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

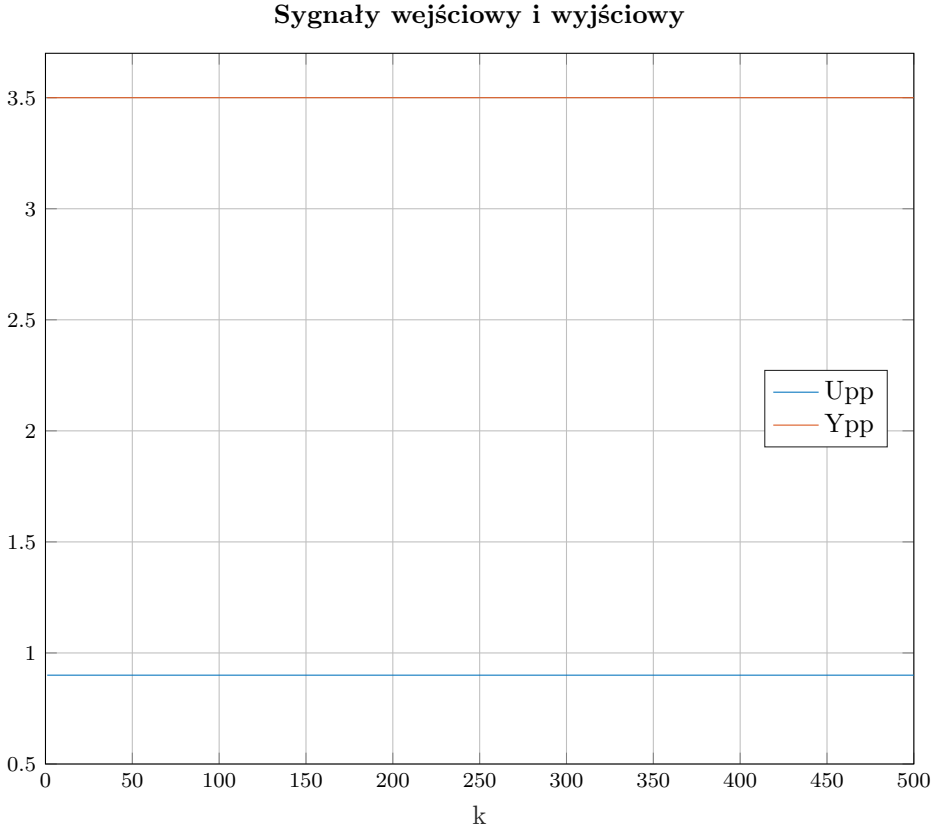
Spis treści

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 Upp. Punkty pracy Upp oraz Ypp odczytaliśmy po wywołaniu w terminalu Matlaba: $symulacja\_obiektu12y\_p1$. 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
```



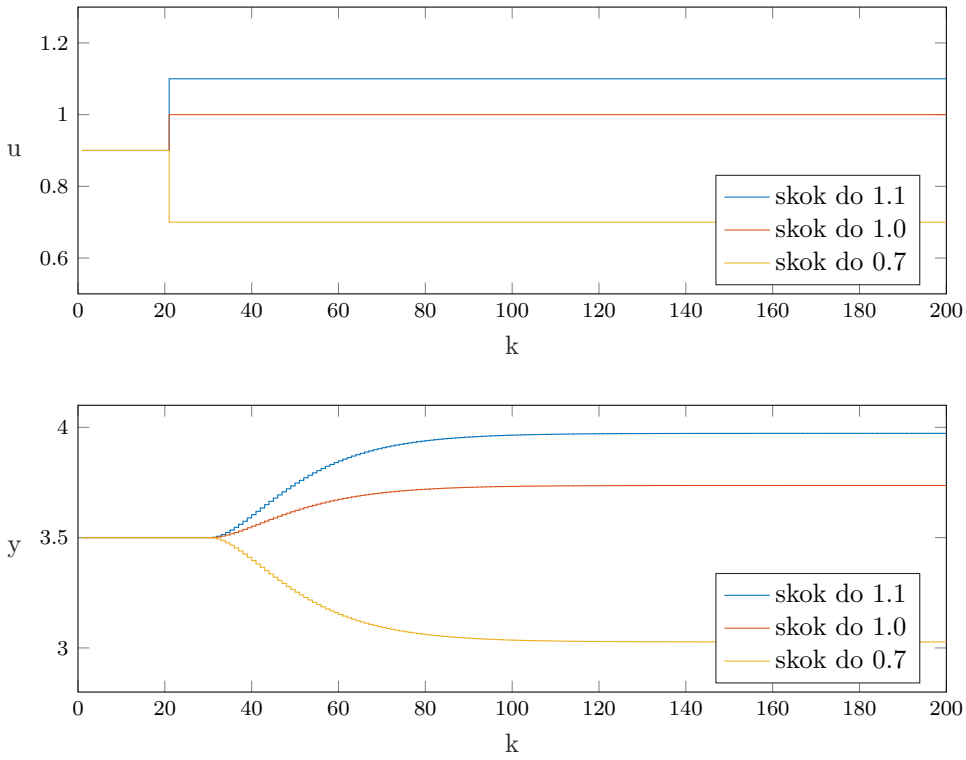
Rys. 1. Punkty pracy Upp (sygnał wejściowy) i Ypp (sygnał wyjściowy)

Jak widać, przy niezmiennym sterowaniu Upp 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

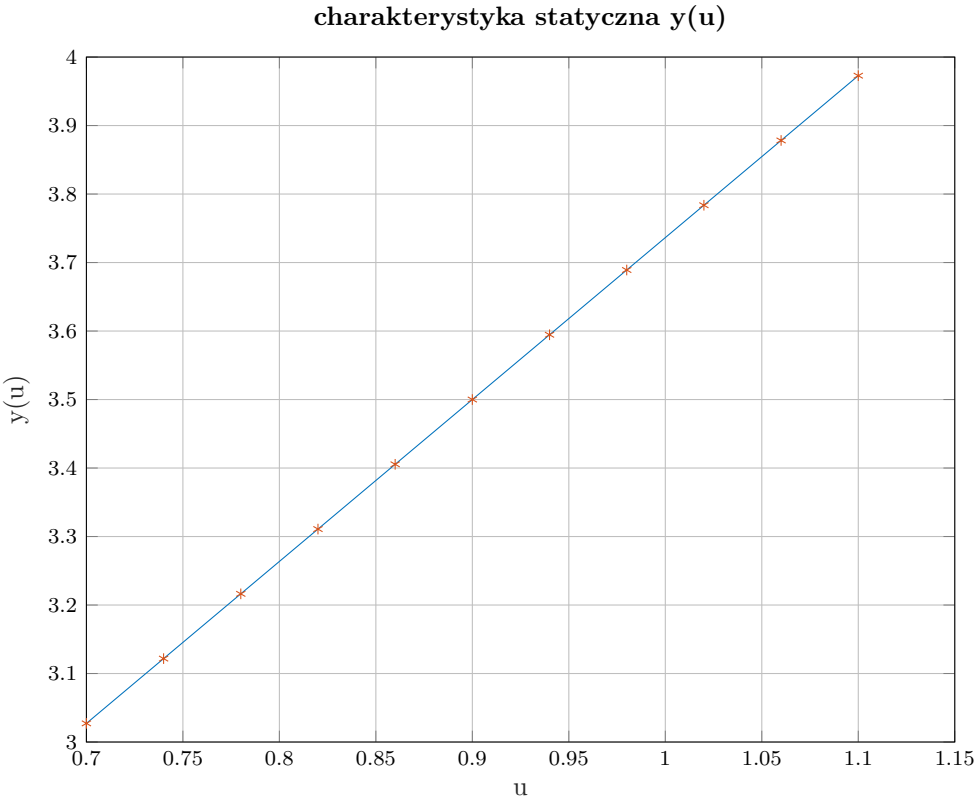
Na pierwszy rzut oka wydaje się, że właściwości statyczne i dynamiczne są liniowe. Poniżej przedstawiono implementację:

```
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:

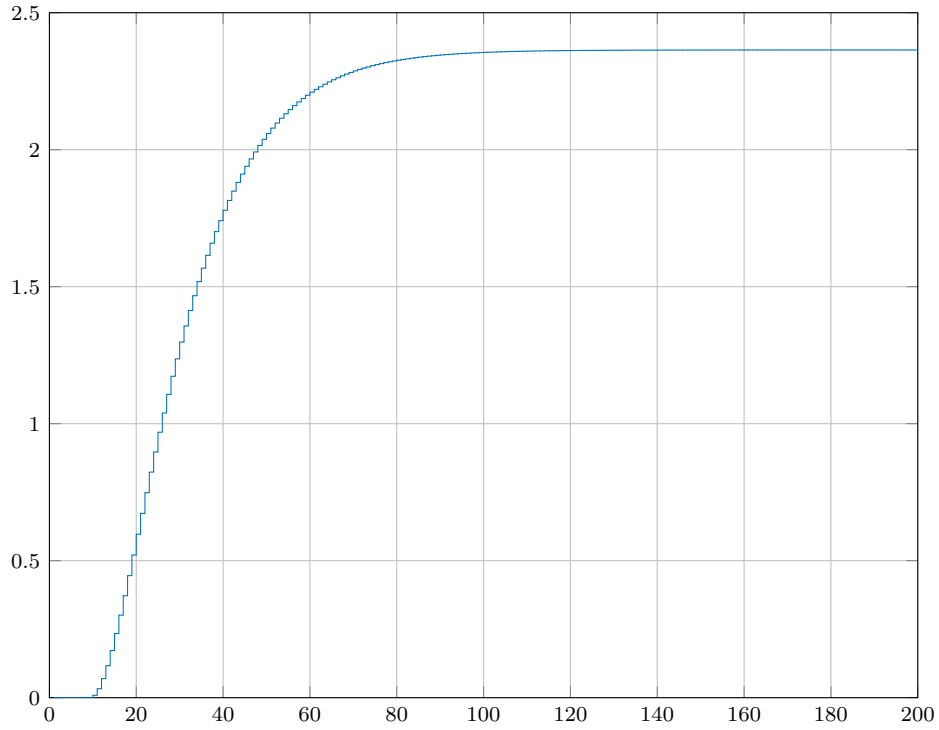
$$s_k = \frac{y(k) - Ypp}{Uk - Upp}$$

gdzie $k = 1, ..., D$ oraz

$$u(k) = \begin{cases} Upp & \text{dla } k < 0 \\ Uk & \text{dla } k >= 0 \end{cases}$$

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:

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

W badanym obiekcie występuje opóźnienie równe 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); % przyrost sterowania w chwili k-tej

        % 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
```

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 $j = 1, ..., D$, 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;

    load(s); % odpowiedź skokowa 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
```

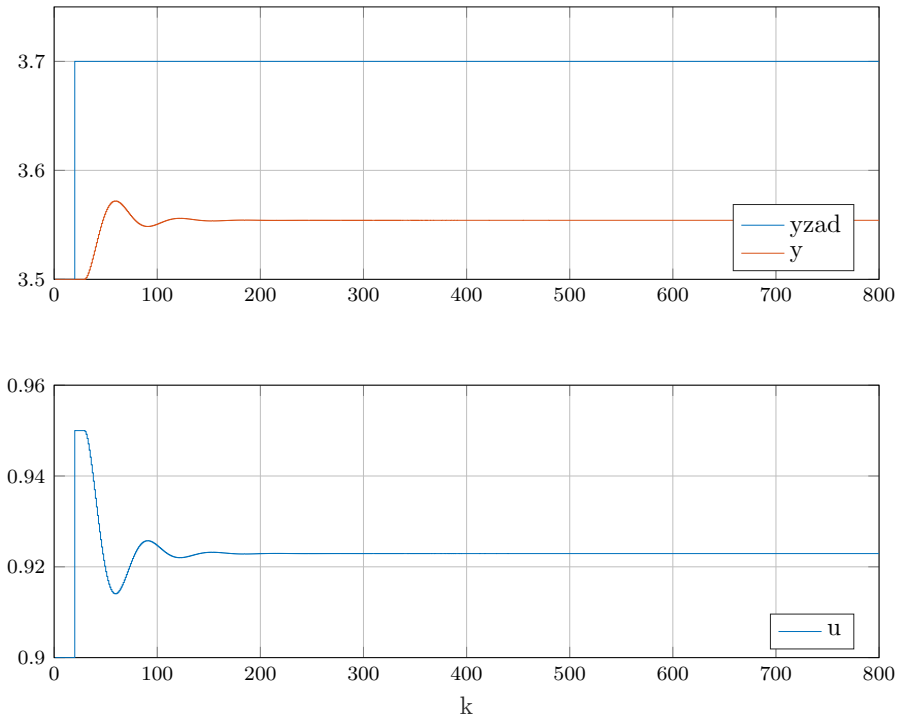
1.5. Dobór nastaw algorytmów regulacji metodą eksperymentalną

1.5.1. PID

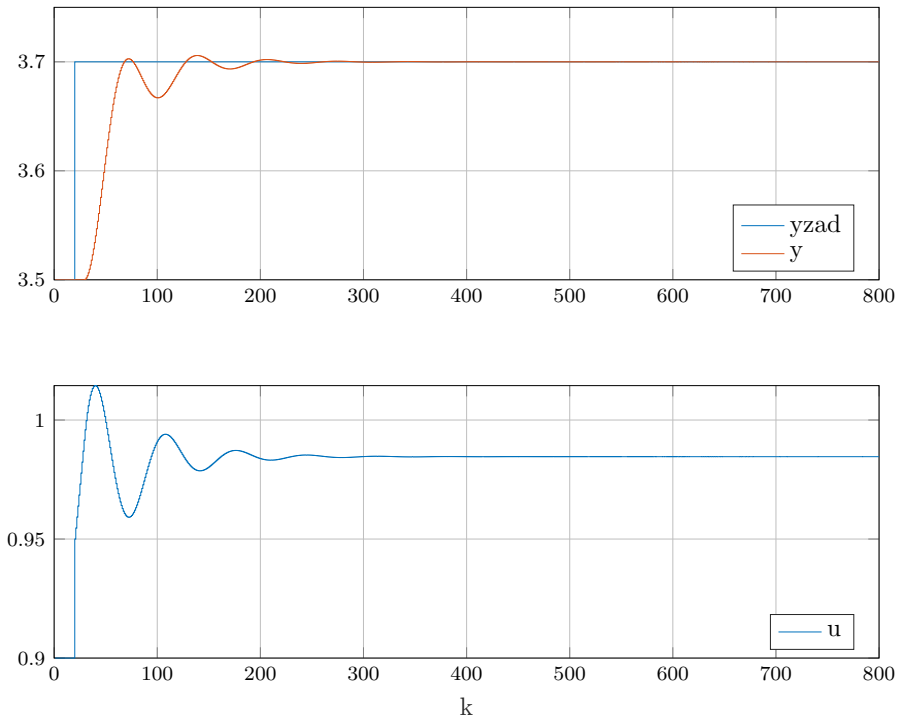
Kolejne kroki doboru nastaw algorytmu PID metodą eksperymentalną:
1. Analiza regulatora P - wyzerowanie członu całkującego (tzn. ustawienie parametru T_i na bardzo dużą wartość) i różniczkującego (tzn. ustawienie parametru T_d na zero). Zwiększanie parametru wzmocnienia K do momentu jak najmniejszego 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:

