# STP - Projekt 1

Mariusz Słapek

Grudzień 2018

## 1 Transmitancja modelu

Proces dynamiczny opisany jest transmitancją ciągła(stałe czasowe w sekundach):

$$\frac{4\,s^2 + 16\,s + 7}{4\,s^3 + 60\,s^2 + 296\,s + 480} \tag{1}$$

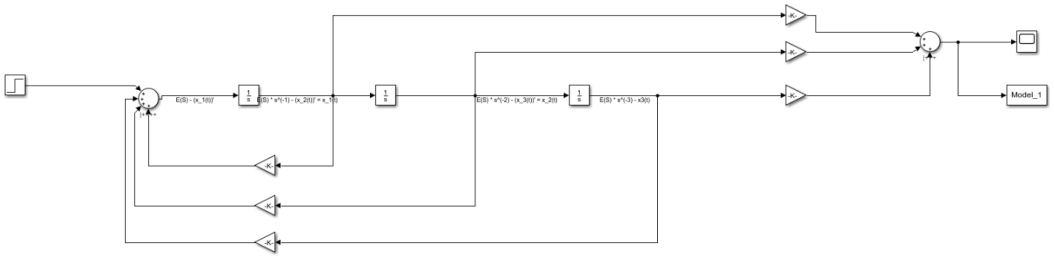
## 2 Zadanie 1

W programie Matlab zostały wyznaczone dwie wersje modeli ciałych w przestrzeni stanu. W tym celu skorzystałem z następujących funkcji:

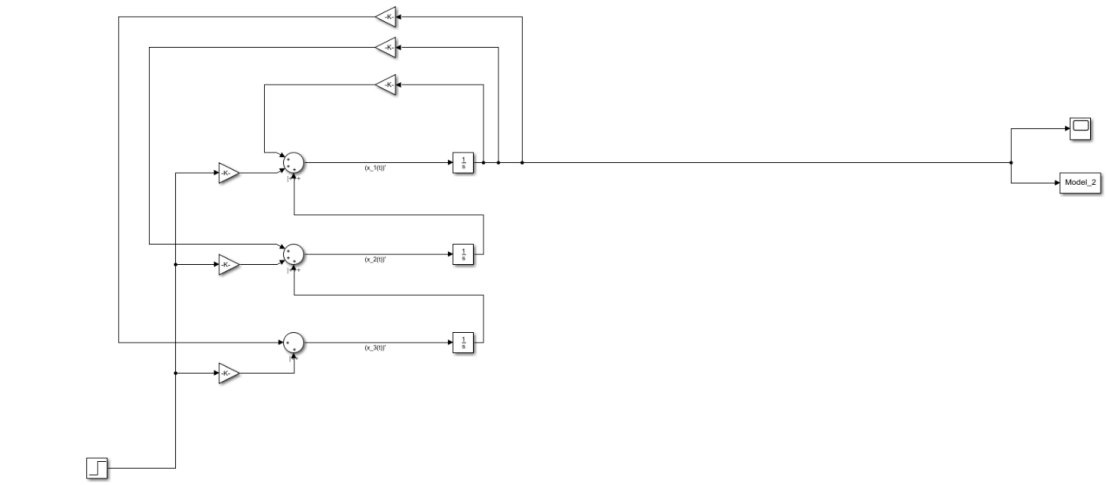
- *tf2ss(licznik, mianownik)* - funkcja, która jako argumnety przyjmuje wartości współczynników w liczniku transmitancji oraz wartości współczynników w mianowniku transmitancji. Funkcja zwraca macierze *A, B, C, D*. Są one macierzami wynikowymi dla pierwszej metody bezpośredniej. Aby wyznaczyć model drugi korzystamy z danych działań:

$$\begin{aligned} A_2 &= A_1^T \\ B_2 &= C_1^T \\ C_2 &= B_1^T \\ D_2 &= D_1 \end{aligned} \tag{2}$$

Rysunki reprezentacji graficznych modeli zostały przedstawione poniżej:




Rysunek 1: Reprezentacja graficzna modelu 1.




Rysunek 2: Reprezentacja graficzna modelu 2.

Dokładniej wykresy (przyjęte warunki) modeli można zobaczyć w plikach: *model_1.slx* oraz *model_2.slx*.

# 3 Zadanie 2

Aby wykazać symbolicznie, iż te dwa modele w przestrzeni stanu można sprowadzić do tej samej transmitancji stworzyłem skrypt w programie Matlab (dane macierze A, A_2 itd. są stworzone w skypcie *zad1.m*):

