Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie

WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI, INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ

Sprawozdanie Sterowanie Adaptacyjne i Estymacja

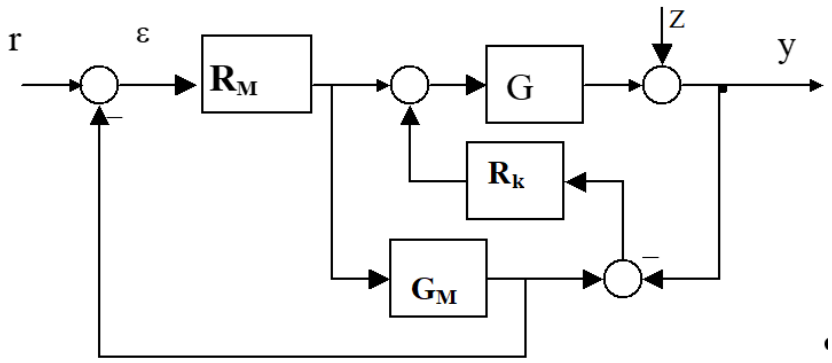
Modelowanie układu sterowania ze strukturą Model Following Control

Autorzy: **Dawid Lisek, Paweł Mańka**
Nr indeksu: 402382, 402697
Kierunek studiów: **Automatyka i Robotyka**
Specjalizacja: **Komputerowe Systemy Sterowania**
Grupa: wt 14:45 - 16:15

Spis treści

1. Cel oraz zakres projektu.

Celem projektu było zamodelowanie układu ze strukturą MFC i obiektem wieloinercyjnym G z zadawanym skokowym sygnałem referencyjnym r i skokowym sygnałem zakłócenia na wyjściu z. W zadaniu należało wykorzystać poniższą strukturę MFC:

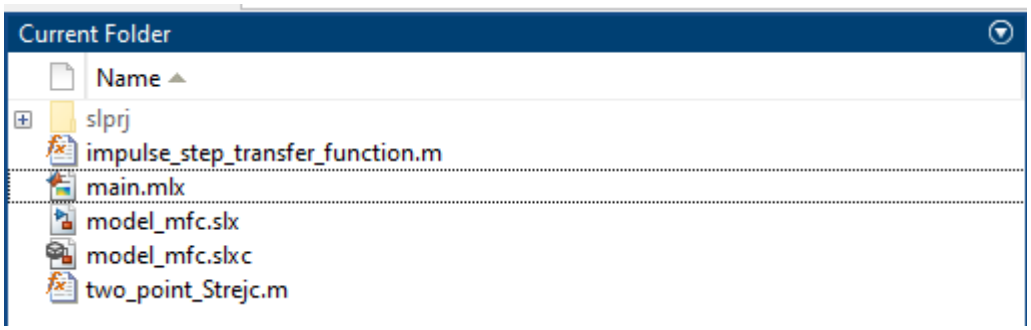


Rysunek 1 Struktura MFC

W projekcie przy pomocy dekonwolucji należało zidentyfikować kształt charakterystyki impulsowej g(t) oraz skokowej h(t). Następnie na podstawie wyznaczonej charakterystyki skokowej przy pomocy dwupunktowej metody Strejca przeprowadziliśmy identyfikację parametrów obiektu dobierając rząd, opóźnienie oraz stałą czasową.
W oparciu o model Strejca został dobrany regulator główny $\mathbf{R_M}$. Regulator od zakłócenia $\mathbf{R_K}$ został dobrany przy pomocy funkcji AutoTuningu dostępnej w pakiecie Matlab/Simulink.

2. Struktura oraz uruchomienie projektu.

Aby uruchomić projekt należy rozpakować pliki w wybranym przez siebie folderze oraz uruchomić plik **main.mlx**. Projekt posiada następującą strukturę:



Rysunek 2 Struktura projektu w programie Matlab

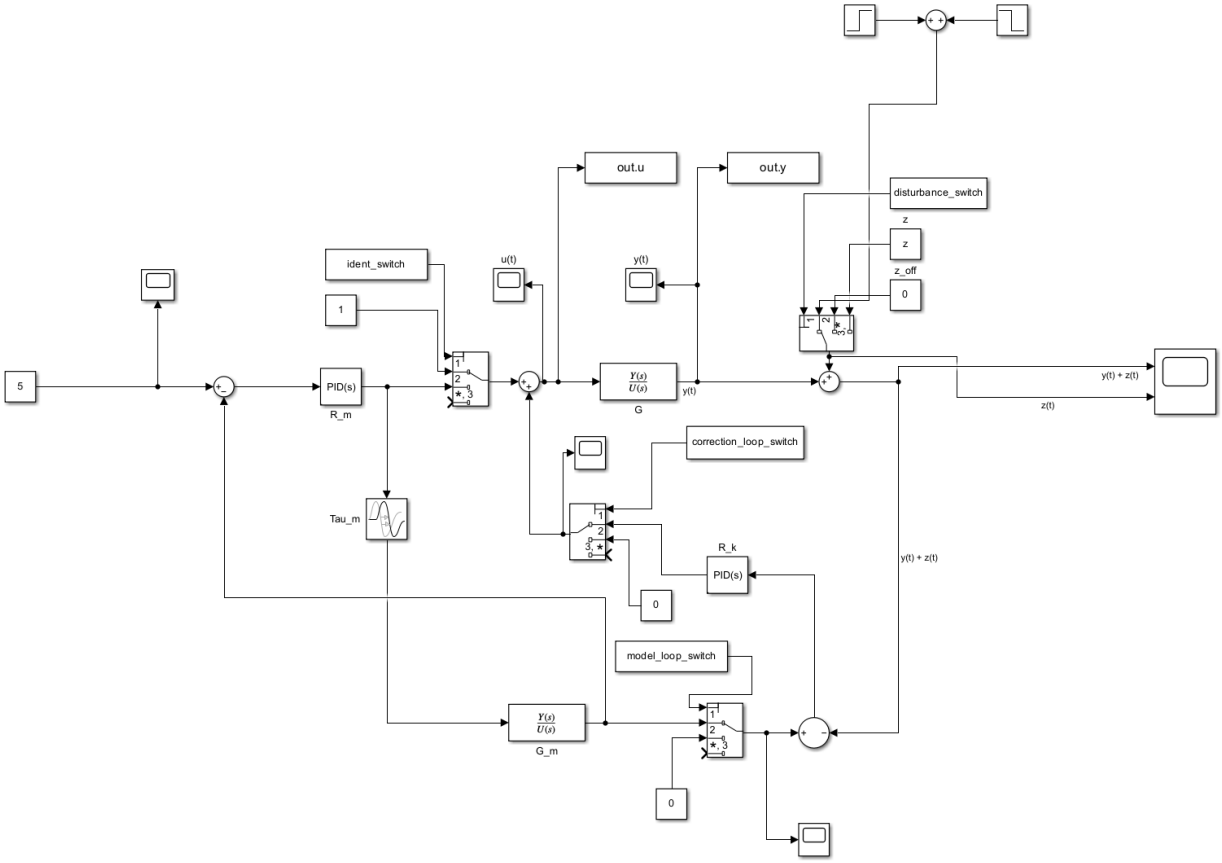
Projekt składa się z 4 plików:

- main.mlx – jest to program główny projektu.

- model_mfc.slx – jest to plik z symulacją w Simulinku.
- impulse_step_transfer_function.m – program z funkcją, która oblicza i zwraca impulsową oraz skokową funkcję przejścia.
- two_point_Strejc.m – program z funkcją, która na podstawie skokowej funkcji przejścia znajduje najlepiej dopasowany model Strejca. Zwraca ona rząd, opóźnienie oraz stałą czasową najlepszego modelu.

3. Działanie programu.

Podstawą działania naszego programu jest struktura MFC, która została utworzona w Simulinku:



Rysunek 3 Struktura MFC w Simulinku

Działanie naszego programu może zostać rozbite na 4 kroki:

- Inicjalizacja symulacji.
- Utworzenie impulsowej oraz skokowej funkcji przejścia.
- Identyfikacja obiektu modelem Strejca.
- Aktualizacja parametrów modelu G_M oraz dobór regulatorów.