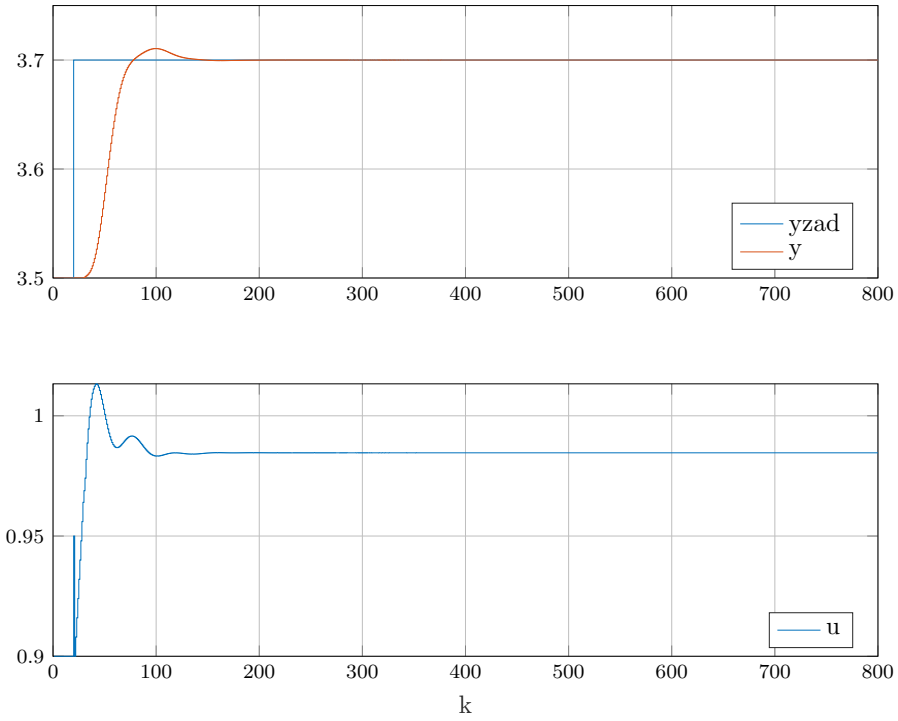
$$K = 0.8, T_i = 10, T_d = 4$$



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: $E = 1.1265$

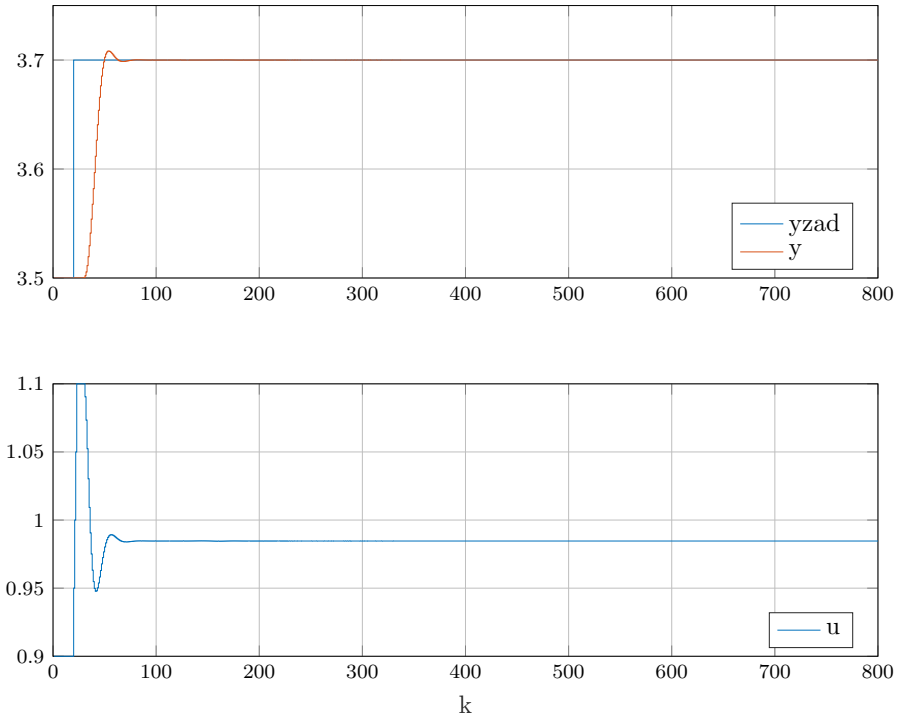
Najmnieszy błąd wskaźnika regulacji miał regulator PI o nastawach $K = 0.6$ i $T_i = 13$. Jednak jakościowo najlepszy przebieg ma regulator PID o nastawach $K = 0.8$, $T_i = 10$, $T_d = 4$. nie ma tutaj znacznego przeregulowania i oscylacji. Sygnał sterujący również nie zmienia się gwałtownie. Wyjątkiem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

1.5.2. DMC

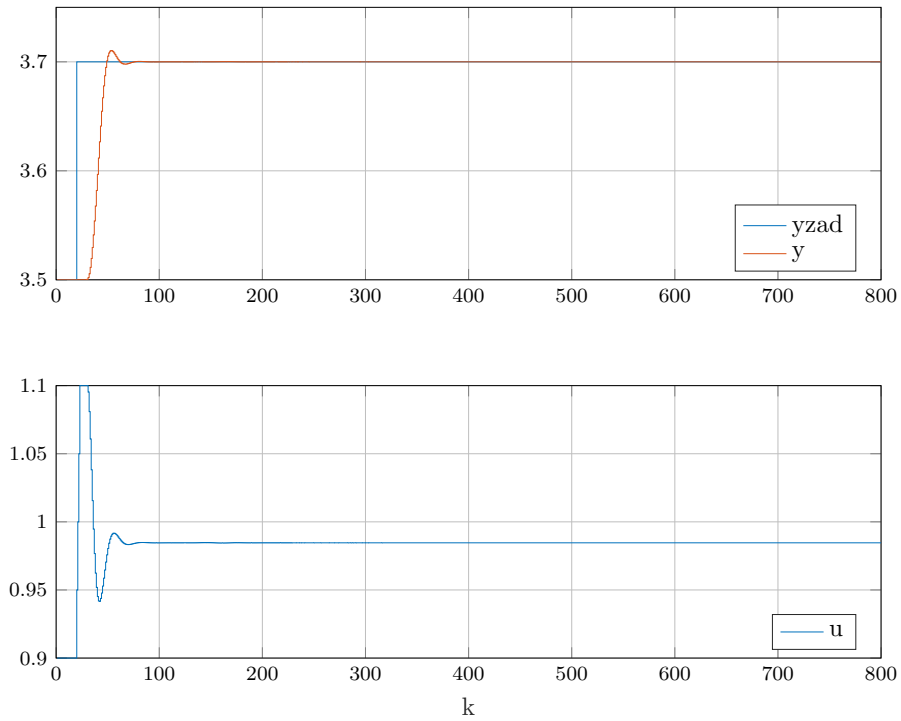
Kolejne kroki doboru nastaw algorytmu DMC metodą eksperymentalną:
1. Ustalenie długości horyzontu dynamiki z odpowiedzi skokowej obiektu - moment gdy sygnał na wyjściu się nie zmienia lub zmienia się nieznacznie. W przypadku naszego obiektu przyjęliśmy $D = 130$
2. Ustawienie wartości horyzontu predykcji 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 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:

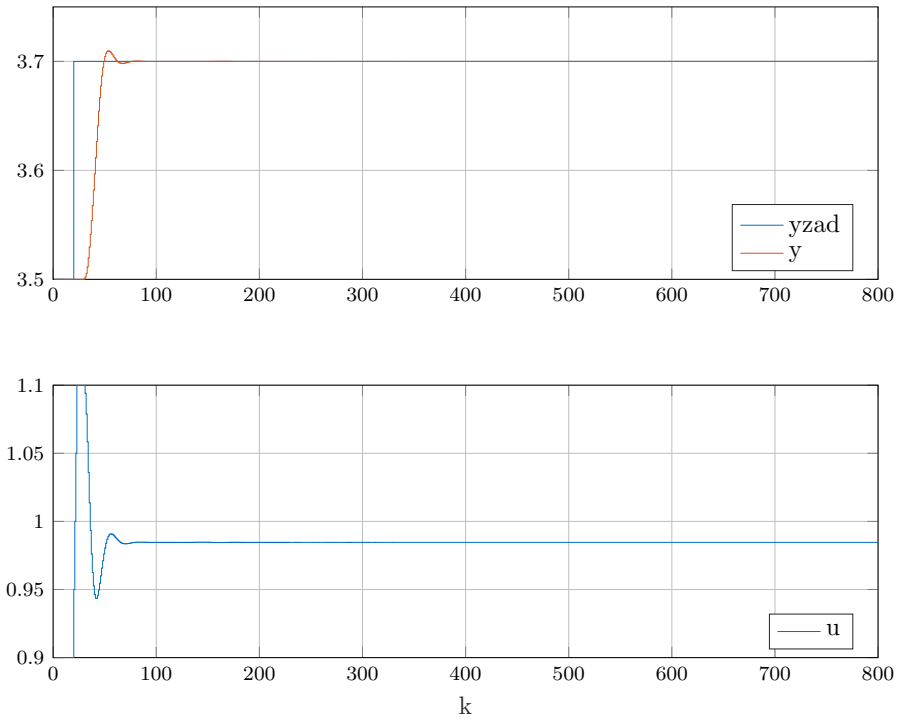
$$N = 40,\ N_u = 5,\ \lambda = 1$$



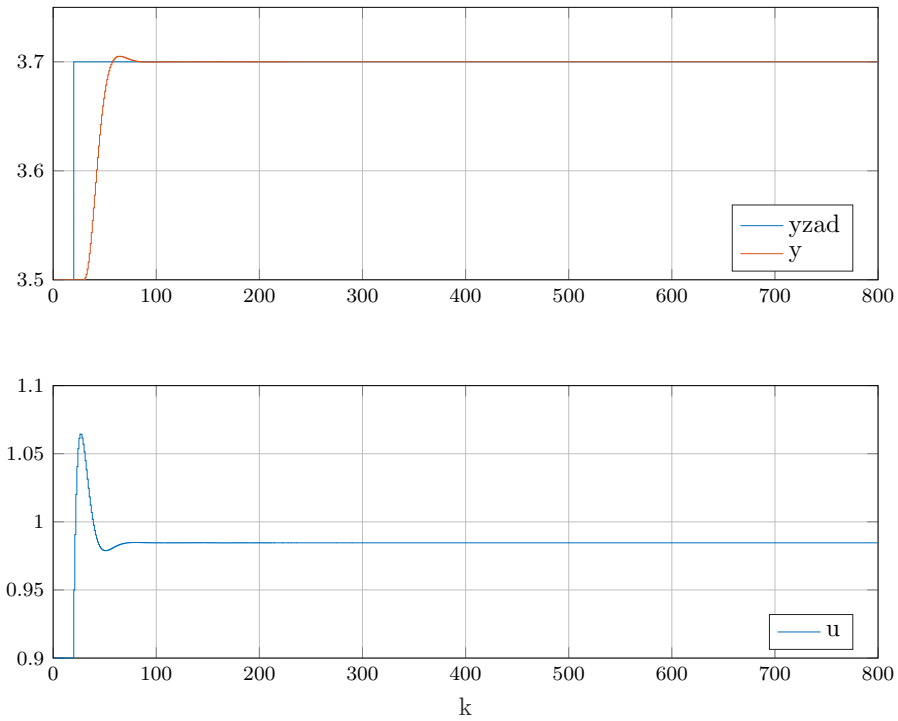
Rys. 8. $N = N_u = D = 130$, $\lambda = 1$
Błąd ilościowy: $E = 0.7239$



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 programi 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 kończy działanie, kiedy osiągnie najmniejszą funkcję błędu.

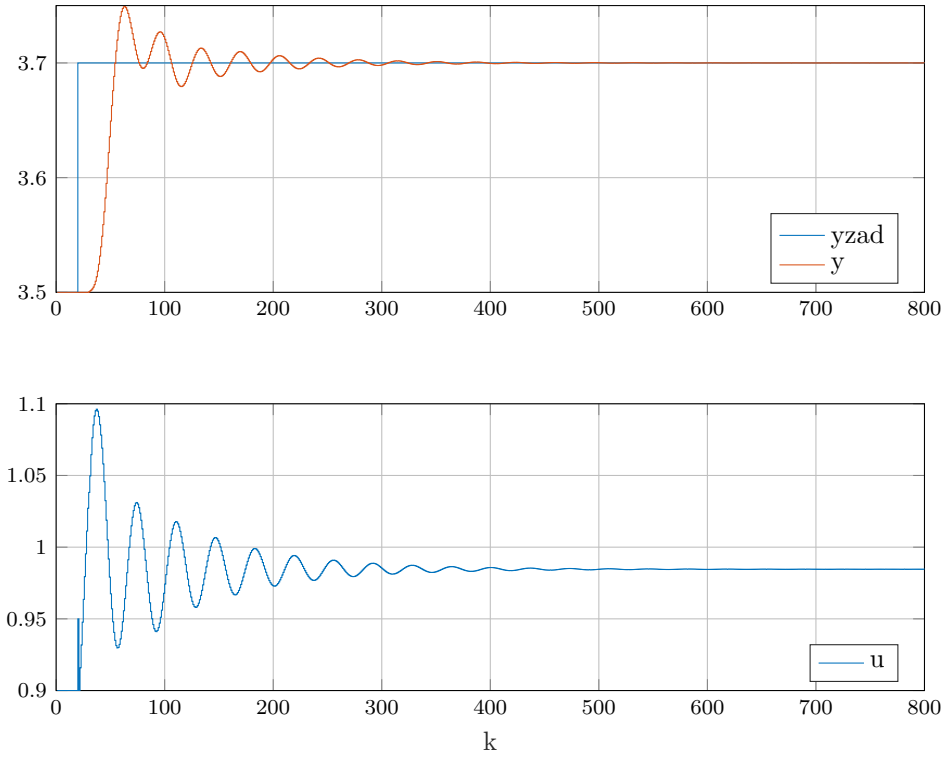
1.6.1. PID

```
    %% 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 trzykrotnośc w porównaniu z eksperymentalną metodą, gdzie obiekt już był stabilny w okolicach 150-tej chwili.

1.6.2. DMC

Implementacja wygląda podobnie w przypadku regulatora DMC.

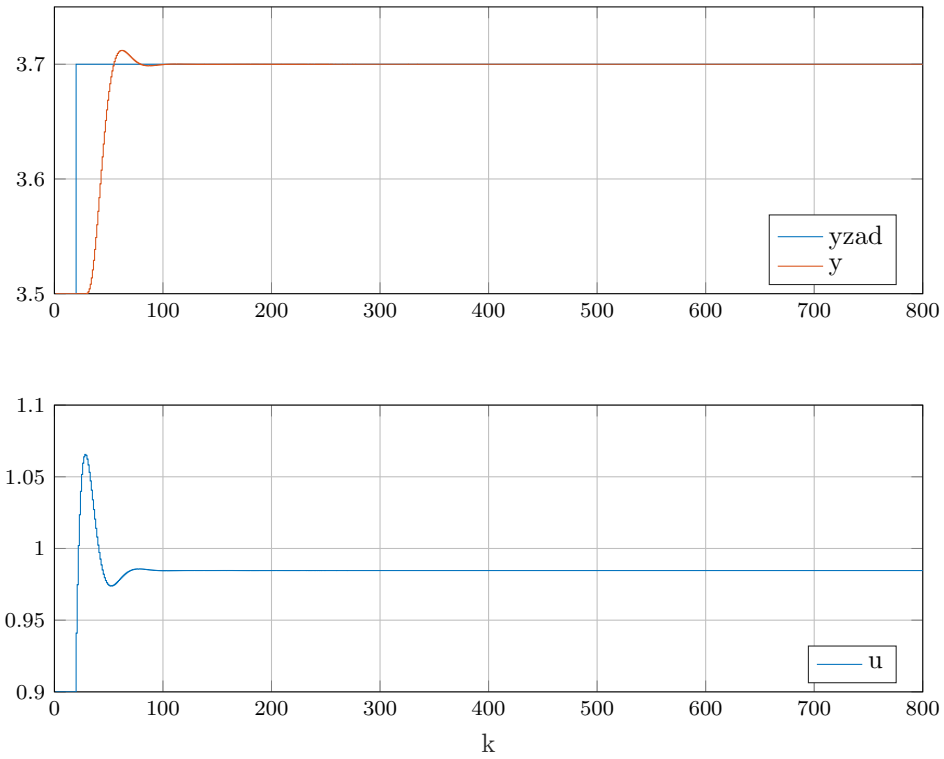
```
    %% Optymalizacja nastaw DMC
%fmincon
objective = @(x) DMC (x(1), x(2), x(3), x(4));
x_initial = [130, 130, 130, 1];
lb = [70, 10, 1, 0.1];
ub = [150, 150, 150, 30];