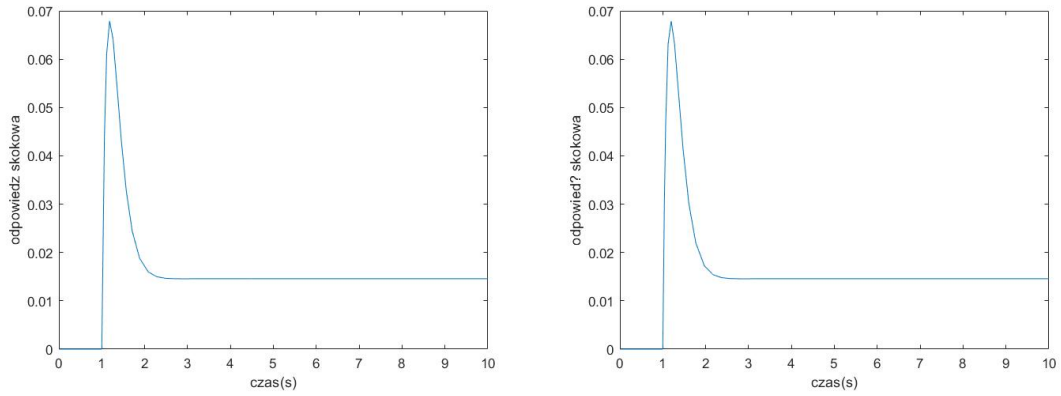
```
% Wykazac symbolicznie, ce obie wersje modelu w przestrzeni stanu mocna
% sprowadzic do tej samej transmitancji

syms s;

disp(size(A));
disp(size(A_2));

G_s1 = C * ((s*eye(size(A)) - A)^(-1)) * B + D;
G_s2 = C_2 * ((s*eye(size(A_2)) - A_2)^(-1)) * B_2 + D_2;

G_s1 = collect(G_s1);
G_s2 = collect(G_s2);

if G_s1 == G_s2
    disp("Transmitancje sa takie same");
else
    disp("Transmitancje sa rozne");
end

% out: Transmitancje sa takie same
```

Program po wykonaniu wyświetlił komunikat: "Transmitancje sa takie same". Oznacza, iż te dwa modele w przestrzeni stanu można sprowadzić do tej samej transmitancji.

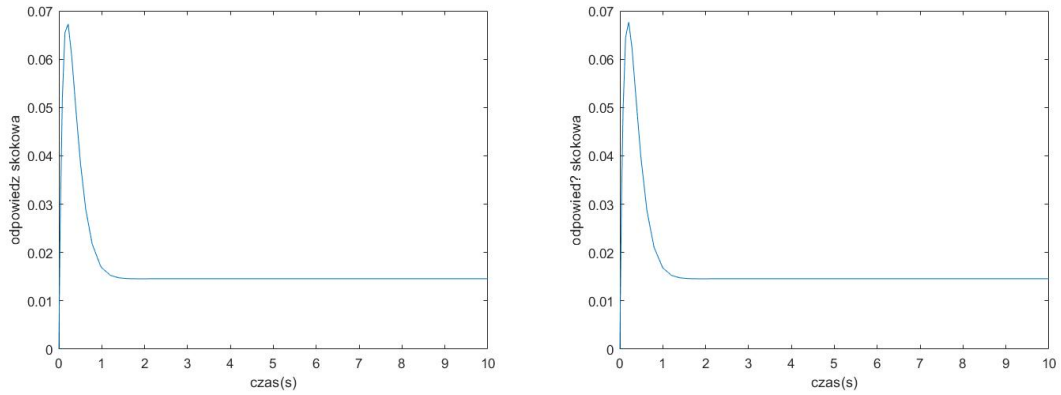
# 4 Zadanie 3

W celu porównania odpowiedzi skokowej transmitancji obu modeli wykonałem szereg eksperymentów. Symulacje przeprowadziłem zarówno dla zerowych i niezerowych warunków początkowych.

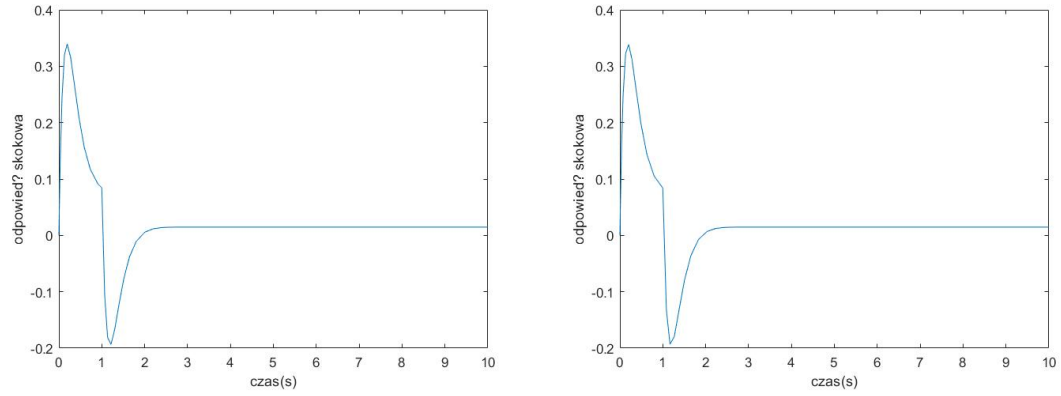
Poniższe zdjęcia przedstawiają odpowiedź skokową dla modelu pierwszego (po lewej stronie) oraz dla modelu drugiego (po prawej stronie):