3.1 Inicjalizacja symulacji.

Przed uruchomieniem programu inicjalizowana jest symulacja oraz wszystkie parametry obiektów i regulatorów.

```
clear all
%inicjalizacja zmiennych symulacji
%inicjalizacja obiektów
k = 1; % wzmocnienie obiektu
G_num = k;
G_denum = [720 1764 1624 735 175 21 1];

G_m_num = [1];
G_m_denum = [5 5 4 2];
tau_m = 0.1;

%inicjalizacja regulatorów
Kp_r_m = 1;
Ki_r_m = 1;
Kd_r_m = 0;

Kp_r_k = 0.5;
Ki_r_k = 0.2;
Kd_r_k = 0.1;

% zakłócenie
z = 0.5;

%inicjalizacja symulacji
sample_time = 0.01;

%inicjalizacja switchy
ident_switch = 1; % 1 - skok jednostkowy; 2 - sygnał z regulatora R_m
correction_loop_switch = 2; % 1 - pętla korekcyjna włączona, 2 - pętla
korekcyjna włączona
model_loop_switch = 2; % 1 - pętla z modelem włączona, 2 - pętla z modelem
włączona
disturbance_switch = 2; % 1 - zakłócenie włączone, 2 - zakłócenie wyłączone
```

Aby ułatwić oraz przyspieszyć symulację zastosowaliśmy bloczek Multiport Switch, dzięki któremu z poziomu programu głównego możemy zmieniać drogę sygnałów wewnątrz symulacji z Simulinka.

Jako obiekt główny G należało przyjąć obiekt o transmitancji:

$$G(s) = \frac{1}{(s+1)(2s+1)(3s+1)(4s+1)(5s+1)(6s+1)} = \frac{1}{720s^6 + 1764s^5 + 1624s^4 + 735s^3 + 175s^2 + 21s + 1}$$

Rysunek 4 Transmitancja obiektu głównego G

Wewnątrz symulacji przyjęliśmy okres próbkowania równy 10 ms. Parametry pozostałych obiektów w fazie inicjalizacji są przyjmowane jako losowe.

3.2 Utworzenie impulsowej oraz skokowej funkcji przejścia.

Po inicjalizacji symulacji na obiekt jest podawany skok jednostkowy oraz rejestrowana jest odpowiedź obiektu. W tym celu na wejście obiektu podajemy skok jednostkowy oraz wyłączamy działanie pętli korekcyjnej.

```
%ustawienie switchy na skok jednostkowy
ident_switch = 1 % 1 - skok jednostkowy; 2 - sygnał z regulatora R_m
correction_loop_switch = 2 % 1 - pętla korekcyjna włączona, 2 - pętla korekcyjna
włączona
```

Następnie na podstawie funkcji stepinfo() obliczany jest czas ustalania się odpowiedzi na obiekcie, dzięki czemu obliczamy czas trwania symulacji.

```
%ustawienie wstępnego czasu trwania symulacji
sys = tf(G_num, G_denum);
system_settling_time = stepinfo(sys).SettlingTime + 20;
stop_time = system_settling_time;
```

Uruchamiamy symulację oraz rejestrujemy sterowanie wchodzące na obiekt oraz jego odpowiedź:

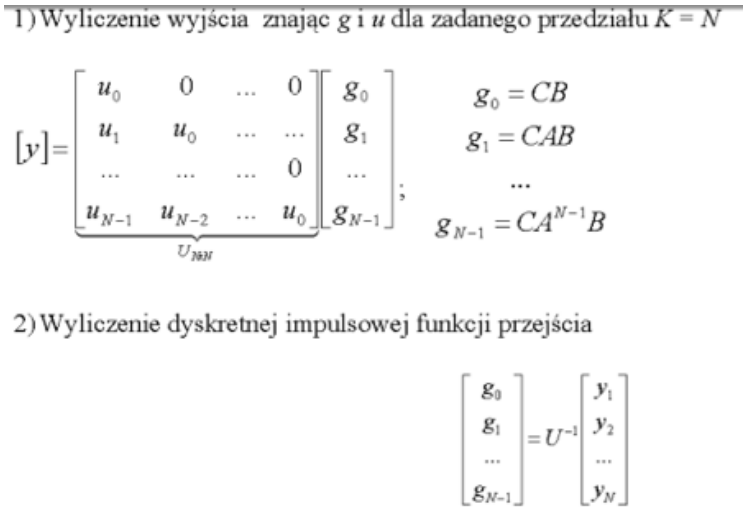
```
sim_out = sim("model_mfc.slx");
u = sim_out.u.Data;
y = sim_out.y.Data;
t = sim_out.tout;
```

Zarejestrowane wyjście, wejście, czas oraz obiekt do funkcji obliczającej impulsową oraz skokową funkcję przejścia:

```
[g, h] = impulse_step_transfer_function(u, y, t, sys);
```

Jako okno czasowe przyjęliśmy czas równy czasowi ustalania się odpowiedzi obiektu na skok jednostkowy.

Wewnątrz funkcji impulse_step_transfer_function() dyskretna impulsowa funkcja przejścia jest obliczana w następujący sposób:



Rysunek 5 Obliczanie dyskretnej impulsowej funkcji przejścia.

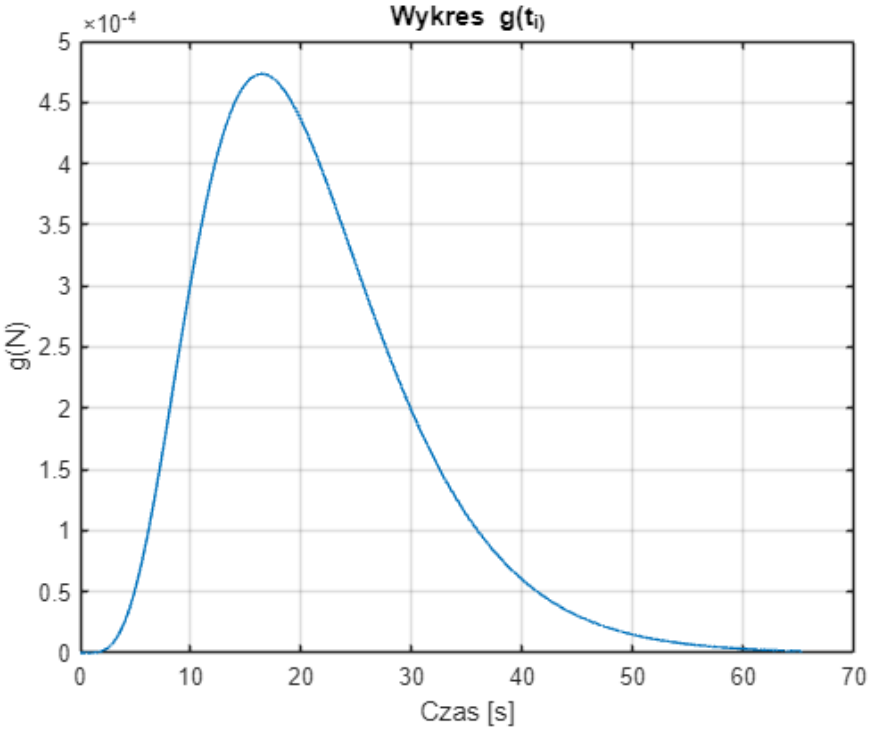
Dyskretna skokowa funkcja przejścia jest obliczana jako suma kumulacyjna impulsowej funkcji przejścia według wzoru:

$$h_k = \sum_{j=0}^{k} g_j$$

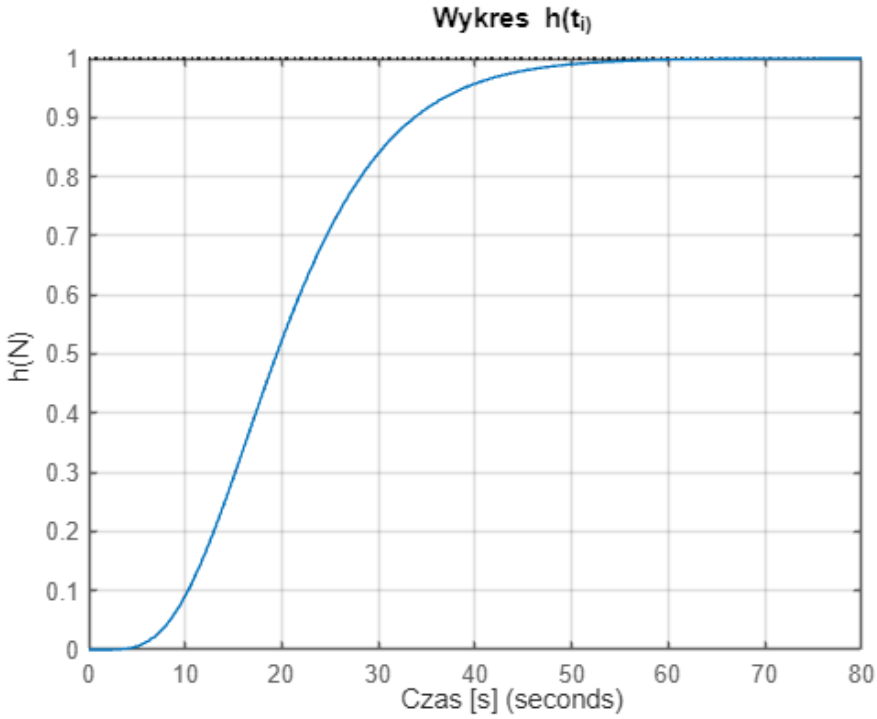
Dodatkowo wewnątrz funkcji wyświetlane są wykresy obu charakterystyk. Kod programu obliczający impulsową oraz skokową funkcję przejścia:

```
function [g, h] = impulse_step_transfer_function(u, y, t, sys)

    U_matrix = zeros(length(u), length(u));
    for i=1:length(u)
        for j=1:length(u)
            if i >= j
                U_matrix(i, j) = u(i - j + 1);
            end
        end
    end

    g = U_matrix\y;
    h = cumsum(g);

    figure
    plot(t, g)
    xlabel('Czas [s]')
    ylabel('g(N)')
    title('Wykres g(t_i)')
    grid on

    figure
    hold on
    plot(t, h, 'r')
    step(sys)
    hold off
    xlabel('Czas [s]')
    ylabel('h(N)')
    title('Wykres h(t_i)')
    grid on
end
```

Dla naszego obiektu uzyskaliśmy poniższej charakterystyki:



Rysunek 6 Dyskretna impulsowa funkcja przejścia.