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 mamy ciekawą sytuację, ponieważ nastawy otrzymane z `fmincona` mają gorsze efekty niż te otrzymane eksperymentalnie przez nas - nieznacznie ale jednak.

2. CZĘŚĆ LABORATORYJNA

2.1. Sprawdzenie możliwości sterowania i pomiaru stanowiska

Sterowanie stanowiska laboratoryjnego odbywało się przy pomocy funkcji w MATLABIE `MinimalWorkingExample()`

Sygnał sterujący wentylatorem W1 oraz grzałką G1 ustawiano jako argument funkcji `sendControls()`, której pierwszym argumentem była tablica z ID elementów sterowalnych, a drugim tablica z wartościami sygnału sterującego. ID wentylatora W1 to 1, a grzałki G1 5.

```
function MinimalWorkingExample()
    addpath('D:\SerialCommunication');
    initSerialControl COM5 % initialise com port
    while(1)

        %% obtaining measurements
        measurements1 = readMeasurements(1);
        measurements3 = readMeasurements(3);

        %% sending new values of control signals
        sendControls([ 1,5], ... send for these elements
                     [ 50,27]); % new values

         measurement = readMeasurements(1:1)
        %% synchronising with the control process
        waitForNewIteration();
    end
end
```

Należało również określić wartość pomiaru temperatury w punkcie pracy

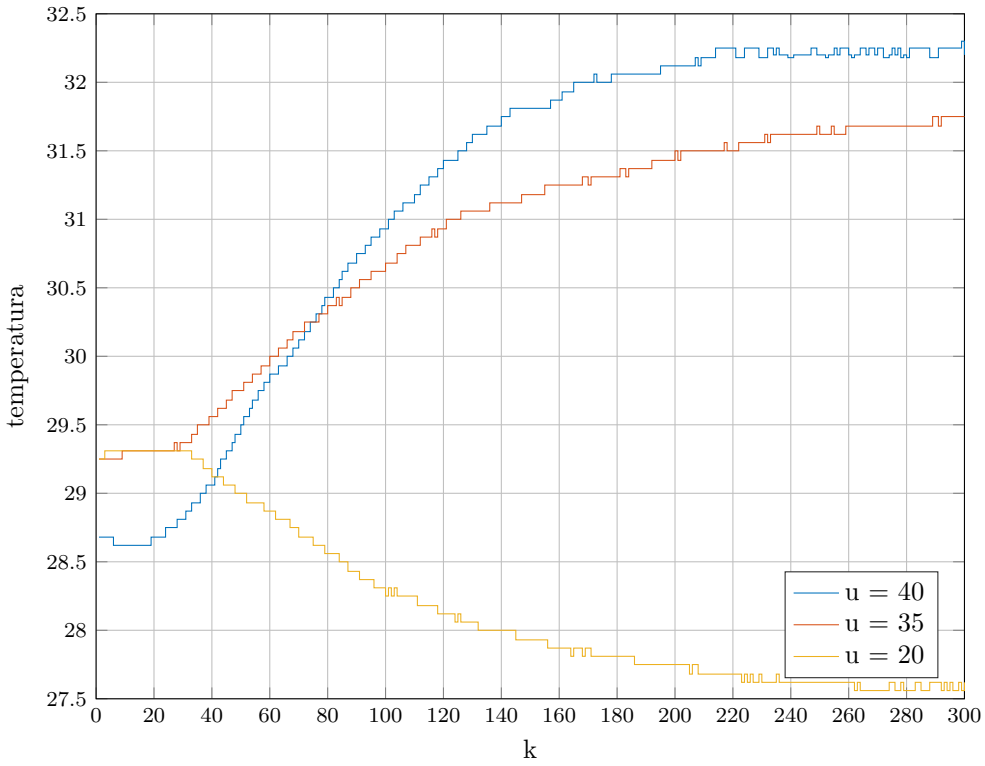
$$G1 = 27$$

Na pierwszym laboratorium temperatura w punkcie pracy $u = 27$ wyniosła 29.12 °C

2.2. Wyznaczenie odpowiedzi skokowej procesu

2.2.1. Odpowiedzi skokowe

Odpowiedzi skokowe procesu wyznaczono poprzez zmianę sygnału sterującego u z punktu pracy i obserwacji wyjścia.



Rys. 1. Odpowiedzi skokowe dla zmian sygnału sterującego

Przy jednym pomiarze przez nieuwagę zmieniono sygnał sterujący u zanim obiekt osiągnął wartość temperatury w punkcie pracy.

2.2.2. Wzmocnienie statyczne

Dla pomiarów rozpoczynających się z punktu pracy: $u = 27$, 29.12 °C otrzymujemy wzmocnienia równe: $K = 0.320$, dla skoku sterowania $u = 8$, $K = 0.328$, dla skoku sterowania $u = -7$, $K = 0.282$, dla skoku sterowania $u = 13$. Ze względu na na inny punkt pracy widać małą różnicę dla zmiany sterowania u = 13, lecz można przyjąć, że wzmocnienie statyczne to średnia wzmocnień dla dwóch pierwszych zmian sterowania, która wynosi: $K = 0.324$.