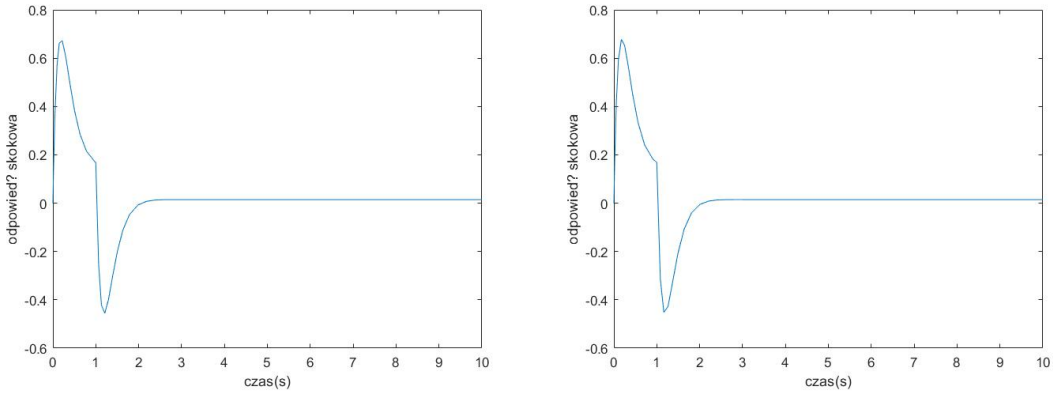

Rysunek 3: Opowiedź skokowa (wyjście) transmitancji obu modeli dla zerowych warunków początkowych




Rysunek 4: Opowiedź skokowa (wyjście) transmitancji obu modeli dla warunków początkowych: 1




Rysunek 5: Opowiedź skokowa (wyjście) transmitancji obu modeli dla warunków początkowych: 5



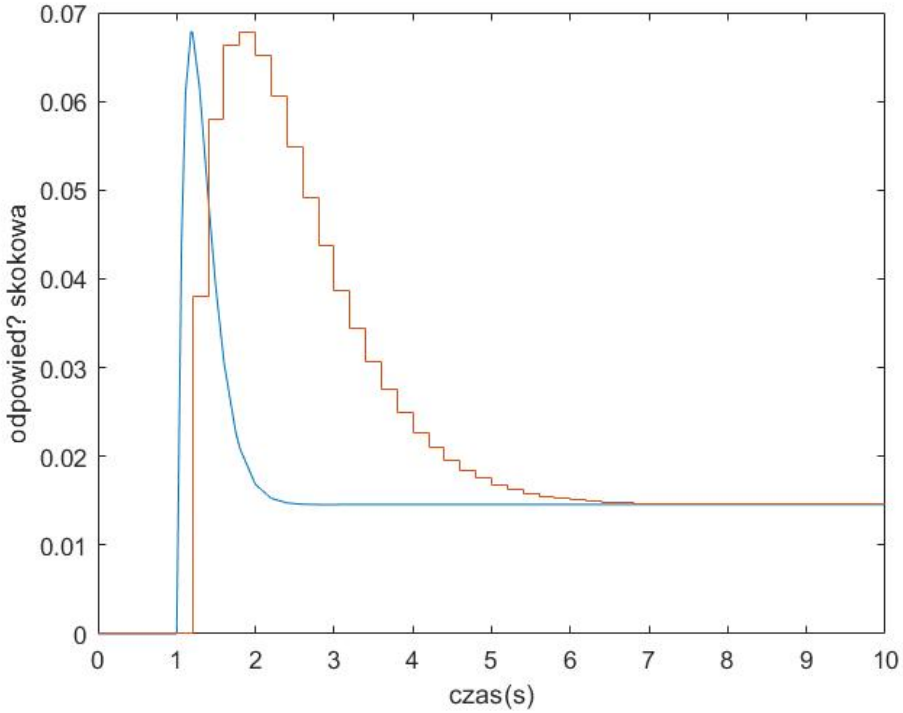

Rysunek 6: Opowiedź skokowa (wyjście) transmitancji obu modeli dla warunków początkowych: 10

Jak widzimy na wyżej załączonych rysunkach odpowiedzi skokowe dla tych samych warunków początkowych dla dwóch modeli są identyczne (kolejne potwierdzenie, iż to ta sama transmitancja).

