Wydział Elektroniki i Technik Informacyjnych
Politechnika Warszawska

Projektowanie układów sterowania (projekt grupowy)

Sprawozdanie z projektu i ćwiczenia laboratoryjnego nr 1, zadanie nr 4

Mateusz Koroś, Ksawery Pasikowski, Mateusz Morusiewicz

Warszawa, 2017

Spis treści

1. Zadanie 1.

Poprawność wartości została potwierdzona poprzez sprawdzenie, czy obiekt, będący w punkcie pracy, pozostanie w nim, jeśli zachowamy stałe sterowanie, równe U_{pp}, zostało to wykonane za pomocą komendy:

```
y = symulacja_obiektu4Y(Upp, Upp, Ypp, Ypp)
```

Otrzymane $y = 0,8 = Y_{pp}$, co potwierdza wartości U_{pp} oraz Y_{pp}.

2. Zadanie 2.

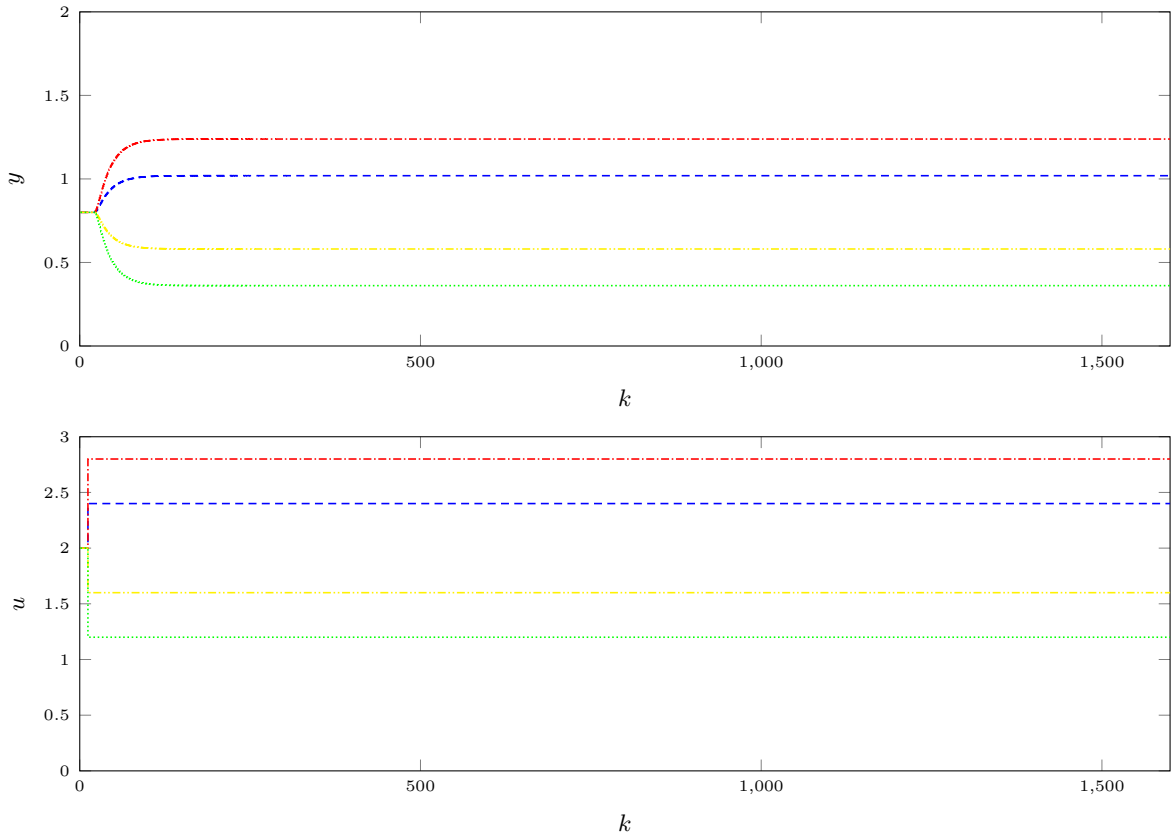
Odpowiedzi skokowe zostały wyznaczone poprzez zmianę sterowania z punktu pracy do wartości:

— $u = 2.4$
— $u = 2.8$
— $u = 1.6$
— $u = 1.2$

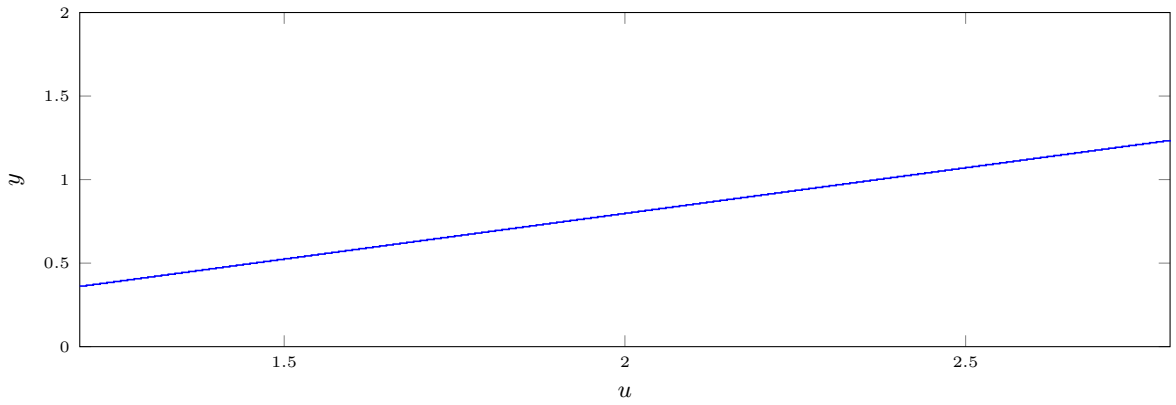
Charakterystykę statyczną procesu widać na wykresie 2.2

Właściwości statyczne i dynamiczne są w przybliżeniu liniowe, stąd wzmocnienie statyczne zostało policzone za pomocą wzoru:

$$K = \frac{\Delta y}{\Delta u} = 0,5484$$



Rys. 2.1. Odpowiedzi skokowe dla różnych wartości skoku



Rys. 2.2. Charakterystyka statyczna

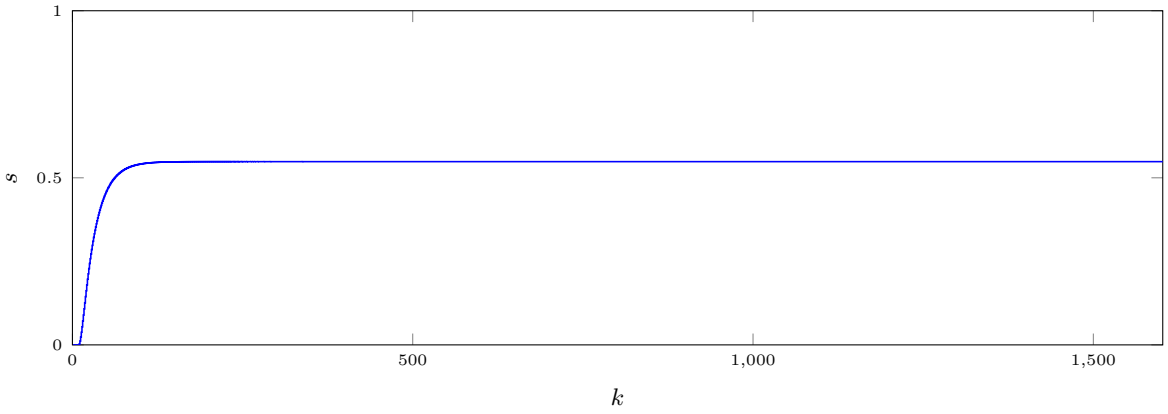
3. Zadanie 3.

Odpowiedź skokowa została zebrana poprzez zmianę sterowania z $u = U_{pp} = 2$ na $u = 2.8$ i zebranie odpowiedzi obiektu. Ponieważ odpowiedź skokowa, używana w algorytmie DMC jest odpowiedzią na skok jednostkowy z sygnału sterowania równego zero, należało ją znormalizować zgodnie z poniższym wzorem:

$$s = \frac{s_0 - Y_{pp}}{\Delta u}$$

gdzie:
— s - znormalizowana odpowiedź skokowa
— s_0 - zebrana odpowiedź skokowa
— Y_{pp} - wyjście układu w punkcie pracy
— Δu - wartość skoku sterowania

Otrzymana odpowiedź skokowa po normalizacji wygląda następująco:



Rys. 3.1. Odpowiedź skokowa

4. Zadanie 4.

4.1. PID

Program, realizujący symulację algorytmu PID wygląda następująco:

```
Kp = 3 ;
Ti = 10 ;
Td = 3.2 ;

r2 = (Kp * Td) / Tp ;
r1 = Kp * ( (Tp/(2*Ti)) - 2*(Td/Tp) - 1 ) ;
r0 = Kp * ( 1 + Tp/(2*Ti) + Td/Tp ) ;

% warunki poczatkowe
u(1:11) = Upp ;
U(1:11) = Upp ;
y(1:11) = Ypp ;
y2(1:11) = Ypp ;
e(1:11) = 0 ;
delta_u = 0;
index = 1;
yzads = [1.05 0.8 1.1 0.9];
yzad = yzads(index);   %skok wartosci zadanej
yzad2 = yzad - Ypp;
yzadVec(1:Kk) = yzad;

% glowna petla symulacji
for k = 12 : Kk
    if mod(k,400) == 0
        index = index + 1;
        if index > length(yzads)
            index = length(yzads);
        end
        yzad = yzads(index);
        yzad2 = yzad - Ypp;
    end
    yzadVec(k) = yzad;

    y(k) = symulacja_obiektu4Y(U(k-10), U(k-11), y(k-1), y(k-2)) ;

    y2(k) = y(k) - Ypp;
    e(k) = yzad2 - y2(k) ;

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

```
    delta_u = u(k) - u(k-1);

    if delta_u > dU_max
        delta_u = dU_max;
    elseif delta_u < -dU_max
        delta_u = -dU_max;
    end

    u(k) = u(k-1) + delta_u;

    if u(k) > U_max - Upp
        u(k) = U_max - Upp;
    elseif u(k) < U_min - Upp
        u(k) = U_min - Upp;
    end

     U(k) = u(k) + Upp;
end
```

4.2. DMC

Natomiast realizacja DMC w wersji analitycznej:

```
s = policz_odp_skok();

D = policzHorDynamiki(s) ;   %horyzont dynamiki
% N = D ;   %horyzont predykcji
% Nu = D ;  %horyzont sterowania
% lambda = 20;
yzads = [1.05 0.8 1.1 0.9];
index = 1;
yzad = yzads(index);   %skok wartosci zadanej
yzadVec(1:Kk) = yzad;

%sygnal sterujacy
u = Upp + zeros(1,N) ;
U = Upp + zeros(1,N);
%uchyb
e = zeros(1,N) ;
%wyjscie ukladu
y = zeros(1,Kk) + Ypp ;

%obliczanie odpowiedzi skokowej
du = (zeros(1,D-1))' ;

M = zeros(N, Nu) ;
for i = 1:N
    for j = 1:Nu
        if (i-j+1 > 0)
            M(i,j) = s(i-j+1) ;
        else
```

```
            M(i,j) = 0 ;
        end
    end
end

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

K = (M'*M + lambda*eye(Nu))^-1 * M' ;

%liczenie ke
ke = 0;
for i = 1:N
    ke = ke + K(1, i);
end

kju = K(1,:)*Mp;
y2 = zeros(Kk, 1);

for k = 12:Kk

    if mod(k,400) == 0
        index = index + 1;
        if index > length(yzads)
            index = length(yzads);
        end
        yzad = yzads(index);
    end
    yzadVec(k) = yzad;

    y(k) = symulacja_obiektu4Y(U(k-10), U(k-11), y(k-1), y(k-2)) ;

    sum = 0;    %suma potrzebna do obliczenia skladowej swobodnej
    for j = 1:D-1
        if(k-j > 0)
            sum = sum + kju(j)*du(k-j);
            %w innym przypadku du = 0 wiec sum sie nie zmienia
        end
    end
    y2(k) = y(k) - Ypp;
    yzad2 = yzad - Ypp;
    du(k) = ke * (yzad2-y2(k)) - sum ;
```

```
    % --- sprawdzenie, czy przyrost znajduje sie w ograniczeniach ---
    if du(k) > dU_max
        du(k) = dU_max;
    elseif du(k) < -dU_max
        du(k) = -dU_max;
    end

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

    if u(k) > U_max - Upp
        u(k) = U_max - Upp;
    elseif u(k) < U_min - Upp
        u(k) = U_min - Upp;
    end