Rysunek 7 Dyskretna skokowa funkcja przejścia

3.3 Identyfikacja parametrów obiektu przy pomocy dwupunktowej metody Strejca.

Mając wyznaczoną dyskretną skokową funkcję przejścia możemy przeprowadzić aproksymację obiektu modelem Strejca. Transmitancja modelu Strejca ma poniższą postać:

$$G_{strejc} = \frac{e^{-\tau s}}{(Ts + 1)^n}$$

Aby stworzyć model Strejca należy znaleźć najlepszą aproksymację naszego obiektu przy pomocy modelu Strejca. Dwupunktowa metoda Strejca składa się z kilku kroków, w których jest wykorzystywana poniższa tabela oraz skokowa funkcja przejścia:

n	***h_n***	***T_n***	***τ_n***	***τ_n***
2	0.264	$T_2 = 0.34605*(T_{90} - t_{02})$	$\tau_2 = t_{02} - T_2$	$\tau_2 = 1.34605*t_{02} - 0.34605*T_{90}$
3	0.323	$T_3 = 0.30099*(T_{90} - t_{03})$	$\tau_3 = t_{03} - 2T_3$	$\tau_3 = 1.60199*t_{03} - 0.60199*T_{90}$
4	0.353	$T_4 = 0.27168*(T_{90} - t_{04})$	$\tau_4 = t_{04} - 3T_4$	$\tau_4 = 1.81504*t_{04} - 0.81504*T_{90}$
5	0.371	$T_5 = 0.25040*(T_{90} - t_{05})$	$\tau_5 = t_{05} - 4T_5$	$\tau_5 = 2.00160*t_{05} - 1.00160*T_{90}$
6	0.384	$T_6 = 0.23393*(T_{90} - t_{06})$	$\tau_6 = t_{06} - 5T_6$	$\tau_6 = 2.16968*t_{06} - 1.16968*T_{90}$
7	0.394	$T_7 = 0.22065*(T_{90} - t_{07})$	$\tau_7 = t_{07} - 6T_7$	$\tau_7 = 2.3239*t_{07} - 1.3239*T_{90}$

Na początku, na charakterystyce skokowej odczytujemy chwilę czasową, w której wykres przekracza amplitudę 0.9. Następnie szukamy czasów, w których charakterystyka skokowa przekracza amplitudy znajdujące się w drugiej kolumnie tabeli – są to odpowiednio czasy od t_{02} do t_{06}. Mając wyznaczone te czasy obliczamy stałą czasową T oraz opóźnienie τ. Sprawdzamy kolejne wiersze dopóki opóźnienie τ nie będzie ujemne.

Identyfikacja obiektu modelem Strejca jest realizowana wewnątrz funkcji two_point_Strejc:

```
[n_m, tau_m, T_m] = two_point_Strejc(t, h, stop_time)
```

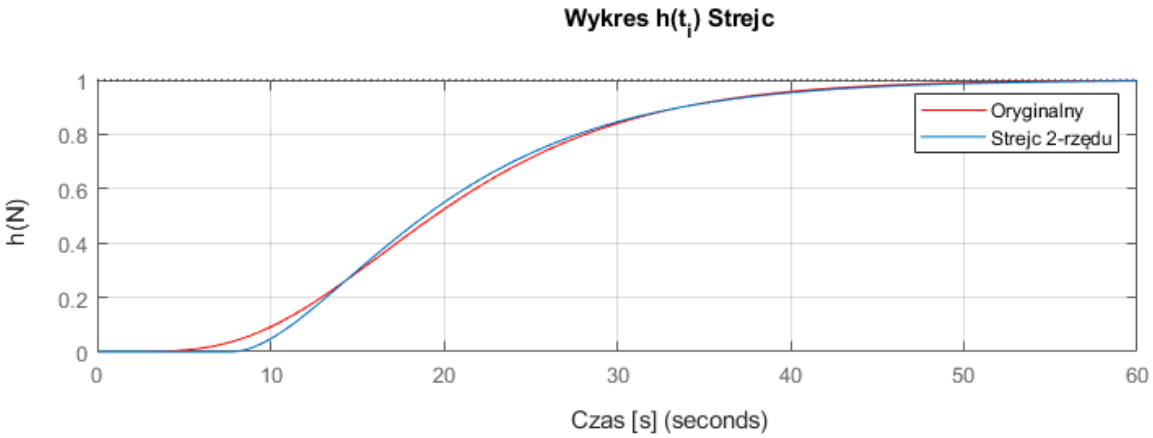
Jako argumenty przyjmuje ona wektor czasowy t, wartości skokowej funkcji przejścia oraz czas ustalania odpowiedzi obiektu. Funkcja wyświetla wszystkie znalezione opóźnienia obiektów od 2 do 7 rzędu. Następnie biorąc pod uwagę tylko dodatnie opóźnienia oblicza dla każdego obiektu całkę z kwadratu błędu oraz zwraca rząd, opóźnienie oraz stałą czasową najlepiej dopasowanego obiektu. Dodatkowo

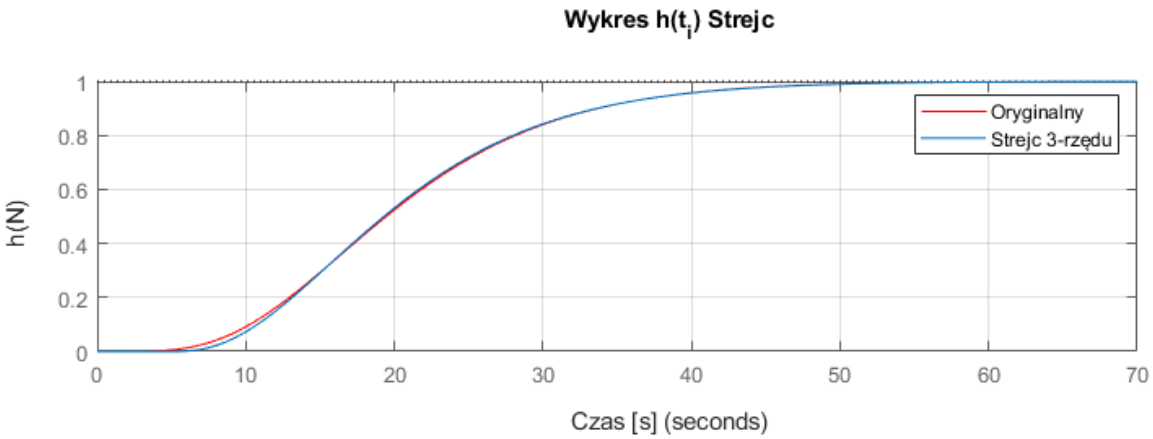
wyświetlany jest wykres porównujący poszczególne obiekty z dodatnim opóźnieniem z obiektem rzeczywistym.