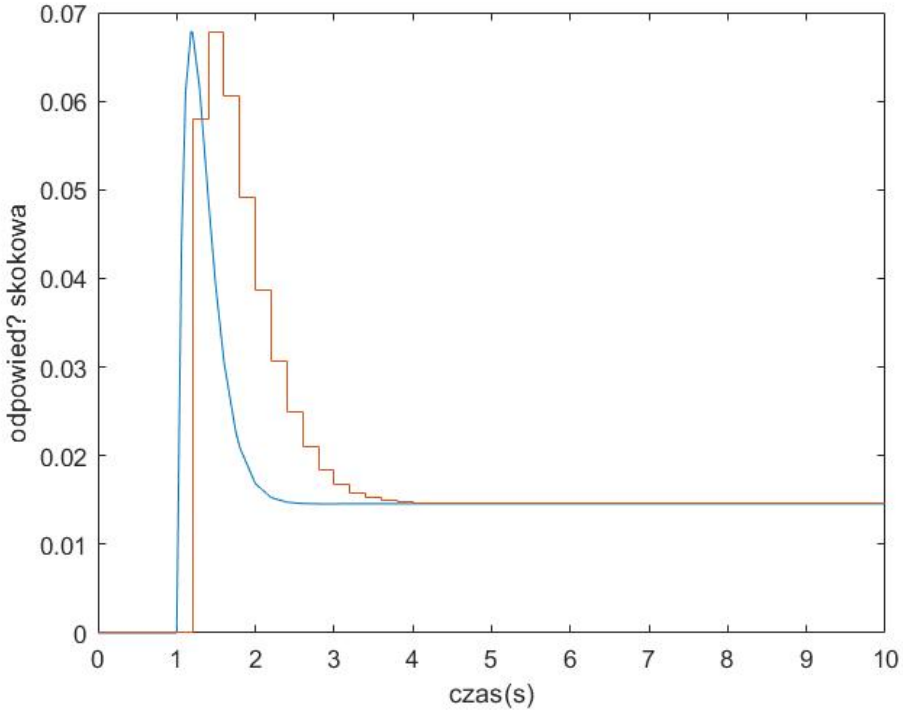
# 5 Zadanie 4

Transmitancje dyskretne *G(z)*, które odpowiadają transmitancji ciągłej *G(s)* wyznaczyłem przy pomocy programu Matlab za pomocą następujących funkcji:

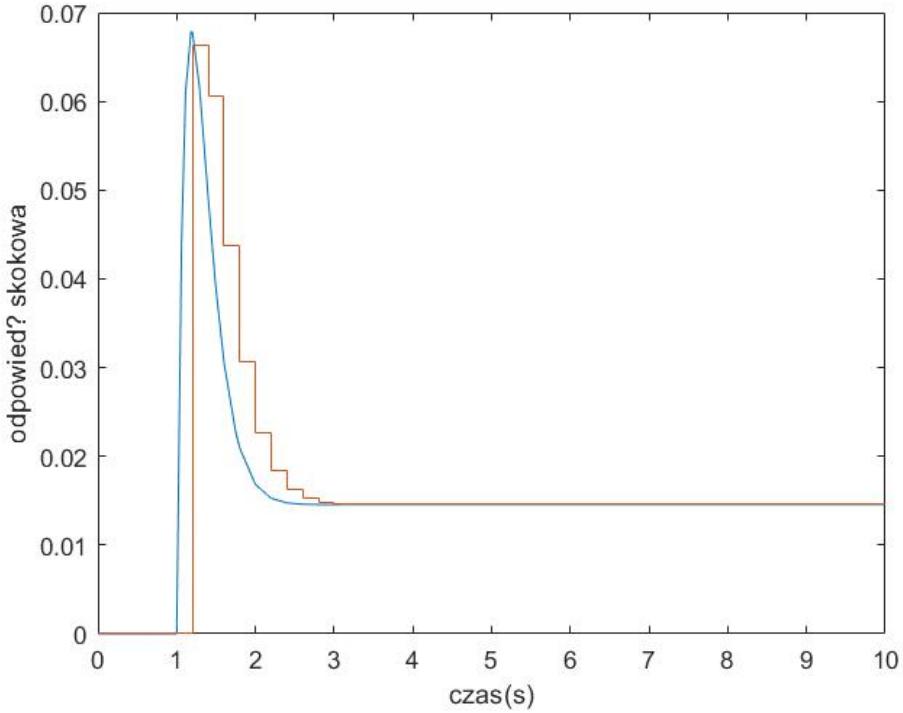
- *ss2tf(A, B, C, D)* - funkcja, która jako argumenty wejściowe przyjmuje macierze *A, B, C, D*, natomiast *output* tej funkcji to wartości współczynników w liczniku transmitancji oraz wartości współczynników w mianowniku transmitancji
- *c2d(G, T_1, 'zoh')* - funkcja, która jako argumenty wejściowe przyjmuje transmitancje, okres próbkowania (T_1) oraz metodę ekstrapolacji. W zadaniu mieliśmy przyjąć, iż metoda ekstrapolacji to ekstrapolator zerowego rzędu. *Output* naszej funkcji to transmitancja dyskretna odpowiadająca transmitancji ciągłej *G(s)*