    U(k) = u(k) + Upp;

end
```

Ograniczenia zarówno wartości jak i szybkości wzrostu sygnału sterującego w obu regulatorach zostały uwzględnione poprzez rzutowanie po obliczeniu sterowania przez algorytm, natomiast przesunięcie do punktu pracy zostało wykonane za pomocą odjęcia wartości Y_{pp} przed algorytmem regulacji i dodania Upp do obliczonego sterowania.

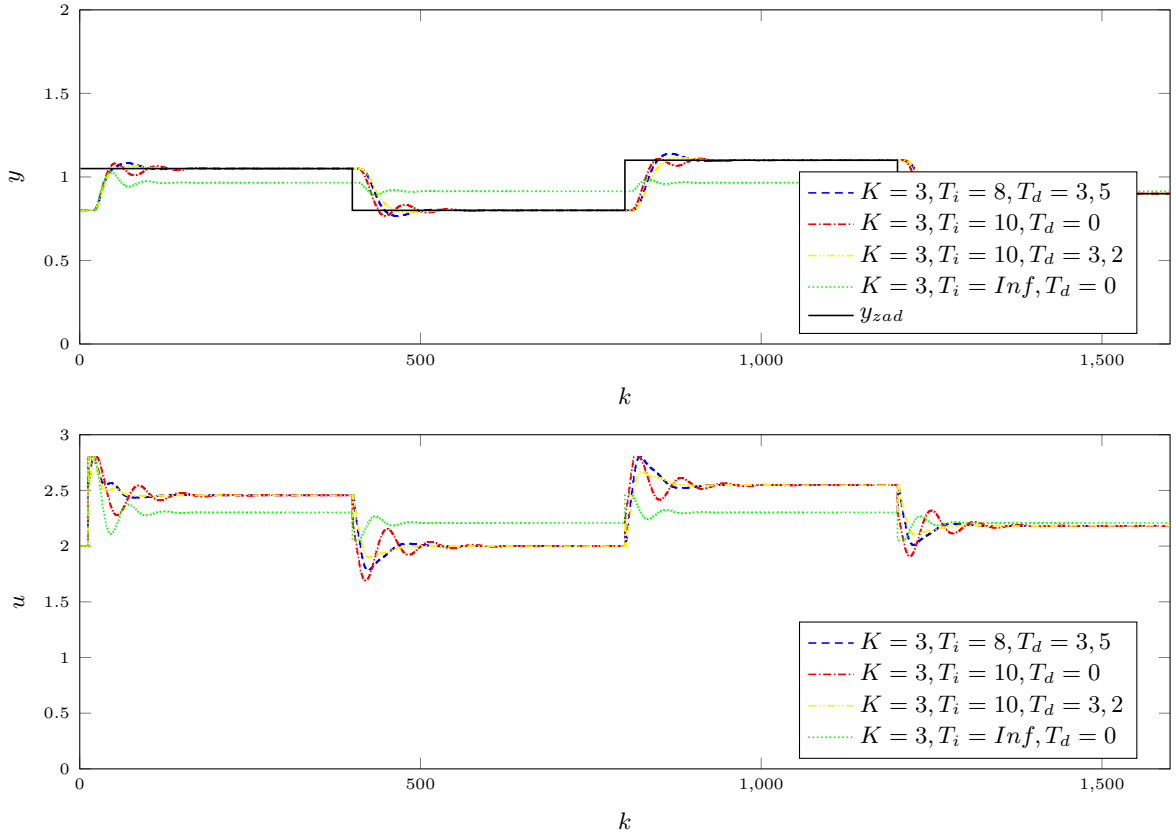
5. Zadanie 5.

5.1. PID

Nastawy regulatora PID zostały dobrane poprzez dodawanie kolejnych członów do regulatora P, co prowadziło do polepszenia jakości regulacji. Jakość regulacji była oceniana jakościowo – na podstawie wykresów – oraz ilościowo – poprzez wskaźnik regulacji, obliczany wzorem:

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

Wskaźnik jakości regulacji dla każdych nastaw wynosił odpowiednio:
— $K_p = 3; T_i = \infty; T_d = 0 \Rightarrow E = 17,2257$
— $K_p = 3; T_i = 10; T_d = 0 \Rightarrow E = 6,0947$
— $K_p = 3; T_i = 8; T_d = 3,5 \Rightarrow E = 6,9931$
— $K_p = 3; T_i = 10; T_d = 3,2 \Rightarrow E = 7,3477$
Jak widać, regulator PI cechował się najmniejszym błędem, jednak regulator PID prowadził do mniejszych oscylacji podczas dążenia do wartości zadanej.



Rys. 5.1. Regulacja PID dla różnych nastaw

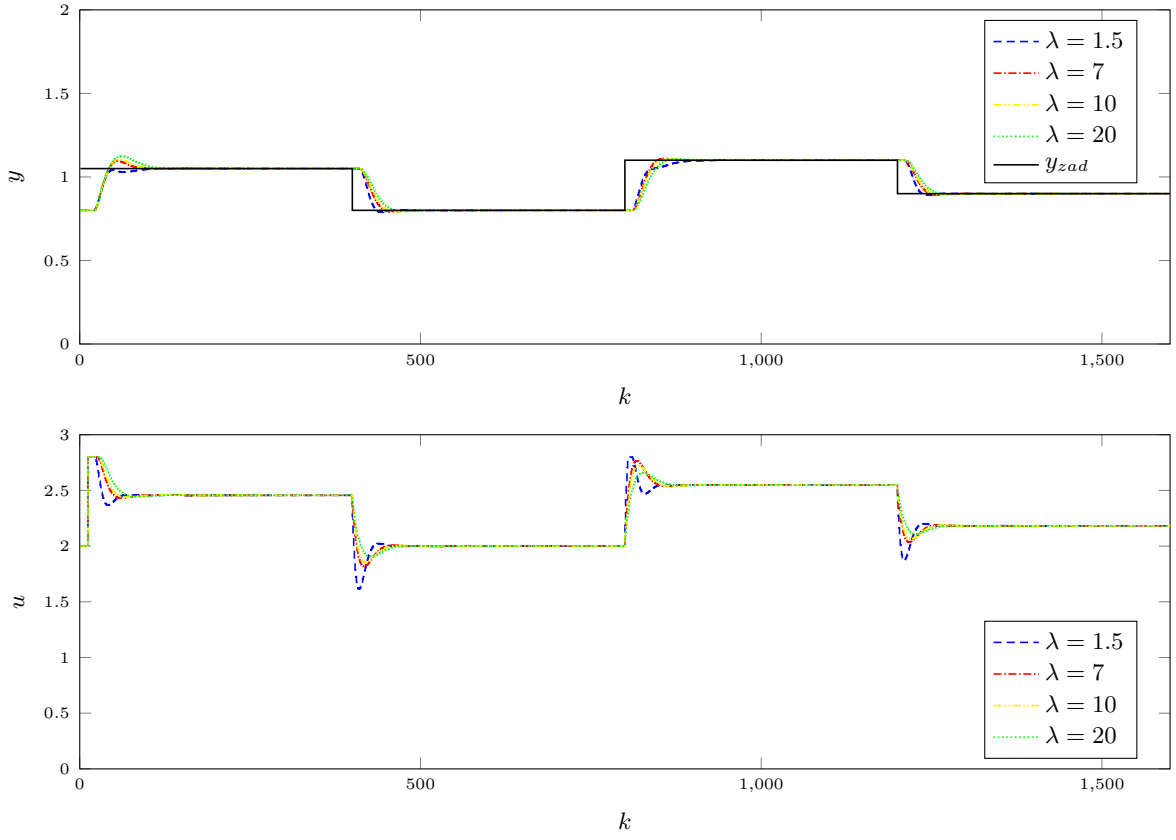
5.2. DMC

