

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

        % ograniczenia szybkości zmian sygnału sterującego
        if du > dumax
            u(k) = u(k-1) + dumax;
        elseif du < -dumax
            u(k) = u(k-1) - dumax;
        end
    end

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

```



end

### 1.4.2. 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  $j = 1, \dots, D$ , 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, s)
% inicjalizacja
Upp = 0.9;      Ypp = 3.5;
Umin = 0.7;     Umax = 1.1;
dumax = 0.05;   % maksymalny przyrost sygnału sterowania
Uk = 1;         % skok
n = 500;        % długość trwania symulacji
kp = 12;
kk = N+D-1;

u(1:kp-1) = Upp;      y(1:kp-1) = Ypp;
yzad(1:kp-1) = Ypp;   yzad(kp:n) = Ypp + 0.2;
e(1:kp-1) = 0;

% odpowiedź skokowa // przekazany zostaje wektor s
%                      // z poprzedniego podpunktu

% 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
        Mp(i, j) = s(i+j) - s(j);
    end
end
```

```

% wektor 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;

for k = kp : n

    % symulacja wyjścia obiektu
    y(k) = symulacja_obiektu12y_p1(u(k-10), u(k-11), ...
        y(k-1), y(k-2));

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

    % obliczenie sumy: k_j_u * delta_u(k-j)
    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 w chwili k
    % (du(k|k))
    dukk = Ke * e - elem;

    % ograniczenia przyrostu sterowania
    if dukk > dumax
        dukk = dumax;
    elseif dukk < -dumax
        dukk = -dumax;
    end

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

    % ograniczenia maksymalnych wartości sterowania
    if u(k) > Umax
        u(k) = Umax;
    elseif u(k) < Umin

```

```

        u(k) = Umin;
    end

end

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

end

```

## 1.5. Dobór nastaw algorytmów regulacji metodą eksperymentalną

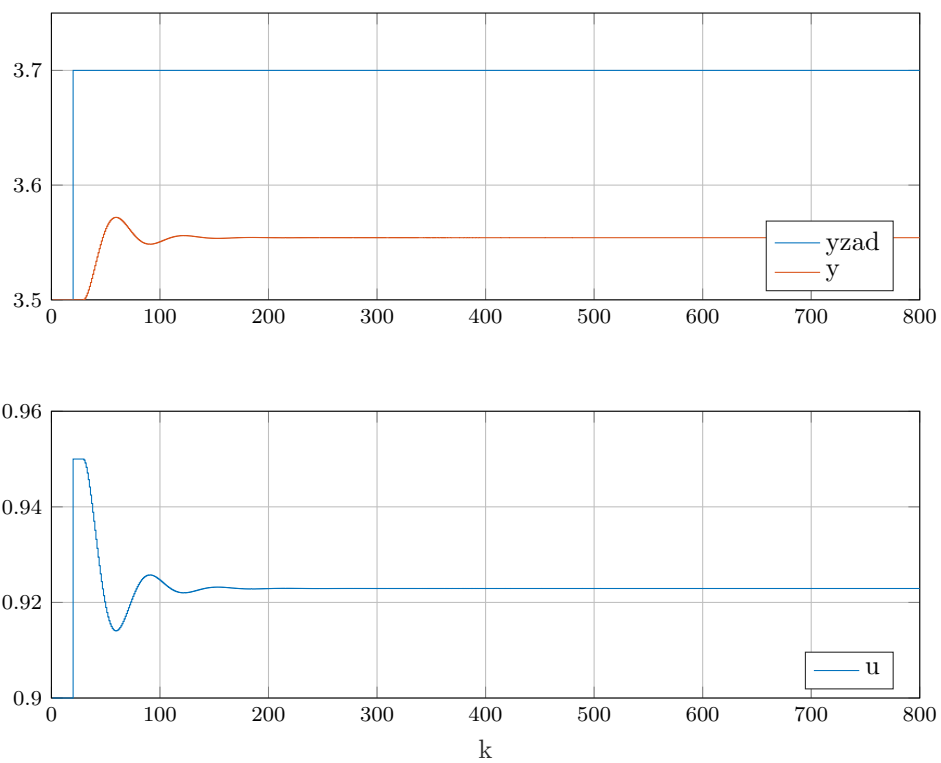
### 1.5.1. PID

Kolejne kroki doboru nastaw algorytmu PID metodą eksperymentalną:

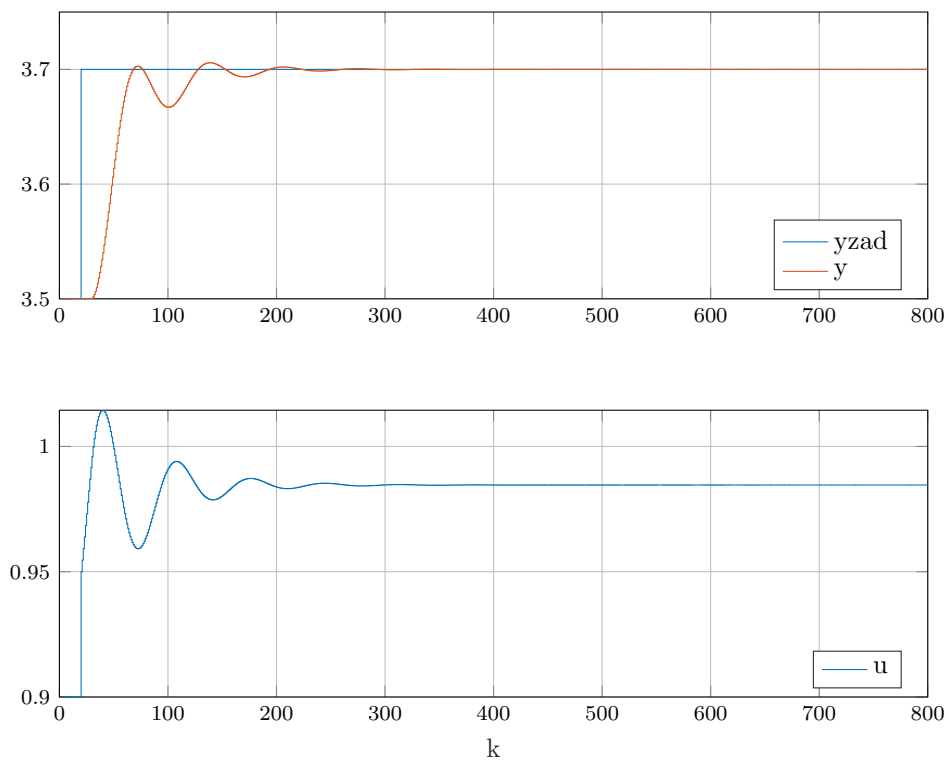
1. Analiza regulatora P - wyzerowanie członu całkującego (tzn. ustawienie parametru  $T_i$  na bardzo dużą wartość) i różniczkującego (tzn. ustawienie parametru  $T_d$  na zero). Zwiększanie parametru wzmocnienia  $K$  do momentu jak najmniejszego uchybu ustalonego i pojawienia się na wyjściu oscylacji.
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:

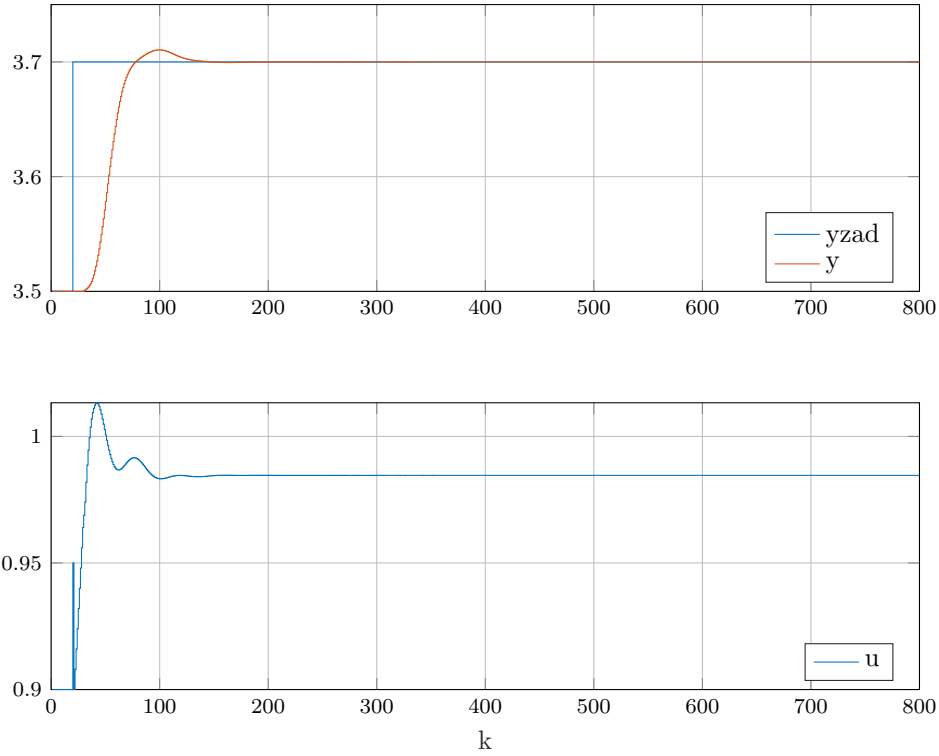
$$K = 0.8, T_i = 10, T_d = 4$$



Rys. 5. Doprowadzanie regulatora P pojawienia się pierwszych oscylacji ( $K = 0.5$ ).  
Błąd ilościowy:  $E = 16.8957$



Rys. 6. Dostrojenie regulatora PI ( $K = 0.6, T_i = 13$ ).  
Błąd ilościowy:  $E = 0.9767$



Rys. 7. Dostrojenie regulatora PID ( $K = 0.8, T_i = 10, T_d = 4$ ).  
Błąd ilościowy:  $E = 1.1265$

Najmniejszy błąd wskaźnika regulacji miał regulator PI o nastawach  $K = 0.6$  i  $T_i = 13$ . Jednak jakościowo najlepszy przebieg ma regulator PID o nastawach  $K = 0.8, T_i = 10, T_d = 4$ . nie ma tutaj znacznego przeregulowania i oscylacji. Sygnał sterujący również nie zmienia się gwałtownie. Wyjątkiem jest "szpilka" na początku sygnału sterującego - jest ona spowodowana członem różniczkującym, który zaraz po rozpoczęciu działania reaguje na bardzo duży uchyb.

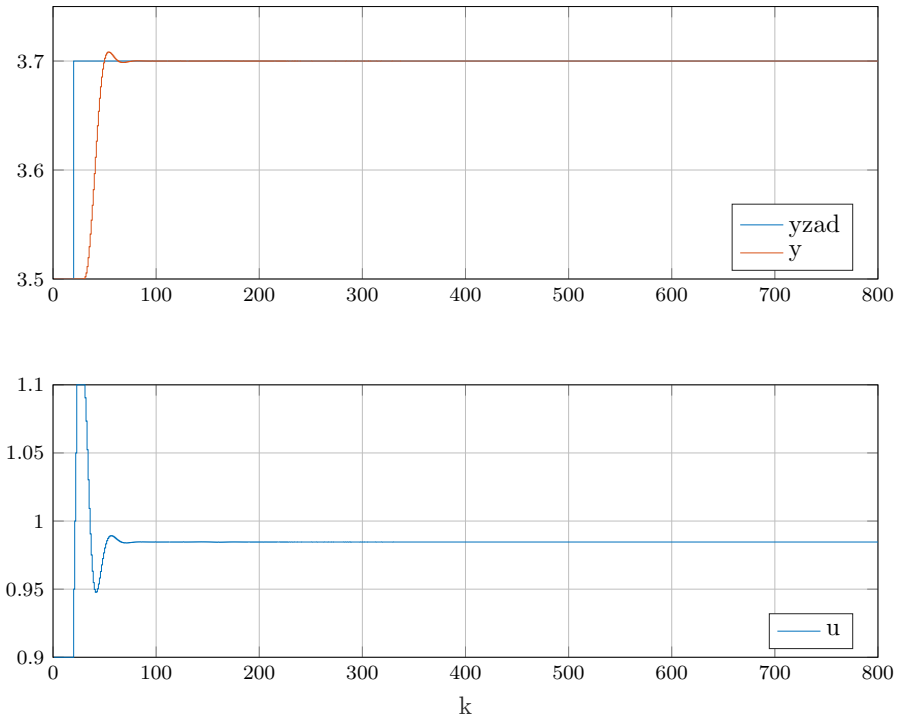
### 1.5.2. DMC

Kolejne kroki doboru nastaw algorytmu DMC metodą eksperymentalną:

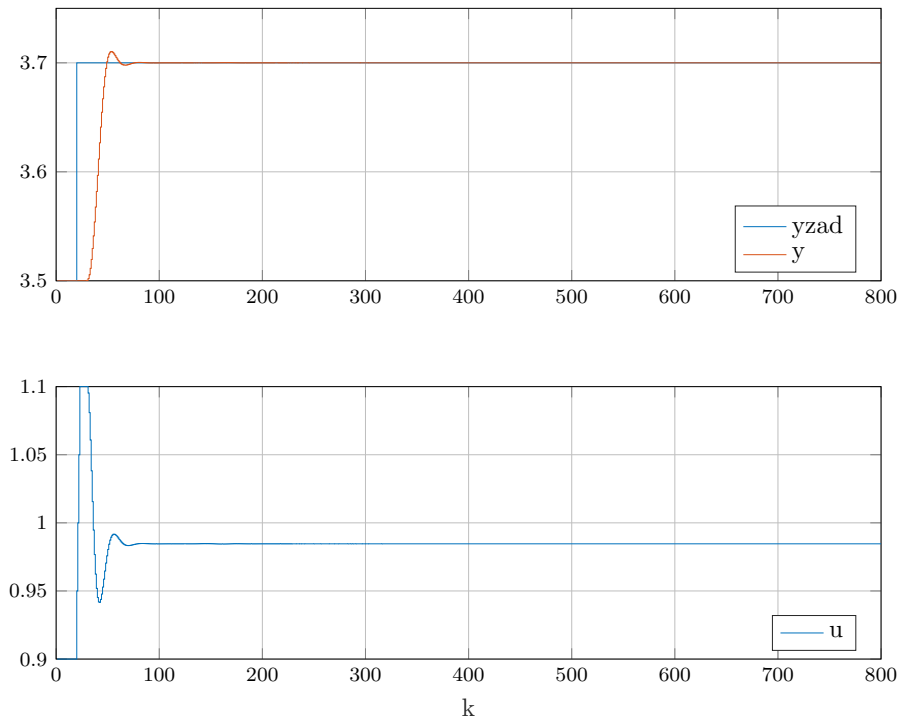
1. Ustalenie długości horyzontu dynamiki z odpowiedzi skokowej obiektu - moment gdy sygnał na wyjściu się nie zmienia lub zmienia się nieznacznie. W przypadku naszego obiektu przyjęliśmy  $D = 130$
2. Ustawienie wartości horyzontu predykcji i horyzontu sterowania równych horyzontowi dynamiki. Ustawienie parametru lambda na 1.
3. Stopniowe zmniejszanie wartości horyzontu predykcji i horyzontu sterowania.
4. Zmniejszenie wartości horyzontu sterowania.
5. Zmiana wartości lambda - 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:

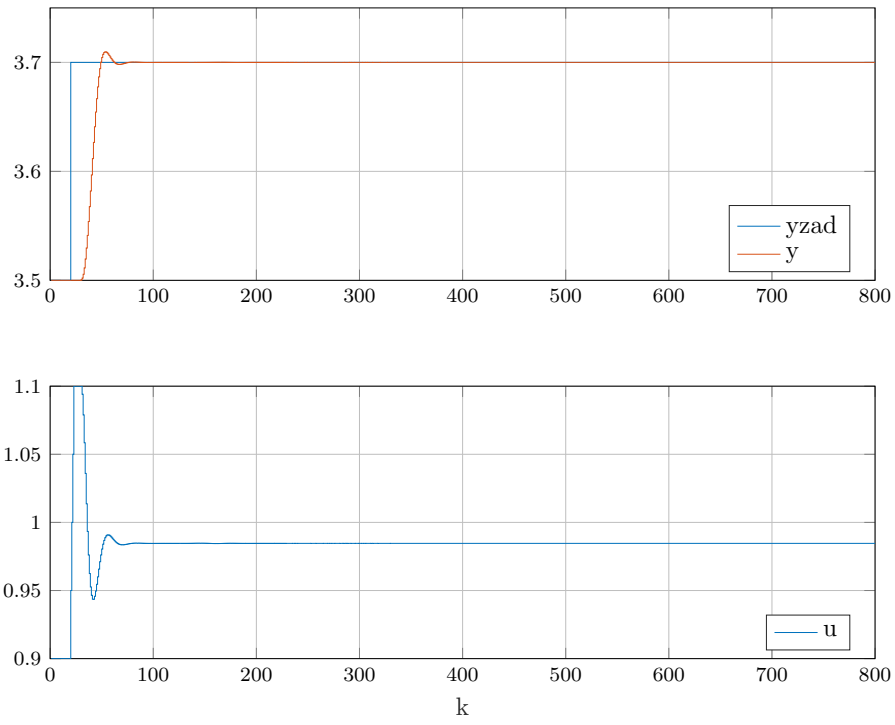
$$N = 40, N_u = 5, \lambda = 1$$



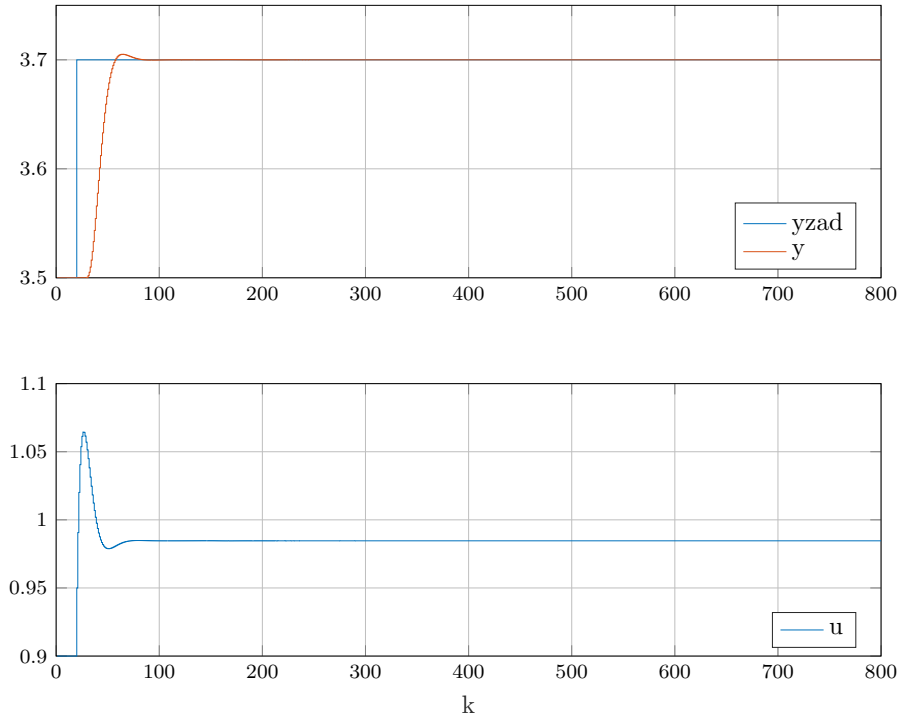
Rys. 8.  $N = N_u = D = 130$ ,  $\lambda = 1$   
 Błąd ilościowy:  $E = 0.7239$



Rys. 9.  $N = N_u = 20$ ,  $\lambda = 1$   
 Błąd ilościowy:  $E = 0.7238$



Rys. 10.  $N = 20, N_u = 5, \lambda = 1$   
 Błąd ilościowy:  $E = 0.7238$



Rys. 11.  $N = 40, N_u = 5, \lambda = 10$   
 Błąd ilościowy:  $E = 0.7751$

Widać, że dla horyzontu predykcji oraz horyzontu sterowania można stosunkowo bezkarnie znacznie zmniejszyć ich długość - wpływ na kształt sygnałów oraz błąd jest znikomy. Po zwiększeniu lambdy musieliśmy zwiększyć horyzont predykcji. Po tym zabiegu błąd wzrósł nieznacznie, ale kształt obu sygnałów znacznie się poprawił - sygnał sterowania nie przyjmuje maksymalnych wartości, a wyjście jest delikatnie wolniejsze.

## 1.6. Dobór nastaw algorytmów regulacji w wyniku optymalizacji

Nastawy algorytmów regulacji można również odnaleźć metodą optymalizacji funkcji błędu, która zależna jest od nastawów regulatorów. W programach MATLAB w takim celu można wykorzystać funkcję `fmincon`. Poniżej przedstawiono implementację funkcji dla obu regulatorów.