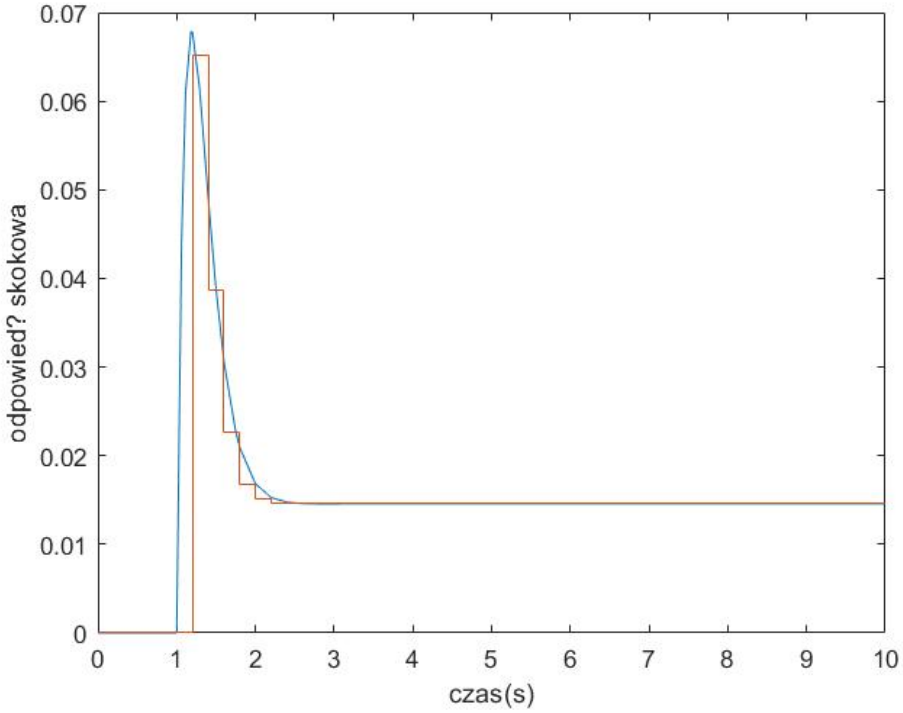

Rysunek 7: Odpowiedzi skokowe transmitancji ciagłej i dyskretnej przy zmianie sygnału wejściowego z 0 na 1 w chwili 1 s dla okresu próbkowania T=0.05s



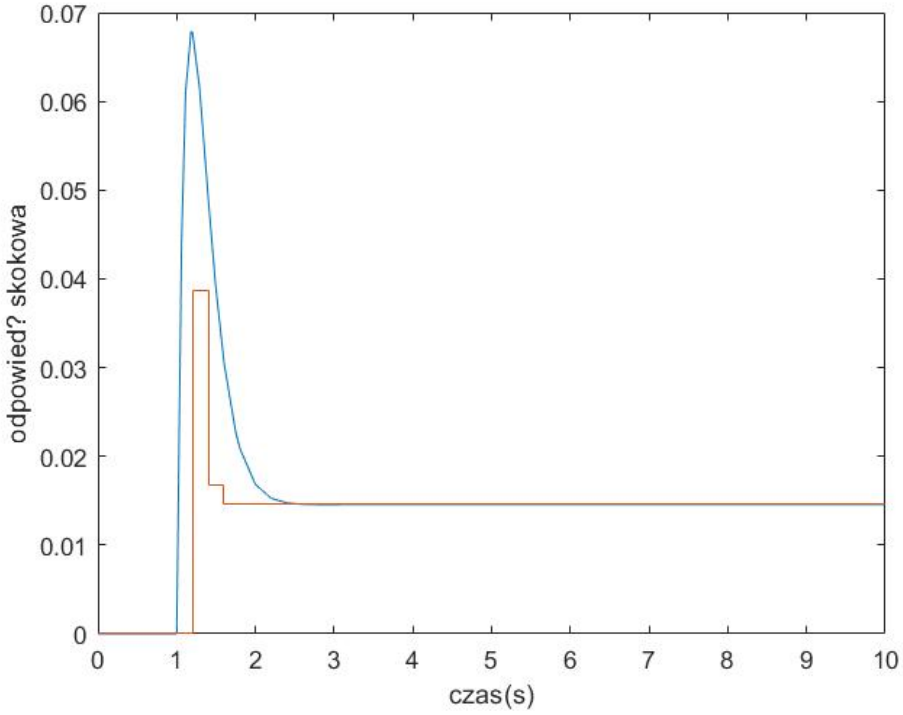

Rysunek 8: Odpowiedzi skokowe transmitancji ciągłej i dyskretnej przy zmianie sygnału wejściowego z 0 na 1 w chwili 1 s dla okresu próbkowania T=0.1s



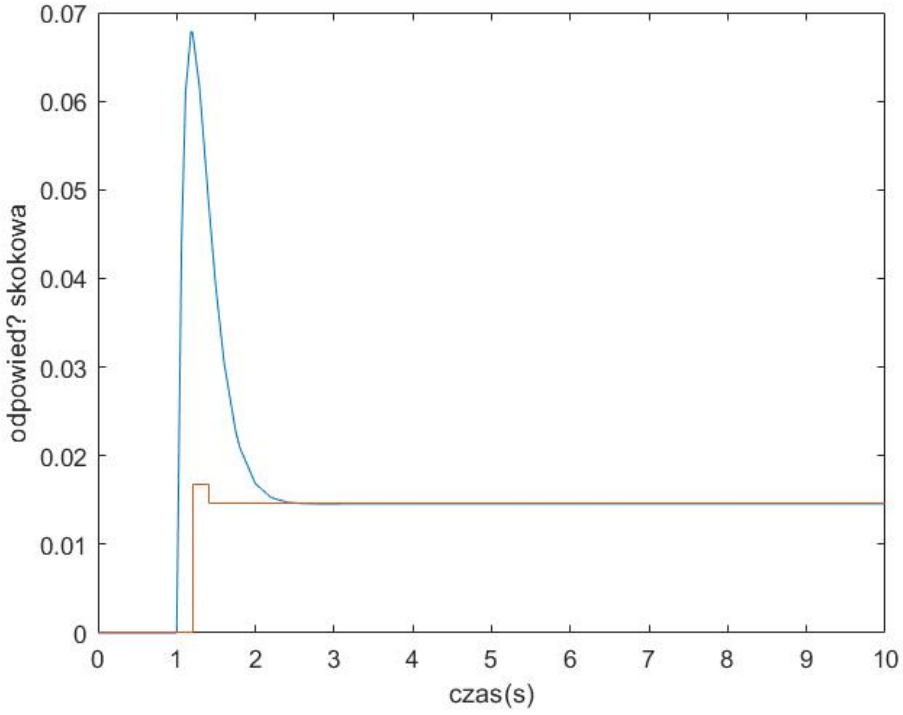

Rysunek 9: Odpowiedzi skokowe transmitancji ciągłej i dyskretnej przy zmianie sygnału wejściowego z 0 na 1 w chwili 1 s dla okresu próbkowania T=0.15s