2.3. Przekształcenie odpowiedzi skokowej

2.3.1. Odpowiedź skokowe wykorzystywana w algorytmie DMC

Należał uzyskać odpowiedź skokową wykorzystywaną w algorytmie DMC (zestaw liczb s_1, s_2, ...), a więc taką, która odpowiadałaby skokowi jednostkowemu sygnału sterującego w chwili $k = 0$ na wartość $u = 1$. Przypomnienie wzoru wykorzystywanego w projekcie:

$$s_k = \frac{y(k) - Ypp}{|Uk - Upp|}$$

gdzie $k = 1, ..., D$ oraz

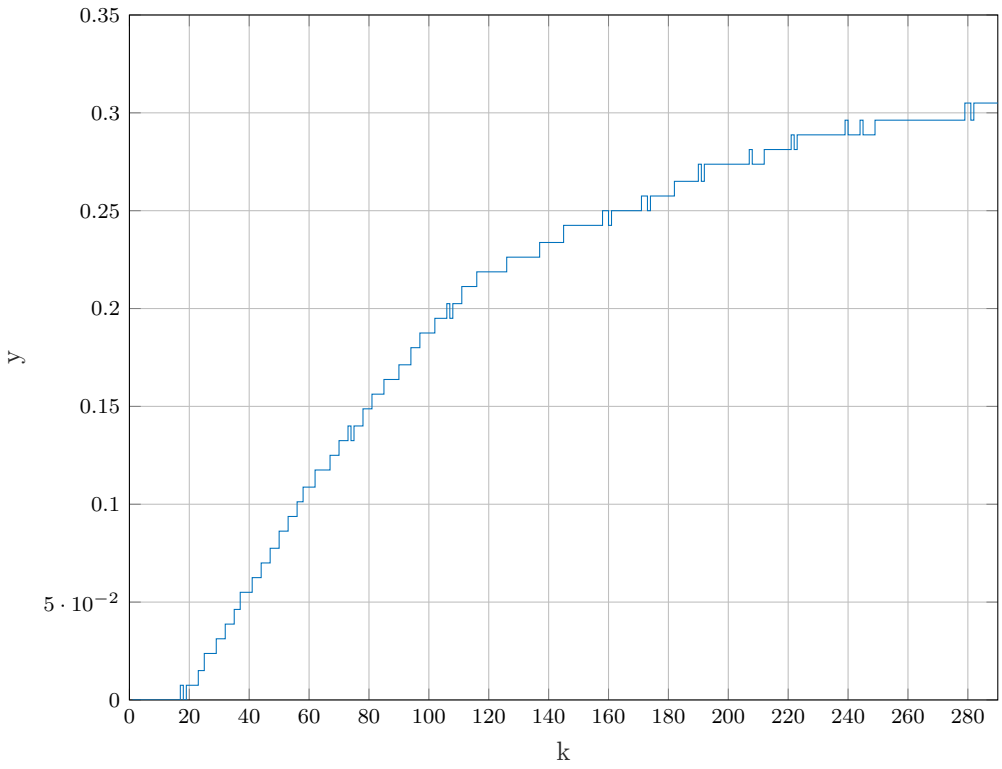
$$u(k) = \begin{cases} Upp & \text{dla } k < 0 \\ Uk & \text{dla } k >= 0 \end{cases}$$

Dla pomiarów z laboratoriów, powyższe równanie zaimplementowano:

```
for k = 1:300
    if k > 10

        sp(k-10) = (z2_27_35(k)-z2_27_35(10))/8;

    end
end
```

Bardzo ważnym szczegółem jest przesunięcie obliczeń o 10 próbek. Odpowiedzi skokowe były mierzone od chwili $k = 0$, natomiast sama zmiana sygnału u następowała dopiero w chwili $k = 10$. Ponieważ zestaw liczb s_k zakłada zmianę sygnału sterującego w chwili $k = 0$ konieczne jest przeliczenie punktów pomiarowych od 10. próbki.

Poniżej przedstawiono wykres odpowiedzi skokowej wykorzystywanej w algorytmie DMC.



Rys. 2. Odpowiedź skokowa dla skoku jednostkowego sygnału sterującego

2.3.2. Aproksymacja odpowiedzi skokowej

Mając odpowiedź skokową, można wyznaczyć model transmitancyjny obiektu. W tym celu należy aproksymować odpowiedź jako człon inercyjny drugiego rzędu z opóźnieniem, który opisany jest następującą transmitancją:

$$G(s) = \frac{K}{(sT_1+1)(sT_2+1)} e^{-T_d T_p s}$$

która po zastosowaniu transformaty Z wygląda:

$$G(z) = \frac{b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} z^{-T_d}$$

gdzie:

$$a_1 = -\alpha_1 - \alpha_2$$

$$a_2 = \alpha_1 \alpha_2$$

$$\alpha_1 = e^{-\frac{1}{T_1}}$$

$$\alpha_2 = e^{-\frac{1}{T_2}}$$

$$b_1 = \frac{K}{T_1 - T_2}[(T_1(1-\alpha_1) - T_2(1-\alpha_2)]$$

$$b_2 = \frac{K}{T_1 - T_2}[(\alpha_1 T_2(1-\alpha_2) - \alpha_2 T_1(1-\alpha_1)]$$

co przekłada się na równanie różnicowe postaci:

$$y(k) = b_1 u(k - T_d - 1) + b_2 u(k - T_d - 2) - a_1 y(k-1) - a_2 y(k-2);$$

Poniżej przedstawiono implementację wzorów w MATLABIE:

```
% wyliczenie stałych
    alfa1 = eu^(-1/T1);
    alfa2 = eu^(-1/T2);
    a1 = -alfa1 - alfa2;
    a2 = alfa1*alfa2;
    b1 = (K/(T1-T2))*(T1*(1-alfa1)-T2*(1-alfa2));
    b2 = (K/(T1-T2))*(alfa1*T2*(1-alfa2)-alfa2*T1*(1-alfa1));

    % inicjalizacja modelu
    ymod(1:duration) = 0;
    u(1:19) = 1;
    u(20:duration) = 1;
    e(1:duration) = 0;

    % fmincon czasem wylicza niepoprawne wspolczynniki
    if(~isnan(a1) && ~isnan(a2) && ~isnan(b1) && ~isnan(b2))

        err = 0;
        for k=20:duration
            ymod(k) = b1*u(k-Td-1) + ...
                b2*u(k-Td-2) - ...
                a1*ymod(k-1) - ...
                a2*ymod(k-2);
```

```
            e(k) = sp(k) - ymod(k);
            err = err + e(k)^2;

        end
    end
```

2.3.3. Optymalizacja parametrów modelu

W celu optymalizacji parametrów aproksymowanego modelu, ponownie skorzystano z funkcji `fmincon`.

```
%% Optymalizacja modelu

% fmincon
objective = @(x) aproksymacja (x(1), x(2), x(3));
x_initial = [90, 100, 0.3];
lb = [10, 11, 0];
ub = [150, 151, 1];

nonlcon = [];

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

%% Funkcja błędu:

% funkcja bledu dla fmincon
function err = aproksymacja(T1, T2, K)

    % opoznienie
    Td = 17;

    % kiedy T1 == T2 wspolczynniki wychodza NaN
    if(T1 ~= T2)
        load("zad2_pomiary.mat");
        pomiary = z2_27_35;

        % odpowiedzi skokowe
        for k = 1:300
            if k > 10
                sp(k-10) = (pomiary(k)-pomiary(10))/8;
            end
        end
        %%%
        ZAIMPLEMENTOWANE PARAMETRY
        ORAZ RÓWNANIE RÓŻNICOWE
        %%%
    end
end
```

Warto zwrócić uwagę, że `fmincon` nie zawsze poprawnie wylicza współczynniki równania różnicowego, także potrzebne jest sprawdzenie, czy są one liczbami rzeczywistymi oraz, że T_1 nie jest równe T_2 (w przypadku równości dochodzi do dzielenia przez 0).