Funkcja `fmincon` przyjmuje handler funkcji symulującej regulację PID, która ostatecznie zwraca wskaźnik jakości regulacji dla wybranych nastawów. Nastawy regulatora dobierane są przez funkcję `fmincon`, która zwraca zatrzymuje się, kiedy osiągnie najmniejszą funkcję błędu.

### 1.6.1. PID

```
%% Optymalizacja nastaw PID

% fmincon
objective = @(x) PID(x(1), x(2), x(3));
x_initial = [0.8; 10; 4];
lb = [0.4, 5, 0];
ub = [10, 1000000, 10];

nonlcon = [];

options = optimoptions('fmincon', 'Display', 'iter');
[x_optimal, fval] = fmincon(objective, x_initial, ...
    [], [], [], [], lb, ub, nonlcon, options);

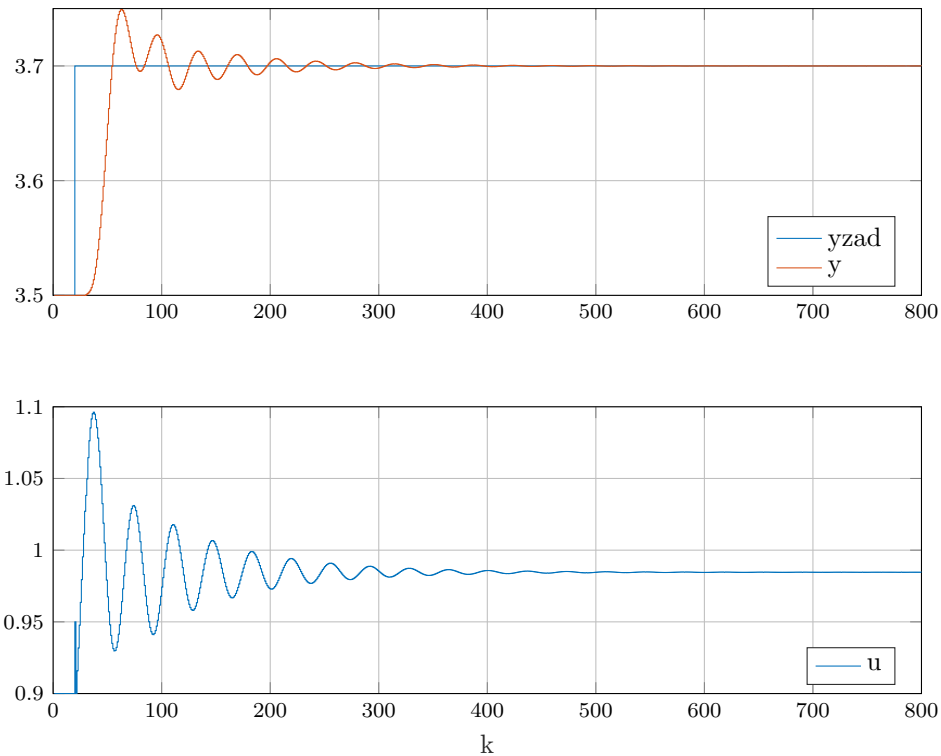
function E = PID(K_pid, Ti, Td)
    IMPLEMENTACJA REGULATORA PID
end
```

$$K = 1.22, T_i = 7.67, T_d = 4.07$$

Przy nastawach optymalnych, wskaźnik jakości regulacji wynosi:

$$E = 0.9713$$





Rys. 12. Regulacja obiektu przy optymalnych nastawach

Pomimo nazwy "optymalne", można mieć wrażenie, że nastawy wyliczone przez funkcję `fmincon` wcale nie są najlepsze. Faktycznie funkcja błędu ma wartość najmniejszą z testowanych. Jednak kształt sygnałów zarówno wyjściowego, jak i sterowania nie są najlepsze. Obiekt ustala się dosyć długo - dopiero w chwili 500-setnej, co stanowi prawie trzykrotność w porównaniu z eksperymentalną metodą, gdzie obiekt już był ustalony w okolicach 150 chwili.