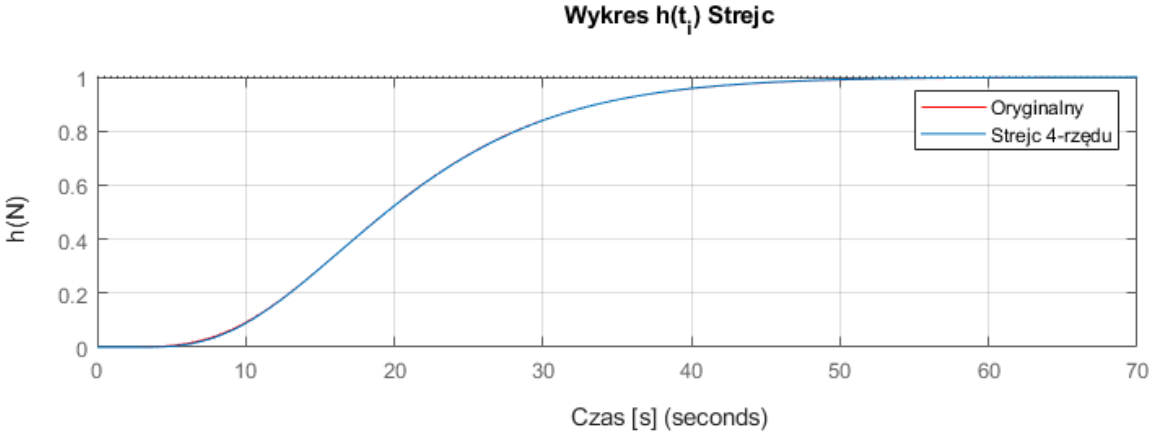
Przykład działania funkcji two_point_Strejc:

```
[n_m, tau_m, T_m] = two_point_Strejc(t, h, stop_time)
```

```
ans = "Znalezione opóźnienia obiektów"
tau_vec = 1×6
    7.6909    4.8107    2.0920   -0.3973   -2.6750   -4.7738
```








```
mean_squared_error = 1×3
    1.4146    0.2592    0.0078
```

```
ans = "Znalezione najlepsze parametry obiektu"
n_m = 4
tau_m = 2.0920
T_m = 4.7327
```

W tym przypadku najlepszy okazał się obiekt czwartego rzędu co potwierdza narysowany w funkcji wykres porównania tego modelu Strejca z obiektem rzeczywistym.

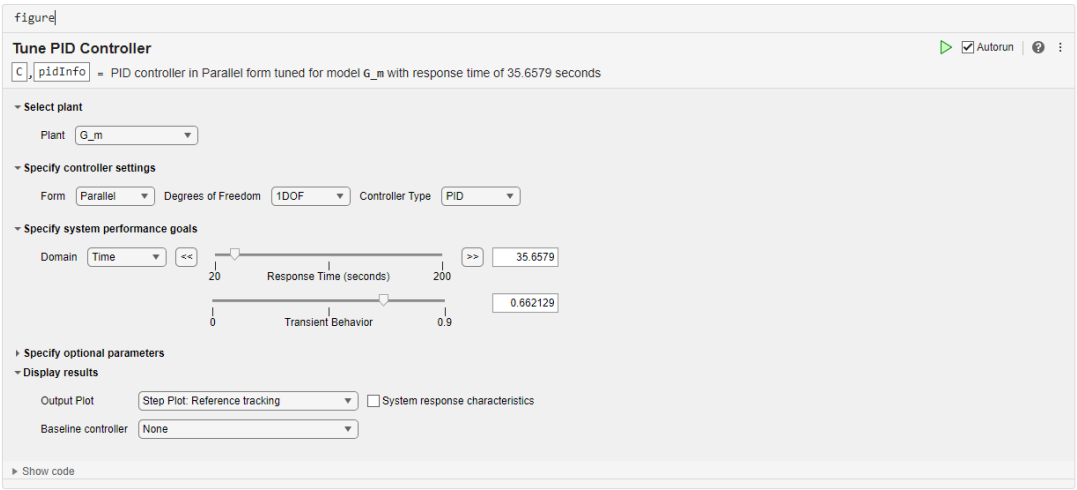
3.4 Aktualizacja parametrów modelu oraz dobór parametrów regulatorów.

Po dobraniu najlepszego modelu przy pomocy metody Strejca aktualizowane są parametry modelu G_M. Do stworzenia modelu wykorzystujemy parametry zwrócone w funkcji two_point_Strejc().
Tworzenie obiektu oraz jego transmitancja:

```
syms s
p = (T_m * s + 1)^(n_m);

G_m_num = k;
G_m_denum = sym2poly(p);
G_m = tf(G_m_num, G_m_denum, 'InputDelay',tau_m)
```

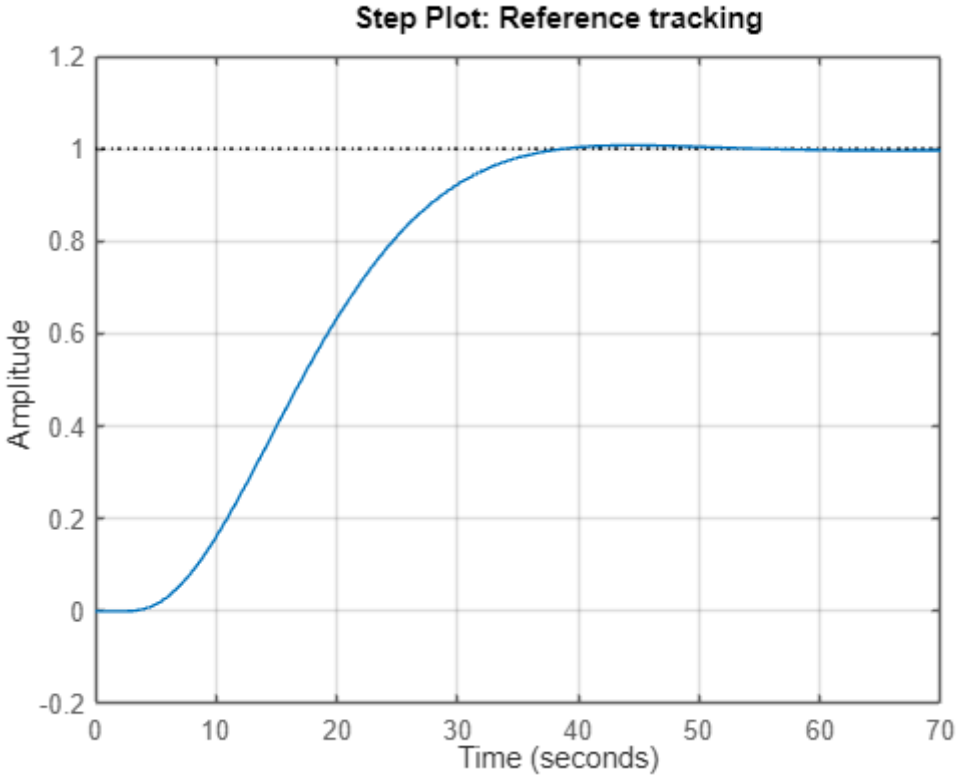
```
G_m =

                                  1
  exp(-2.09*s) * ---------------------------------------------
                 501.7 s^4 + 424 s^3 + 134.4 s^2 + 18.93 s + 1
```

Kolejnym krokiem jest dobór optymalnych nastaw regulatora R_M oraz R_K. Parametry regulatora R_K są dobierane przy pomocy narzędzia Tune PID Controller dostępnego w pakiecie Matlab/Simulink. W narzędziu należy podać obiekt do którego będzie strojony regulator PID. Poniżej przestawiony jest wygląd narzędzia do strojenia regulatora PID:



Rysunek 8 Tune PID Controller dla regulatora Rk

Dodatkowo w oknie obok wyświetla się odpowiedź obiektu z dobranym regulatorem.



Rysunek 9 Odpowiedź obieku Gm z dostrojonym regulatorem

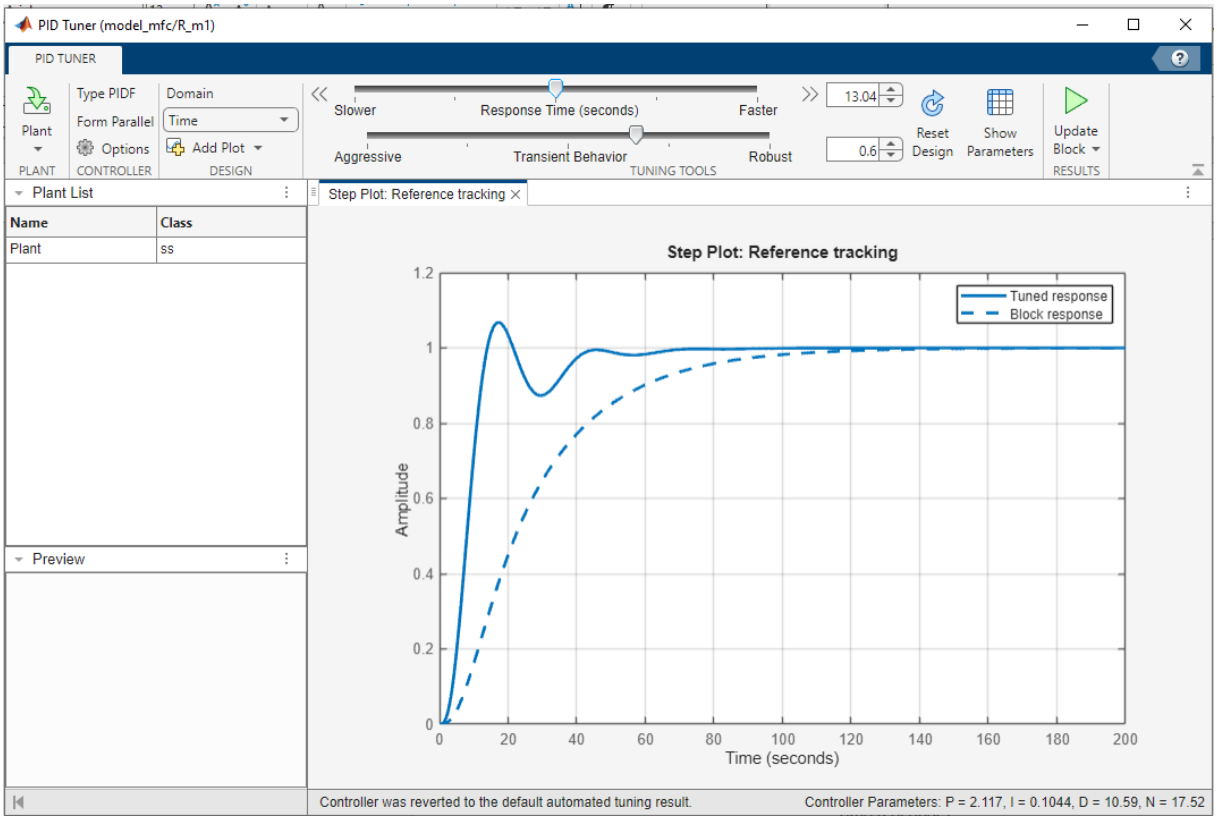
Regulator główny R_M jest strojony według poniższych reguł:

Dla modelu Strejca z n ≥1 formuły dla regulatora PI.
Dla modelu Strejca z n>2 formuły dla regulatora PID.

	Kp	Ti	Td
Regulator PI $R(s)=K_p(1+\frac{1}{T_i s})$	$\frac{1}{4k}\frac{n+2}{n-1}$	$\frac{T}{3}(n+2)$	
Regulator PID $R(s)=K_p(1+\frac{1}{T_i s}+T_d s)$	$\frac{1}{46k}\frac{7n+16}{n-2}$	$\frac{T}{15}(7n+16)$	$T\frac{n^2+4n+3}{7n+16}$

Rysunek 10 Wzory na parametry regulatora głównego Rm

Podczas realizacji projektu próbowaliśmy również przeprowadzić strojenie regulatora głównego R_M za pomocą opcji Autotuningu jednak okazało się, że parametry obliczone przy pomocy powyższych wzorów zapewniają asymptotyczny kształt sygnału sterującego przy dość optymalnym czasie regulacji. Narzędzie Autotuningu znalazło regulator z mniejszym czasem regulacji, lecz w sygnale sterującym występowało przeregulowanie. Całość została pokazana na poniższym obrazie:



Rysunek 11 Próba dostrojenia regulatora Rm przy pomocy Autotuningu

Następnie aktualizujemy parametry regulatorów R_K oraz R_M. Znalezione nastawy regulatora R_M:

```
Kp_r_m = (1/(46 * k)) * ((7*n_m + 16)/(n_m - 2))
```

```
Kp_r_m = 0.4783
```

```
Ti_r_m = (T_m / 15)*(7*n_m + 16)
```

```
Ti_r_m = 13.8825
```

```
Ki_r_m = Kp_r_m / Ti_r_m
```

```
Ki_r_m = 0.0345
```

```
Td_r_m = T_m * ((n_m^2 + 4*n_m + 3)/(7*n_m + 16))
```

```
Td_r_m = 3.7646
```

```
Kd_r_m = Kp_r_m * Td_r_m
```

```
Kd_r_m = 1.8005
```

Znalezione nastawy regulatora R_K

```
Kp_r_k = C.Kp
```

```
Kp_r_k = 0.7727
```

```
Ki_r_k = C.Ki
```

```
Ki_r_k = 0.0559
```

```
Kd_r_k = C.Kd
```

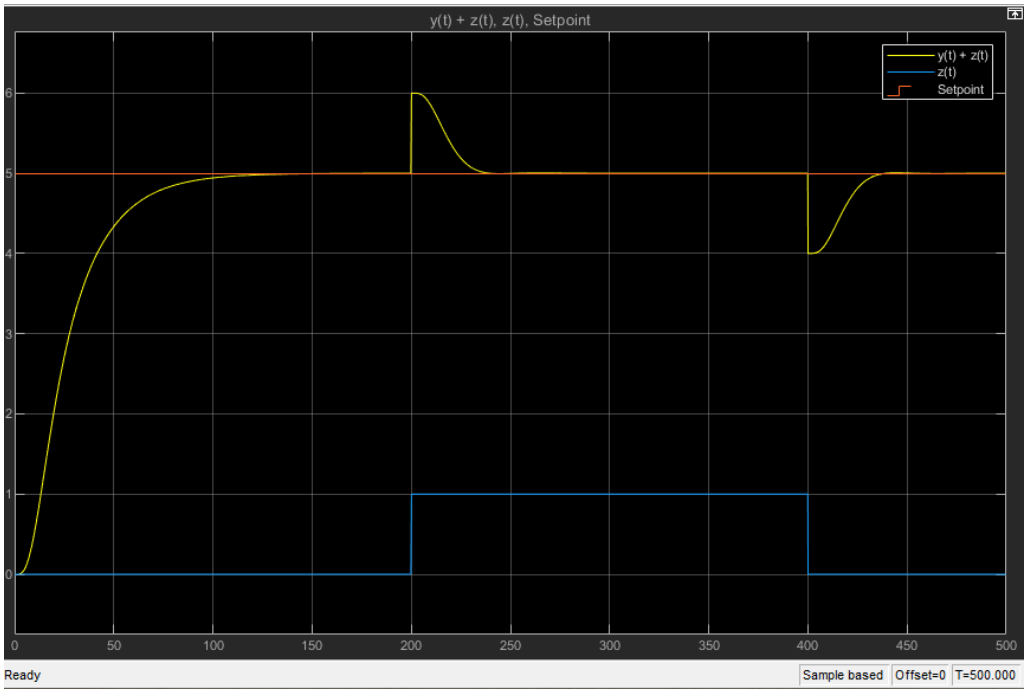
```
Kd_r_k = 2.6718
```

3.5 Włączenie pętli korekcyjnej oraz dodanie zakłócenia na wyjście.

Po doborze parametrów regulatorów możemy sprawdzić działanie struktury Model Following Control na zewnętrzne zakłócenie w postaci skoku jednostkowego. Na początku jednak musimy ustawić wszystkie przełączniki w odpowiedniej pozycji tj. włączyć pętlę modelową oraz korekcyjną i włączyć zakłócenie. Czas symulacji ustawiamy na 500 sekund.

```
ident_switch = 2; % 1 - skok jednostkowy; 2 - sygnał z regulatora R_m
correction_loop_switch = 1; % 1 - pętla korekcyjna włączona, 2 - pętla
korekcyjna wyłączona
model_loop_switch = 1; % 1 - pętla z modelem włączona, 2 - pętla z modelem
wyłączona
disturbance_switch = 1; % 1 - zakłócenie włączone, 2 - zakłócenie wyłączone
stop_time = 500;
sim("model_mfc.slx")
```

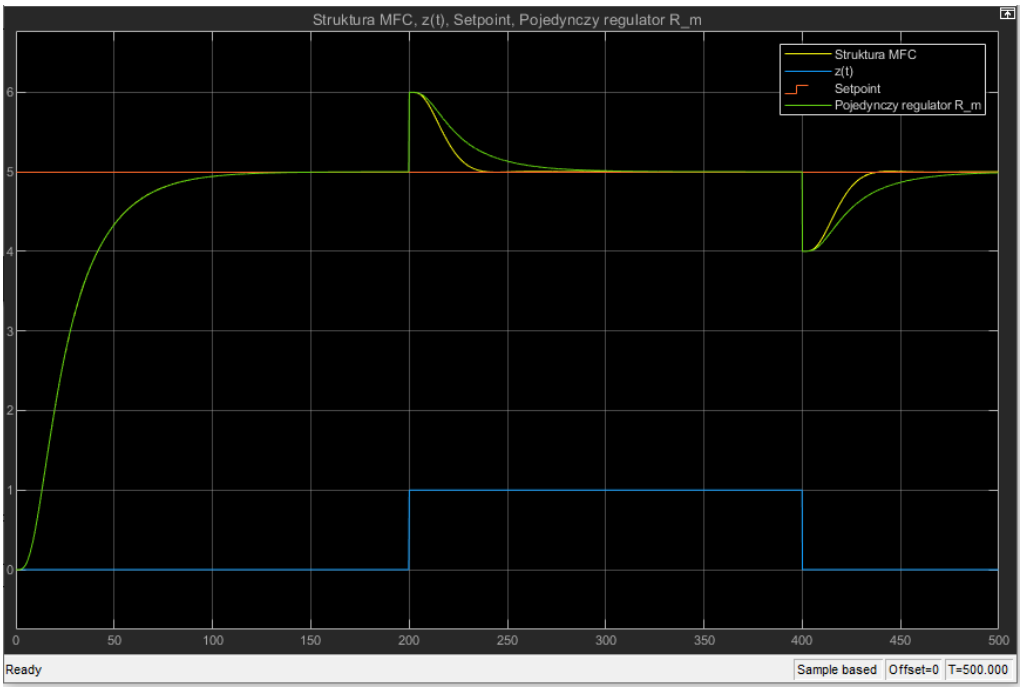
Jako zakłócenie przyjęliśmy skok jednostkowy, który jest uruchamiany w 200 sekundzie oraz wyłączany w 400 sekundzie.