Dobieranie parametrów regulatora DMC polegało na eksperymentalnym doborze parametru λ przy wartościach $D = N = N_u = 125$ i porównywaniu wyników. Horyzonty zostały obliczone na podstawie odpowiedzi skokowej(jest to moment, w którym odpowiedź się ustabilizowała)

Wskaźnik jakości regulacji dla różnych wartości parametru λ wynosił odpowiednio:

— $\lambda = 1,5 \Rightarrow E = 5,6471$
— $\lambda = 7 \Rightarrow E = 6,2316$
— $\lambda = 10 \Rightarrow E = 6,4707$
— $\lambda = 20 \Rightarrow E = 7,0259$

Regulator o $\lambda = 1,5$ miał najmniejszy błąd i nie wprowadzał przeregulowania, jednak najwolniej dążył do wartości zadanej. Jeśli pożądaną cechą obiektu jest jak najszybsze osiąganie y_{zad} z możliwością lekkiego przeregulowania, najlepszym regulatorem jest ten z $\lambda = 7$, jednak jeśli zależy nam na jak najmniejszym przeregulowaniu, zastosowalibyśmy ten z najniższą wartością parametru λ.



Rys. 5.2. Regulacja DMC dla różnych wartości parametru λ

6. Zadanie 6.

6.1. PID

Optymalizacja wskaźnika jakości w przypadku algorytmu PID została dokonana za pomocą dwóch funkcji programu Matlab:
— fmincon – znajdująca minimum ograniczonej funkcji nieliniowej wielu zmiennych
— ga – znajdująca minimum funkcji używając algorytmu genetycznego
Funkcje te zostały wywołane w poniższy sposób:

```
[paramsPID1, ~, ~] = fmincon(@policzPID, [2 10 3.2], [], ...
 [], [], [], [0 0.1 0], []);
[paramsPID2, ~, ~] = ga(@policzPID, 3, [], [], ...
 [], [], [0 0.1 0], []);
```

Szukane parametry regulatora (K, T_i oraz T_d) zostały ograniczone z dołu, K oraz T_d zostały ustawione jako nieujemne, natomiast T_i nie mniejsze od 0.1, aby nie było ono zerowe, co skutkowałoby dzieleniem przez zero w algorytmie. Parametry wyliczone przez funkcje oraz jakość regulacji widać na wykresach 6.1 oraz 6.2. Błędy regulacji dla parametrów wyznaczonych danymi funkcjami wynoszą:
— fmincon: $E = 6,2202$
— ga: $E = 6,2618$
W przypadku obu funkcji, parametry regulatora PID były dostateczne, obiekt był stabilny, jednak zbyt wolny i gorszy od regulatora z parametrami, wyznaczonymi metodą eksperymentalną. Funkcja fmincon poradziła sobie trochę lepiej, obiekt szybciej osiągał wartość zadaną i cechował się mniejszym błędem, jednak była to niewielka różnica.

6.2. DMC

Optymalizacja wskaźnika jakości w przypadku algorytmu DMC została dokonana za pomocą funkcji ga, gdyż pozwalała ona na ograniczenie szukanych parametrów N oraz N_u do liczb całkowitych. Trzeci szukany parametr – λ – jest liczbą rzeczywistą. Funkcja ga została wywołana w poniższy sposób:

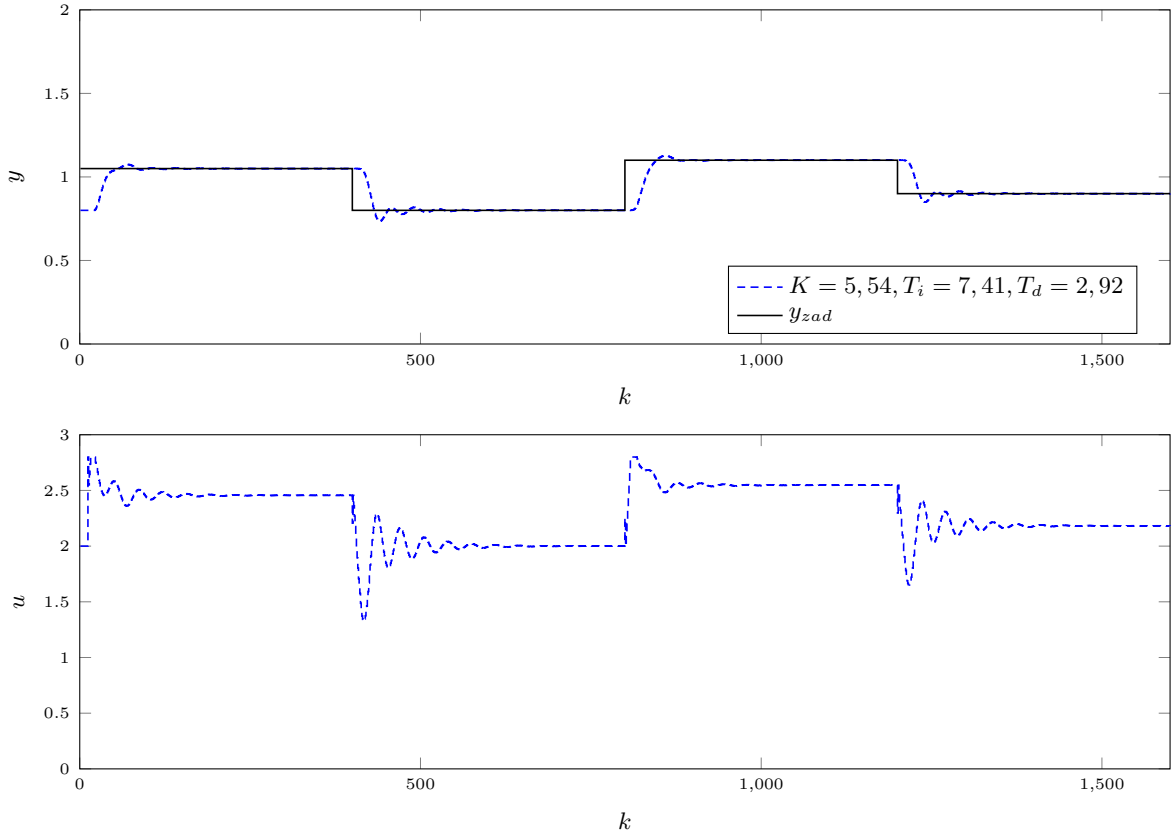
```
[paramsDMC, ~, ~] = ga(@policzDMC, 3, [], [], [], [], ...
 [1 1 0], [Kk Kk Inf], [], [1 2]);
```