Po przeprowadzonej optymalizacji, otrzymano optymalne parametry to:

$$T_1 = 10$$

$$T_2 = 75$$

$$K = 0.307$$

Uśredniając opóźnienie ze zmierzonych odpowiedzi skokowych (zarówno przy podnoszeniu jak i obniżaniu temperatury obiektu), otrzymuje się opóźnienie:

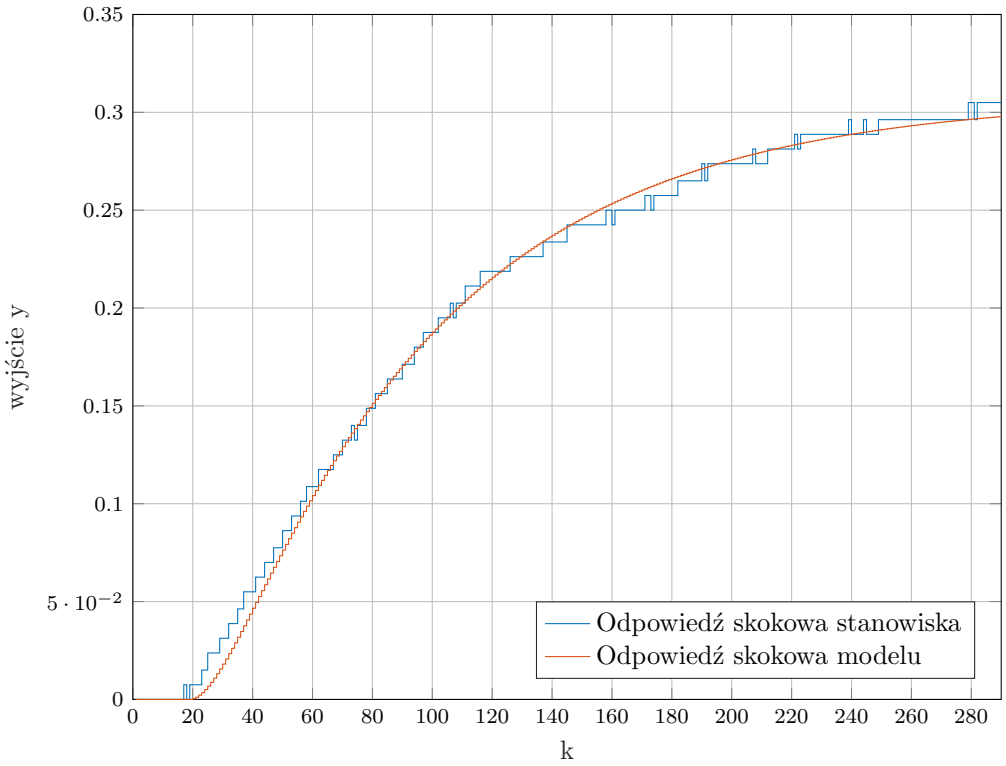
$$T_p = 17$$

A zatem transmitancja modelu będzie wyrażać się wzorem:

$$G(s) = \frac{0.307}{(10s+1)(75s+1)} e^{-17T_p s}$$

2.3.4. Porównanie aproksymowanego modelu z obiektem

Poniżej przedstawiono na jednym rysunku odpowiedź skokową modelu oraz odpowiedź skokową stanowiska grzewczo-chłodzącego.



Rys. 3. Odpowiedź skokowa modelu oraz stanowiska grzewczo-chłodzącego

2.4. Regulacja PID oraz DMC

Regulacja PID:

```
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
```

Regulacja DMC:

```
function du = regulacjaDMC(k, u, ypom, yzad, D, N, K, Mp)

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

    Ke = 0;
    for i = 1 : N
        Ke = Ke + K(1, i);
    end

    elem = 0;
    for j = 1:D-1

        ku = K(1,:) * Mp(:,j);

        if k-j<= 1
            du_kmj = 0;
        else
            du_kmj = u(k-j) - u(k-j-1);
        end

        elem = elem + ku*du_kmj;
    end

    du = Ke * e(k) - elem;

end
```

Macierz M oraz Mp liczone są tak samo jak w projkcie. Po wyznaczeniu odpowiedzi skokowej obiektu, liczone są jednorazowo dla zadanych paramterów długości horyzontów.

2.5. Dobór nastaw regulatorów PID oraz DMC metodą eksperymentalną

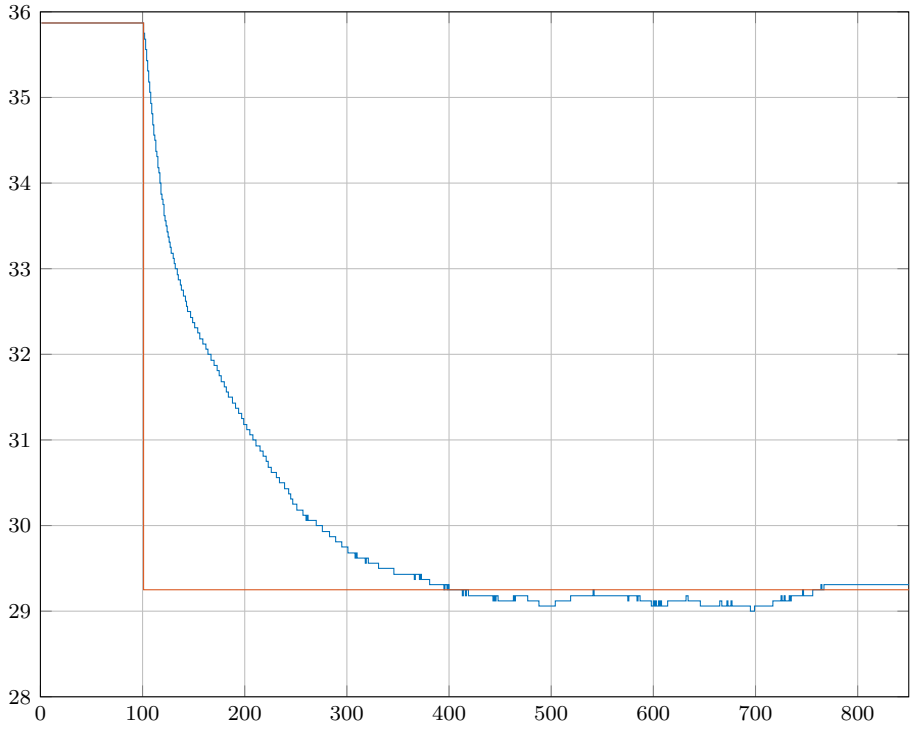
Parametry dobieraliśmy eksperymentalnie, lecz ze względu na długie stałe czasowe pozwoliliśmy sobie na zamieszczenie jedynie wykresów dla optymalnych nastaw, gdzie wyjście ustaliło się po czasie reprezentatywnym na wykresie.