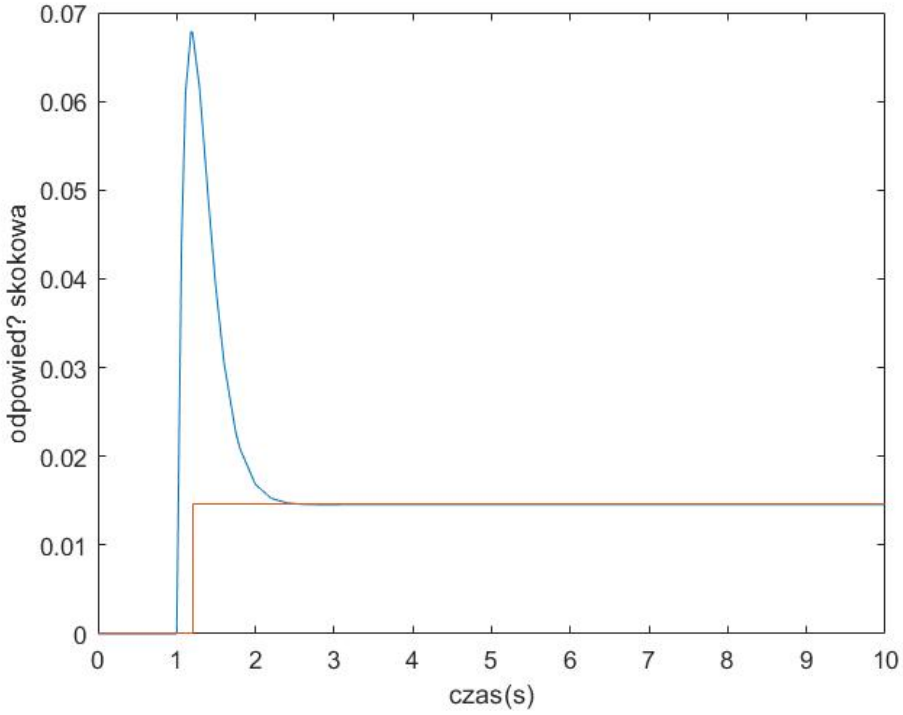

Rysunek 10: Odpowiedzi skokowe transmitancji ciągłej i dyskretnej przy zmianie sygnału wejściowego z 0 na 1 w chwili 1 s dla okresu próbkowania T=0.25s




Rysunek 11: Odpowiedzi skokowe transmitancji ciągłej i dyskretnej przy zmianie sygnału wejściowego z 0 na 1 w chwili 1 s dla okresu próbkowania T=0.5s




Rysunek 12: Odpowiedzi skokowe transmitancji ciągłej i dyskretnej przy zmianie sygnału wejściowego z 0 na 1 w chwili 1 s dla okresu próbkowania T=1.0s




Rysunek 13: Odpowiedzi skokowe transmitancji ciągłej i dyskretnej przy zmianie sygnału wejściowego z 0 na 1 w chwili 1 s dla okresu próbkowania T=5.0s

Początkowo czasy symulacji miały być inne. Zobaczyłem natomiast, iż czasy tj. 5s i 10s są zbyt duże dla tego modelu (co widać na *Rysunku 13*). Do danych testowych dodałem okresy próbkowania tj. T = 0.05, T = 0.15s oraz T = 0.25 s.

# 6 Zadanie 5

Regulator ze sprzężeniem od stanu wyznaczyłem symbolicznie za pomocą programu Matlab za pomocą poniższego skryptu.

```
% wyznaczyc symbolicznie regulator ze sprzezeniem od stanu

K = sym('k', [1 3]);

% s_b - biegun ukladu zamknietego

syms s_b;

L = (s - s_b)^(3);
P = det(s * eye(size(A)) - A + B*K);

% MATLAB: a = ( b == c ) <-- assign true to a when b equals c, false in
% another case, but when we have expression with symbol, it solve us this
% equation

zeroL = ( L == 0 );
zeroP = ( P == 0 );

sol = ( coeffs(L, s, 'All') == coeffs(P, s, 'All') );

out = solve(sol, K);
```

```
% dla 3 przykladowych wartosci bieguna sprawdzic otrzymane wyniki
% numerycznie

s_b1 = -0.1;
s_b2 = -1;
s_b3 = - 10;

K_1 = acker(A, B, [ s_b1, s_b1, s_b1 ]);
K_2 = acker(A, B, [ s_b2, s_b2, s_b2 ]);
K_3 = acker(A, B, [ s_b3, s_b3, s_b3 ]);
```

W skrypcie tym korzystam z funkcji:

- *acker(A, B, [bieguny])* - funkcja, której argumentami wejściowymi są macierze *A, B* oraz wektor biegunów. *Output* tej funkcji to współczynniki regulatora ze sprzężeniem od stanu.