### 1.6.2. DMC

Implementacja wygląda podobnie w przypadku regulatora DMC.

```
%% Optymalizacja nastaw DMC
% fmincon
objective = @(x) DMC(x(1), x(2), x(3), x(4));
x_initial = [130, 130, 130, 1];
lb = [70, 10, 1, 0.1];
ub = [150, 150, 150, 30];

nonlcon = [];

options = optimoptions('fmincon', 'Display', 'iter');
[x_optimal, fval] = fmincon(objective, x_initial, ...
    [], [], [], [], lb, ub, nonlcon, options);

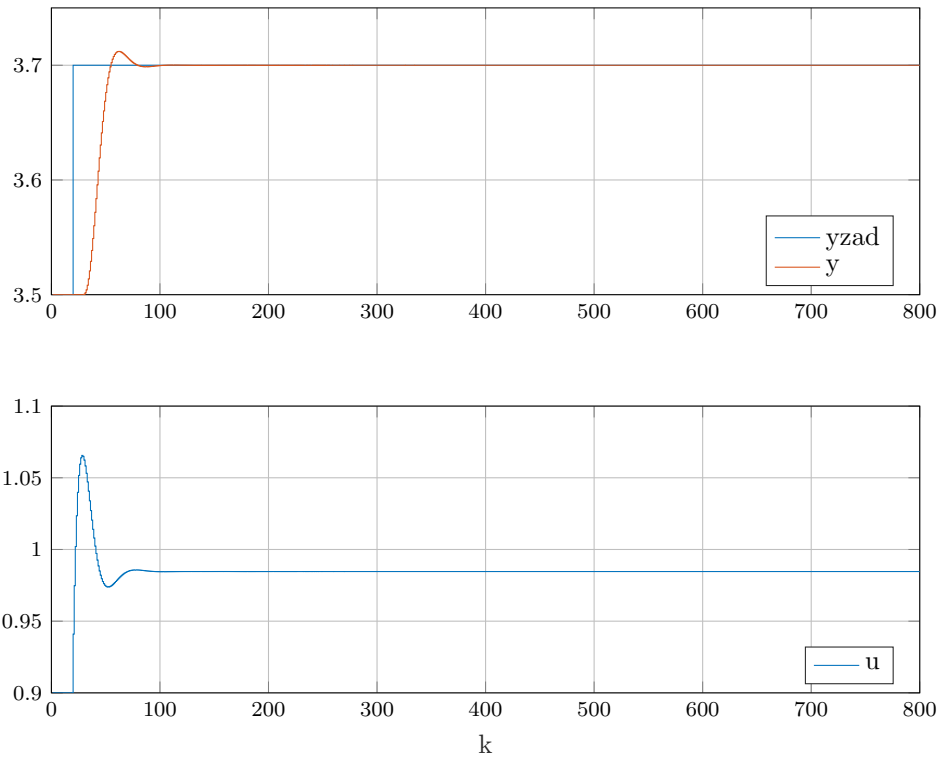
function E = DMC(D, N, Nu, lambda)
    %%%
    IMPLEMENTACJA REGULATORA DMC
    %%%
end
```

Po uruchomieniu skryptu, otrzymuje się optymalne nastawy:

$$D = 141, N = 116, N_u = 129, \lambda = 20$$

Przy nastawach optymalnych, wskaźnik jakości regulacji wynosi:

$$E = 0.7867$$



Rys. 13. Regulacja obiektu przy optymalnych nastawach

Tutaj ciekawa sytuacja, ponieważ nastawy otrzymane z **fmincona** mają gorsze skutki niż te otrzymane eksperymentalnie przez nas - nieznacznie ale jednak.

## 2. CZĘŚĆ LABORATORYJNA

2.1. Sprawdzenie możliwości sterowania i pomiaru stanowiska

2.2. Wyznaczenie odpowiedzi skokowej procesu

2.3. Przekształcenie odpowiedzi skokowej

2.3.1. Odpowiedź skokowa wykorzystywana w algorytmie DMC

2.3.2. Aproksymacja odpowiedzi skokowej

2.3.3. Optymalizacja parametrów modelu

2.3.4. Porównanie aproksymowanego modelu z obiektem

2.4. Regulacja PID oraz DMC

2.5. Dobór nastawów regulatorów PID oraz DMC metodą eksperymentalną