Nastawy regulatora PID:
$K = 5$, $T_i = 60$, $T_d = 0.3$

Nastawy regulatora DMC:
$D = 580$, $N = 580$, $N_u = 80$, $\lambda = 1$

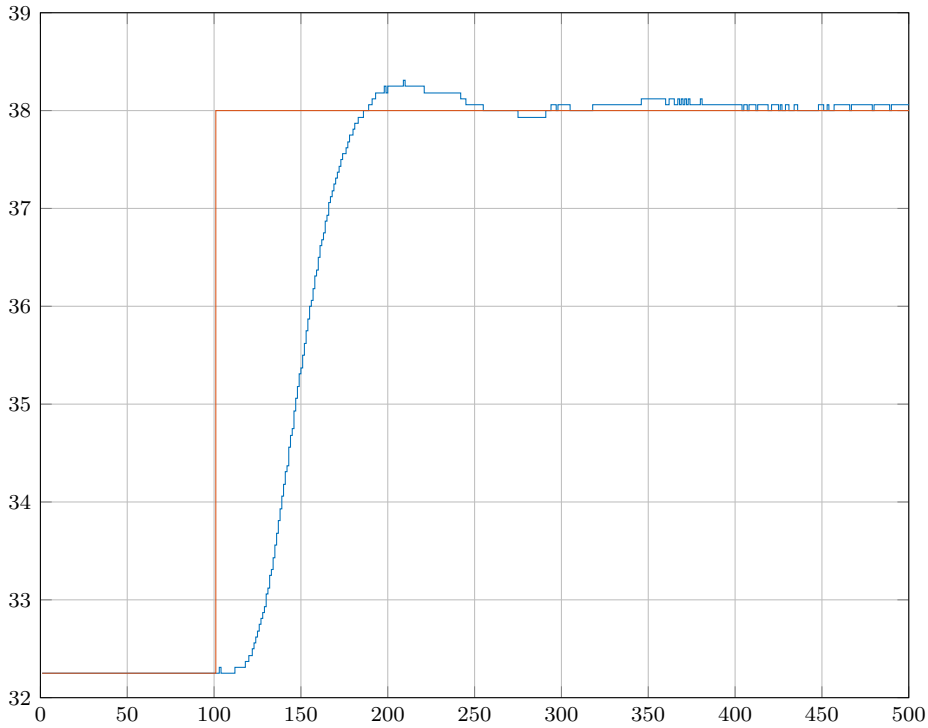
2.5.1. Omówienie wyników i ocena jakości regulacji

Poniżej przedstawione zostały przykłady chłodzenia oraz grzania dla ustalonych nastaw dla regulatora DMC oraz PID:



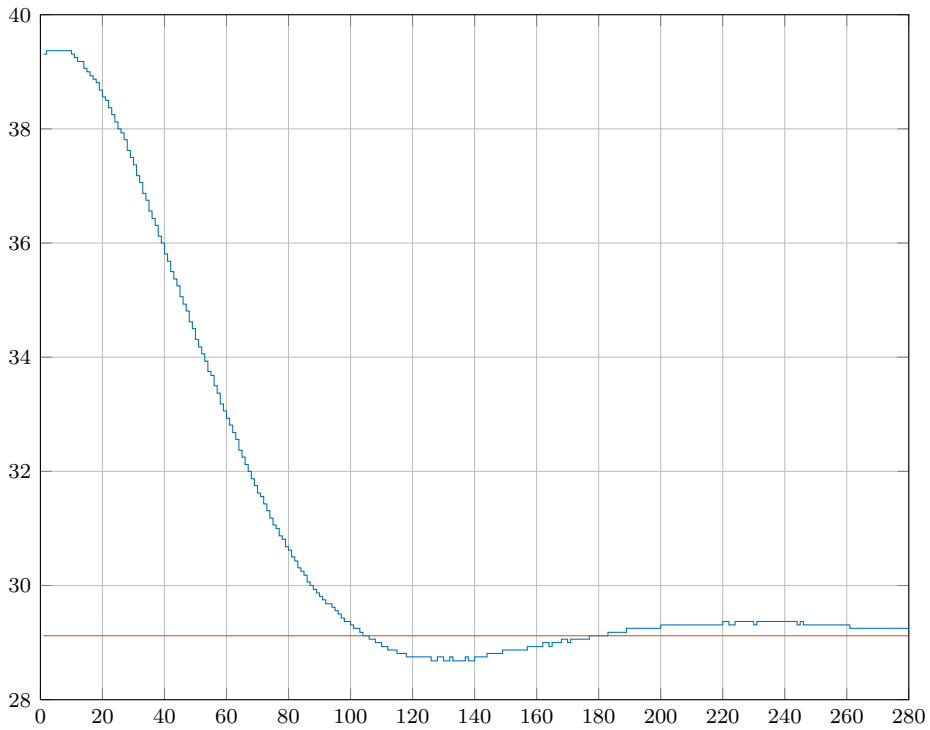
Rys. 4. DMC: Chłodzenie z 36°C do 29°C, Nu = 80, $\lambda = 1$

Błąd średniokwadratowy: E = 1471.5374



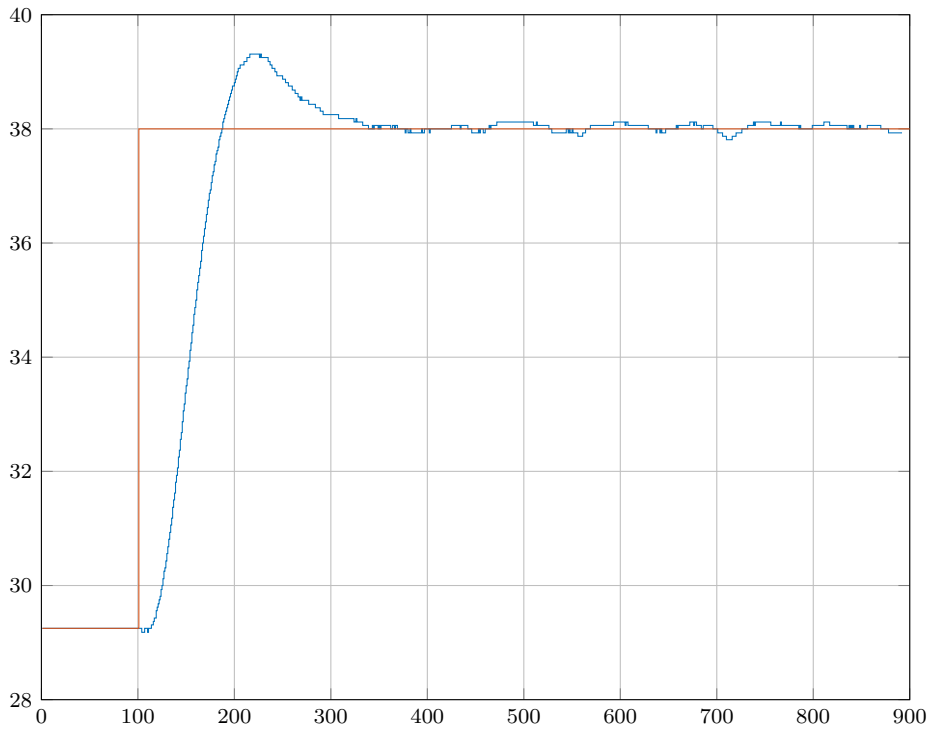
Rys. 5. DMC: Grzanie z 32°C do 38°C, Nu = 80, $\lambda = 1$

Błąd średniokwadratowy: E = 1279.4381



Rys. 6. PID: Chłodzenie z 38°C do 29°C, K=5; Ti=60; Td=0.3

Błąd średniokwadratowy: E = 4047.4239



Rys. 7. PID: Grzanie z 32°C do 38°C, K=5; Ti=60; Td=0.3

Błąd średniokwadratowy: E = 3001.1457

Analizując powyższe wykresy oraz błędy średniokwadratowe można stwierdzić, że zdecydowanie lepiej radzi sobie regulator DMC niż PID. Generuje on mniejsze przesterowanie i mniejszy błąd. Mimo tego warto zauważyć, że grzanie i chłodzenie wykonywane było dla różnych temperatur, więc trudno jednoznacznie ocenić te regulatory, w szczególności na podstawie wartości błędu średniokwadratowego. Czas dążenia do wartości zadanej jest podobny dla obydwu regulatorów, co sugerować może, że jakość działania jest relatywnie podobna z podkreśleniem przewagi DMC nad PIDem na polu przesterowania.