Parametry wyznaczone przez funkcję ga wynoszą:

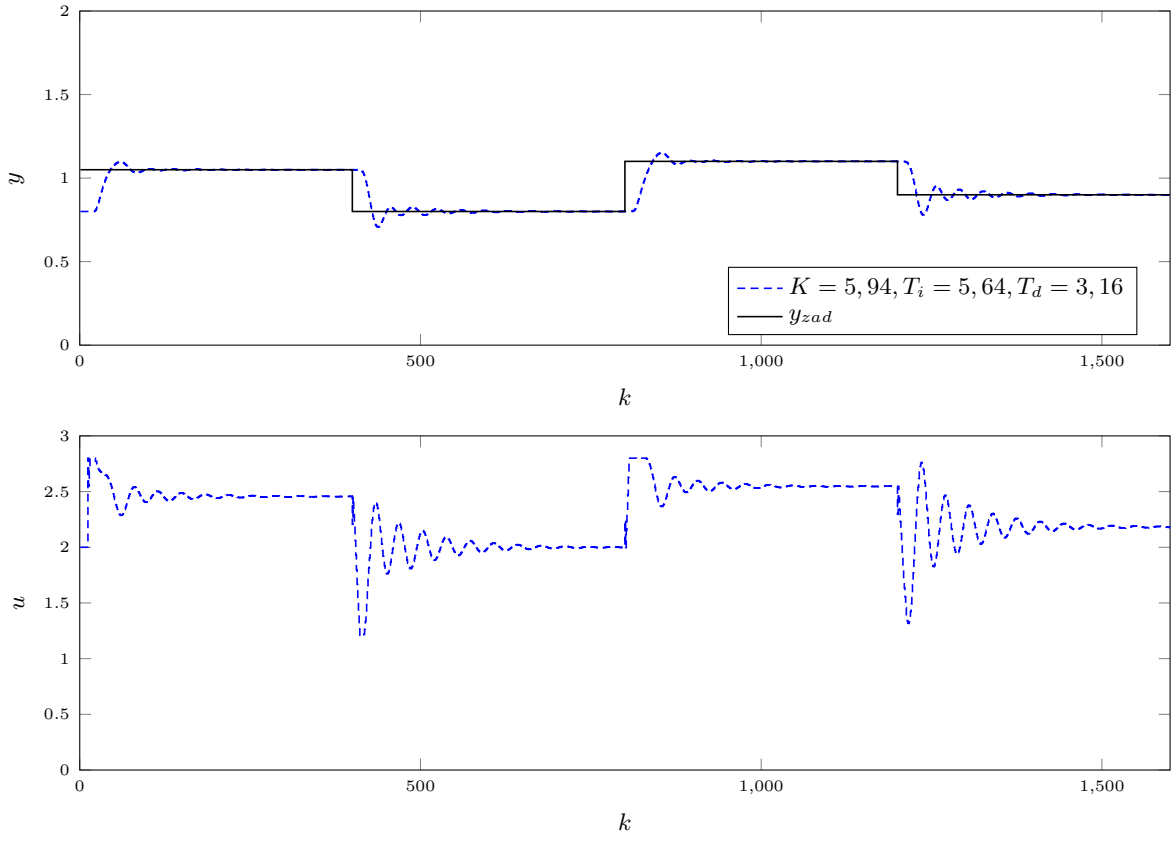
$$N = 759; N_u = 4; \lambda = 0,6096$$

Jakość regulacji widać na wykresie 6.3

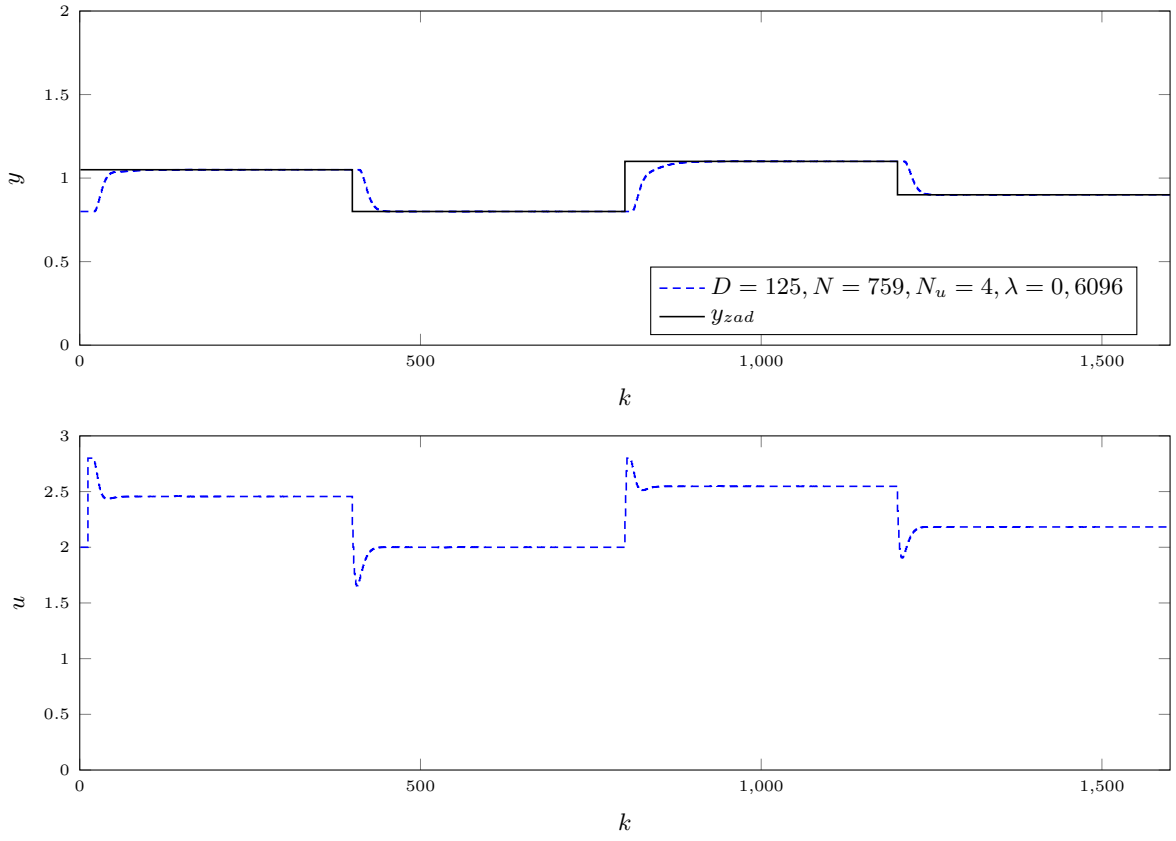
W tym przypadku błąd regulacji wynosi $5,5934$, jest on więc mniejszy niż w przypadku regulatorów, wyznaczonych metodą eksperymentalną. Jak widać, regulator bardzo dobrze poradził sobie ze zmianami wartości zadanej, osiągał je szybko, choć nieco wolniej niż ten z parametrami, wyznaczonymi w zadaniu 5.



Rys. 6.1. Regulacja PID dla nastaw, obliczonych funkcją fmincon



Rys. 6.2. Regulacja PID dla nastaw, obliczonych funkcją ga



Rys. 6.3. Regulacja DMC dla parametrów, obliczonych funkcją ga