Rysunek 12 Działanie struktury MFC

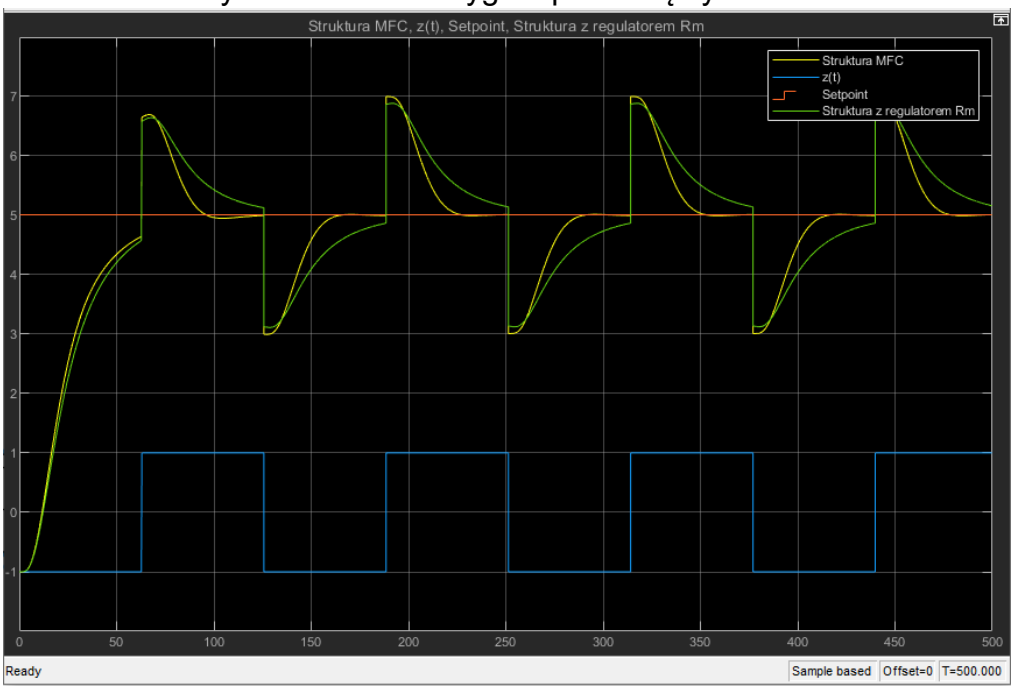
3.6 Testy struktury MFC.

Porównaliśmy działanie struktury MFC ze strukturą z pojedynczym regulatorem R_M.

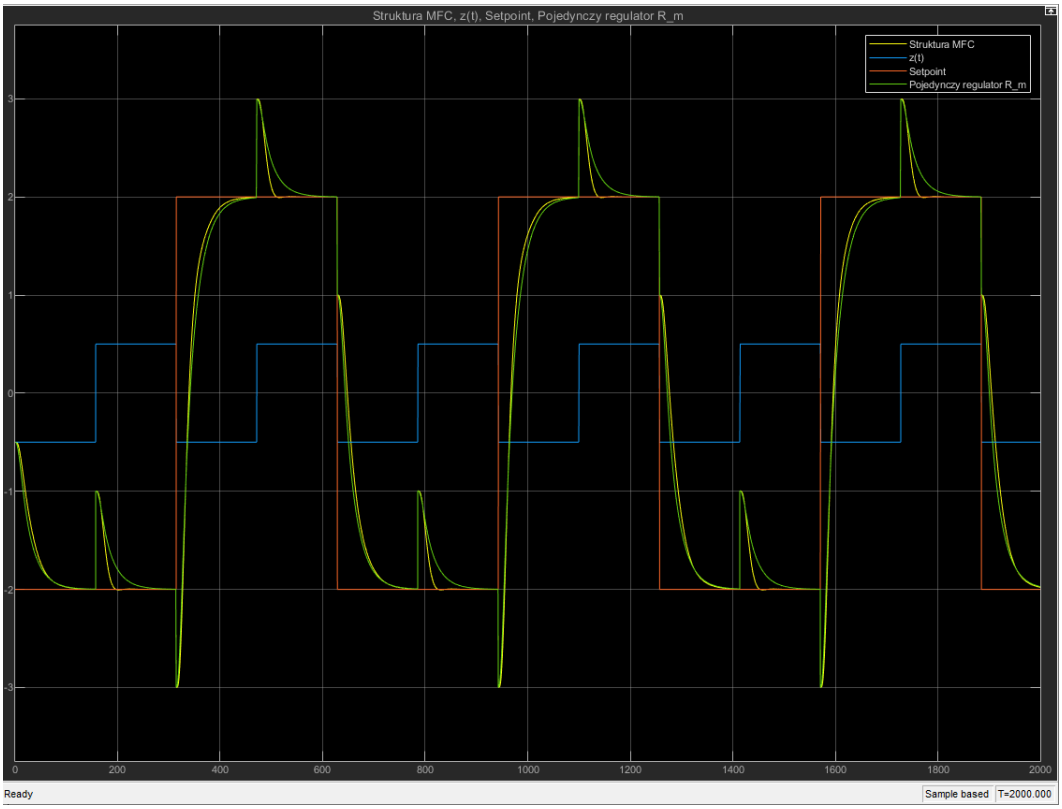


Rysunek 13 Porównanie struktury MFC z układem z pojedynczym regulatorem

Struktura z układem MFC charakteryzuje się o wiele mniejszym czasem regulacji uchybu co zostało przedstawione na powyższym wykresie.
Dodatkowo zmieniliśmy zakłócenie na sygnał prostokątny:



Rysunek 14 Sygnał zakłócający w postaci sygnału prostokątnego.



Rysunek 15 Działanie struktury z prostokątnym sygnałem referencyjnym i zakłócającym

3.7 Nowy obiekt.

Następnie zmieniamy obiekt na obiekt o trasmitancji:

$$G(s) = \frac{1}{(s+1)(2s+1)(3s+1)(4s+1)(7s+1)(9s+1)}$$

Rysunek 16 Transmitancja nowego obiektu

Po zmianie obiektu powtarzamy kroki wykonywane w podpunktach od 3.2 do 3.5. Wyznaczamy impulsową i skokową funkcję przejścia, identyfikujemy obiekt, dobieramy regulatory, aktualizujemy model i parametry regulatorów.
Stworzenie nowej transmitancji obiektu rzeczywistego:

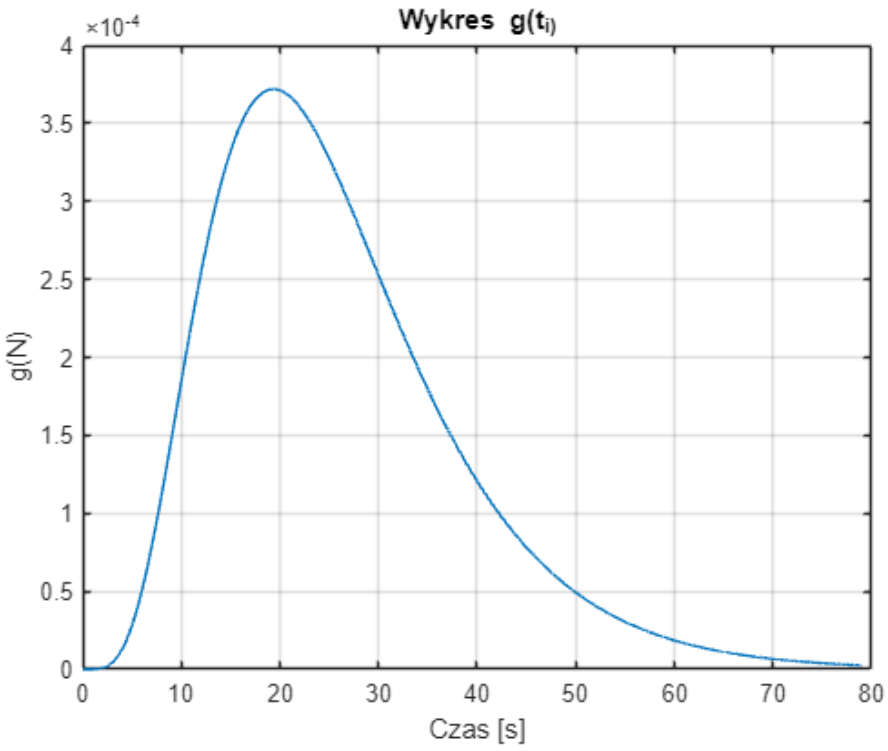
```
p = (s + 1)*(2*s + 1)*(3*s + 1)*(4*s + 1)*(7*s + 1)*(9*s + 1);

G_num = 1;
G_denum = sym2poly(p)
```

```
G_denum = 1×7
        1512        3534        3029        1240         258          26 …
```

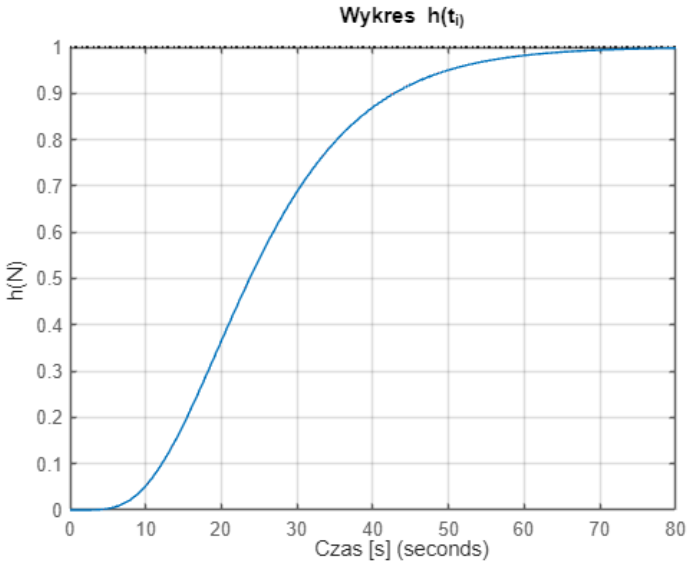
Obliczenie impulsowej i skokowej funkcji przejścia:

```
 %ustawienie switchy na skok jednostkowy

 ident_switch = 1 % 1 - skok jednostkowy; 2 - sygnał z regulatora R_m
 correction_loop_switch = 2 % 1 - pętla korekcyjna włączona, 2 - pętla
korekcyjna włączona
 model_loop_switch = 2 % 1 - pętla z modelem włączona, 2 - pętla z modelem
włączona

 %ustawienie wstępnego czasu trwania symulacji
 sys = tf(G_num, G_denum);
 system_settling_time = stepinfo(sys).SettlingTime + 20;
 stop_time = system_settling_time;

 sim_out = sim("model_mfc.slx");
 u = sim_out.u.Data;
 y = sim_out.y.Data;
 t = sim_out.tout;

 [g, h] = impulse_step_transfer_function(u, y, t, sys);
```

Otrzymane charakterystyki nowego obiektu:



Rysunek 17 Charakterystyka impulsowa nowego obiektu

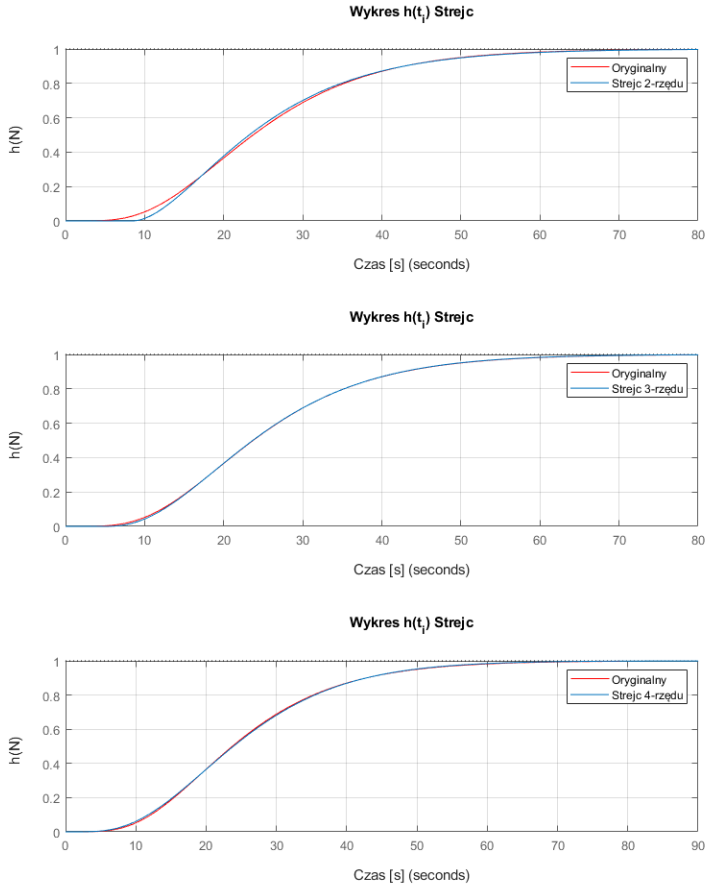


Rysunek 18 Charakterystyka skokowa nowego obiektu

Następnie identyfikujemy nowy obiekt modelem Strejca:

```
[n_m, tau_m, T_m] = two_point_Strejc(t, h, stop_time);
```

```
ans = "Znalezione opóźnienia obiektów"
tau_vec = 1×6
    8.4019    4.4717    0.7981   -2.5362   -5.5815   -8.3923
```



Rysunek 19 Porównanie różnych rzędów modelu Strejca

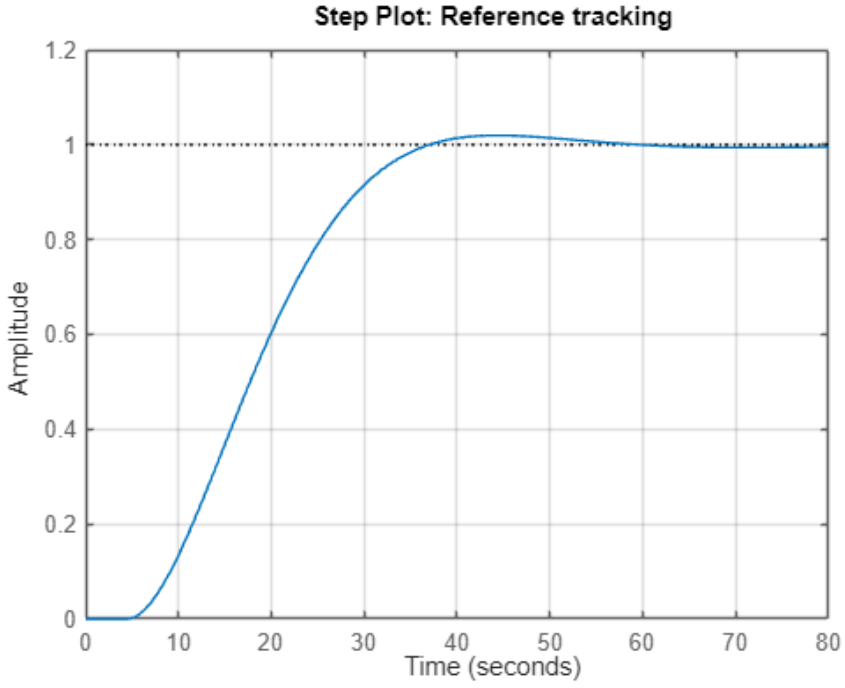
```
mean_squared_error = 1×3
    1.0408    0.0600    0.1228
ans = "Znalezione najlepsze parametry obiektu"
n_m = 3
tau_m = 4.4717
T_m = 7.1842
```

W tym przypadku najlepszy okazał się model Strejca trzeciego rzędu, ponieważ posiada on mniejszą całkę z kwadratu błędu. Można to zaobserwować porównując wykres modelu trzeciego oraz czwartego.
W kolejnym kroku aktualizujemy obiekt G_M parametrami modelu Strejca trzeciego rzędu:

```
syms s
p = (T_m * s + 1)^(n_m);

G_m_num = k;
G_m_denum = sym2poly(p);
G_m = tf(G_m_num, G_m_denum, 'InputDelay',tau_m)
```

```
G_m =

                                1
  exp(-4.47*s) * -----------------------------------
                 370.8 s^3 + 154.8 s^2 + 21.55 s + 1
```

Ponownie dobieramy nowy regulator R_K przy pomocy funkcji Tune PID Controller:



Rysunek 20 Odpowiedź obiektu Gm z dostrojonym regulatorem Rk

Oraz obliczamy parametry regulatora R_M:

```
Kp_r_m = (1/(46 * k)) * ((7*n_m + 16)/(n_m - 2))
```

```
Kp_r_m = 0.8043
```

```
Ti_r_m = (T_m / 15)*(7*n_m + 16)
```

```
Ti_r_m = 17.7209
```

```
Ki_r_m = Kp_r_m / Ti_r_m
```

```
Ki_r_m = 0.0454
```

```
Td_r_m = T_m * ((n_m^2 + 4*n_m + 3)/(7*n_m + 16))
```

```
Td_r_m = 4.6600
```

```
Kd_r_m = Kp_r_m * Td_r_m
```

```
Kd_r_m = 3.7483
```

Znalezione nowe nastawy dla regulatora R_K

```
Kp_r_k = C2.Kp
```

```
Kp_r_k = 1.0011
```

```
Ki_r_k = C2.Ki
```

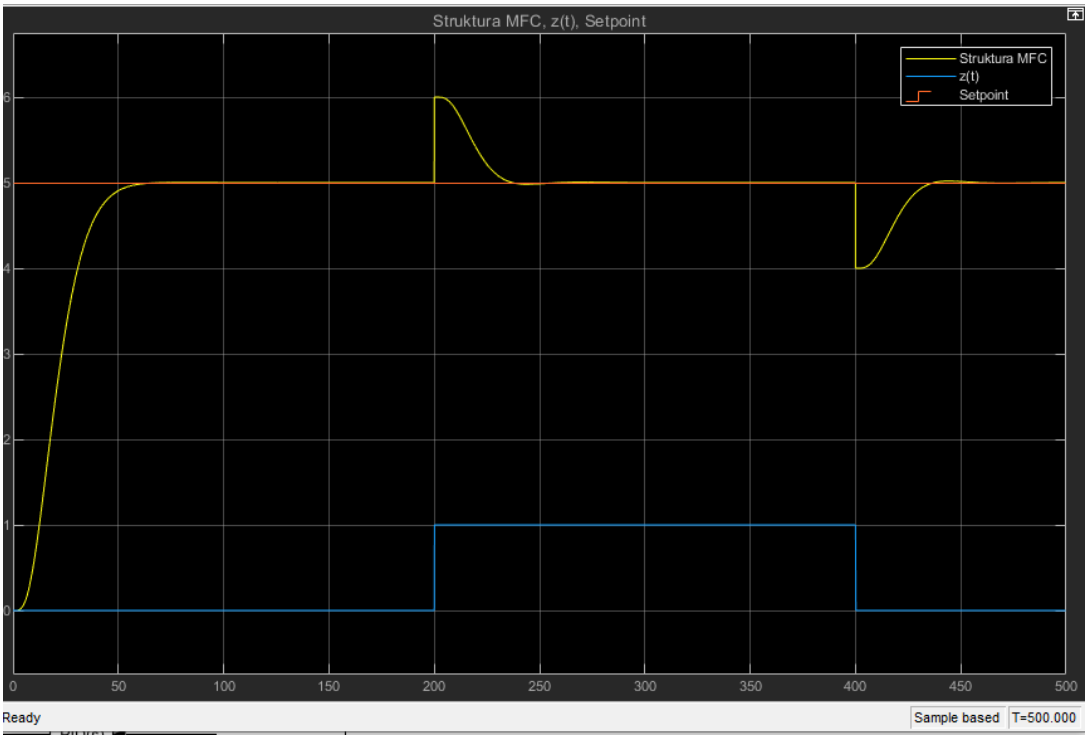
```
Ki_r_k = 0.0545
```

```
Kd_r_k = C2.Kd
```

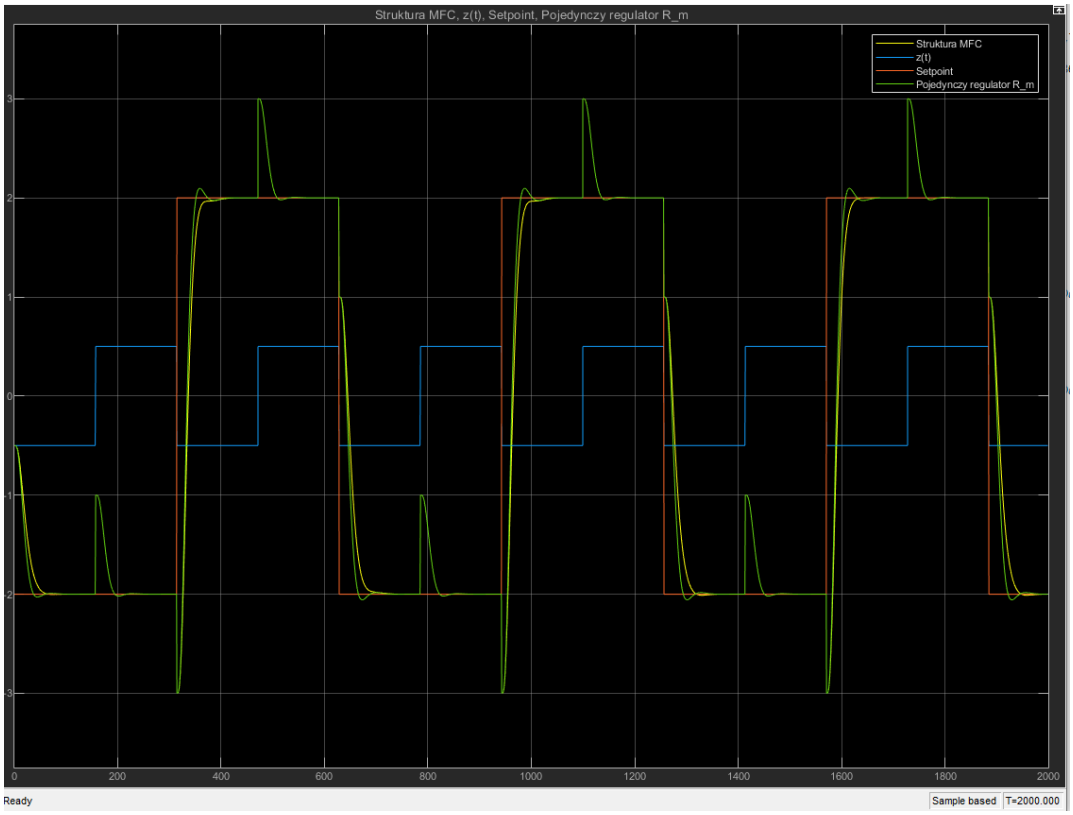
```
Kd_r_k = 4.5998
```

Na końcu zmieniamy położenia przełączników oraz uruchamiamy symulację.

Uzyskane wyniki:



Rysunek 21 Symulacja dla nowego obiektu.



Rysunek 22 Porównanie z pojedynczym regulatorem Rm

Jak widać identyfikacja obiektu modelem Strejca oraz dobór nowych parametrów regulatorów działają poprawnie. Podczas doboru parametrów regulatorów staraliśmy się zachować aperiodyczny charakter odpowiedzi układu na sygnał sterujący.

4. Podsumowanie oraz wnioski.

Układ Model Following Control jest połączeniem klasycznego zagadnienia strojenia regulatorów PID z teorią sterowania adaptacyjnego. Znajomość modelu matematycznego sterowanego obiektu jest szeroko wykorzystywana w metodach testowania układów takich jak Software in the loop (SIL) lub Hardware in the loop (HIL). Dzięki znajomości modelu matematycznego jesteśmy w stanie dostrajać parametry regulatorów w czasie rzeczywistym, dzięki czemu cały układ regulacji staje się dużo odporniejszy na zewnętrzne zakłócenia, które oddziałują nie tylko na proces, ale również na parametry obiektu regulacji. Oprócz tego układ dużo szybciej usuwa uchyb z procesu regulacji.

Zaletą struktury MFC jest również usunięcie uchybu występującego pomiędzy obiektem rzeczywistym, a modelem. Dzięki temu układ sterowania zachowa się dokładnie tak jak zachowałby się znany nam model matematyczny.

Zrealizowany projekt pozwolił nam dogłębnie zapoznać się z działaniem struktury MFC oraz zastosowaniem w praktyce identyfikacji obiektu przy pomocy dwupunktowej metody Strejca. Ugruntowaliśmy dodatkowo naszą wiedzę z zakresu strojenia regulatorów PID.

W przyszłości powyższy projekt można udoskonalić stosując obliczenia równoległe na kilku wątkach procesora. Przykładowo na jednym wątku uruchomiony byłby proces identyfikacji parametrów obiektu oraz obliczania optymalnych nastaw regulatorów. Na dwóch kolejnych wątkach uruchomione zostałyby dwa niezależnie działające regulatory - główny i korekcyjny.