Wykonując skrypt otrzymałem następujące zależności elementów wektora $K$ od potrójnego bieguna $s_b$ układu zamkniętego:

$$\begin{cases} k_1 = -3 * s_b - 15 \\ k_2 = 4 * s_b^2 - 74 \\ k_3 = -s_b^3 - 120 \end{cases}$$

Dla wartości numerycznych przyjąłem następujące bieguny: -0.1, -1, -10. Wszystkie bieguny są ujemne, gdyż wiemy, że bieguny stabilne to te, które spełniają warunek:

$$\Re(s) < 0 \tag{3}$$

Dla tak przyjętych biegunów otrzymałem następujące wartości współczynników $k$ regulatora:

- $s_b$ = -0.1
$$\begin{cases} k_1 = -14.75 \\ k_2 = -73.97 \\ k_3 = -119.999 \end{cases}$$
- $s_b$ = -1
$$\begin{cases} k_1 = -12 \\ k_2 = -71 \\ k_3 = -119 \end{cases}$$
- $s_b$ = -10
$$\begin{cases} k_1 = -15 \\ k_2 = 226 \\ k_3 = 880 \end{cases}$$

# 7 Zadanie 6

Zasymulowałem obiekt ze sprzężeniem od stanu. Przyjąłem jako warunek początkowy obiektu

$$x(0) = [-1, 2, 1]^T \tag{4}$$

i warunek końcowy:

$$x(t_k) = [0, 0, 0]^T \tag{5}$$

Jakość regulacji przyjąłem jako szybkość zbieżnych zmiennych stanu oraz wartości i szybkości zmian sygnału sterującego.

Dla tego zadania stworzyłem skrypt w programie Matlab przedstawiony poniżej:

```
% wyznaczyc symbolicznie regulator ze sprzezeniem od stanu

K = sym('k', [1 3]);

% s_b - biegun ukladu zamknietego

syms s_b;

L = (s - s_b)^(3);
P = det(s * eye(size(A)) - A + B*K);

% MATLAB: a = ( b == c ) <-- assign true to a when b equals c, false in
% another case, but when we have expression with symbol, it solve us this
% equation

zeroL = ( L == 0 );
zeroP = ( P == 0 );

sol = ( coeffs(L, s, 'All') == coeffs(P, s, 'All') );

out = solve(sol, K);

% dla 3 przykladowych wartosci bieguna sprawdzic otrzymane wyniki
% numerycznie

s_b1 = -0.01;
s_b2 = -0.05;
s_b3 = -0.1;
s_b4 = -0.15;
s_b5 = -0.25;
s_b6 = -0.5;
s_b7 = -0.75;
s_b8 = -1;
s_b9 = -2;
s_b10 = -5;
s_b11 = -1.75;
s_b12 = -15;

K_1 = acker(A, B, [ s_b1, s_b1, s_b1 ]);
K_2 = acker(A, B, [ s_b2, s_b2, s_b2 ]);
K_3 = acker(A, B, [ s_b3, s_b3, s_b3 ]);
K_4 = acker(A, B, [ s_b4, s_b4, s_b4 ]);
K_5 = acker(A, B, [ s_b5, s_b5, s_b5 ]);
K_6 = acker(A, B, [ s_b6, s_b6, s_b6 ]);
K_7 = acker(A, B, [ s_b7, s_b7, s_b7 ]);
K_8 = acker(A, B, [ s_b8, s_b8, s_b8 ]);
K_9 = acker(A, B, [ s_b9, s_b9, s_b9 ]);
K_10 = acker(A, B, [ s_b10, s_b10, s_b10 ]);
K_11 = acker(A, B, [ s_b11, s_b11, s_b11 ]);
K_12 = acker(A, B, [ s_b12, s_b12, s_b12 ]);
% wartosc podowana do Matlaba w celu uniwersalnosci pomiarow

% 2(-0.05) 5(-0.25) 6(-0.5) 7(-0.75)  8 (-1) 9(-2) 10(-5) - te wartości wybrałem
K_x = K_11;
```

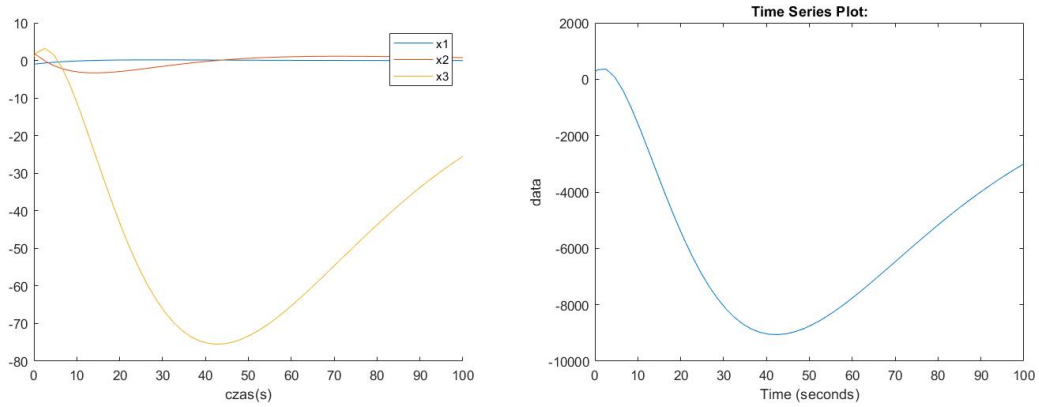
W programie tym korzystałem z funkcji:

- *acker(A, B, [bieguny])* - funkcja, której argumentami wejściowymi są macierze $A$, $B$ oraz

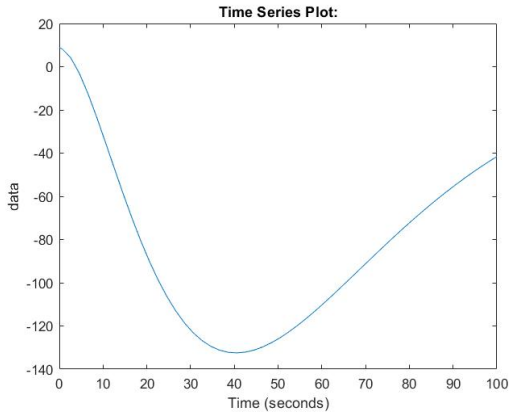
wektor biegunów. *Output* tej funkcji to współczynniki regulatora ze sprzężeniem od stanu.

- *coeffs(L, s, 'all')* - funkcja zwraca wszystkie współczynniki wielomiany stojące przy kolejnych potęgach s (w naszym przypadku). Jako argumenty wejściowe podajemy funkcję, zmienną zależną.

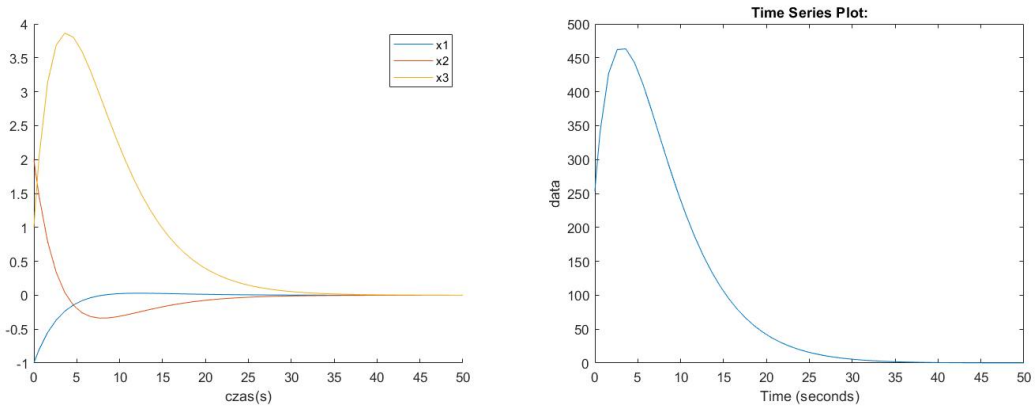
Przyjąłem pewne bieguny regulatora i dla nich obliczałem generowałem wykresy, które są przedstawione poniżej:



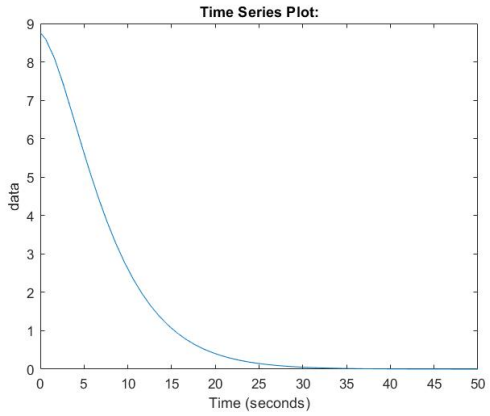

Rysunek 14: Wykres zmiennych stanu (po lewej) oraz sterowania (po prawej) dla bieguna: $s_b$ = -0.05



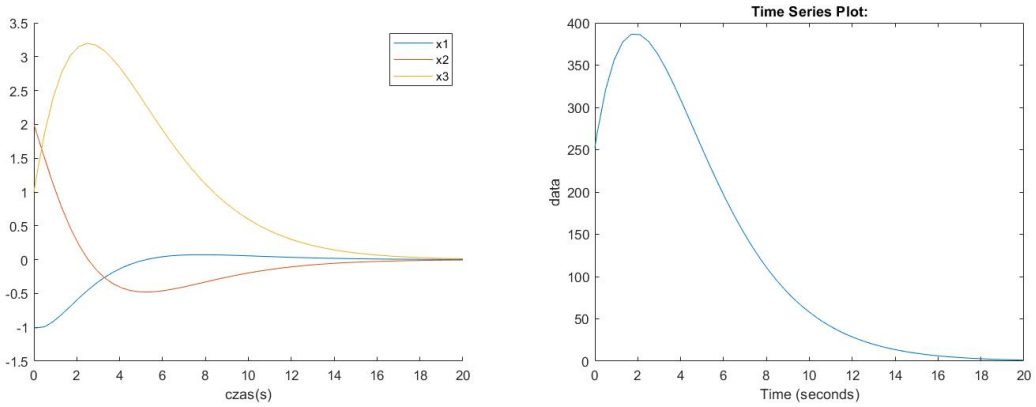

Rysunek 15: Wyjście obiektu dla bieguna: $s_b$ = -0.05



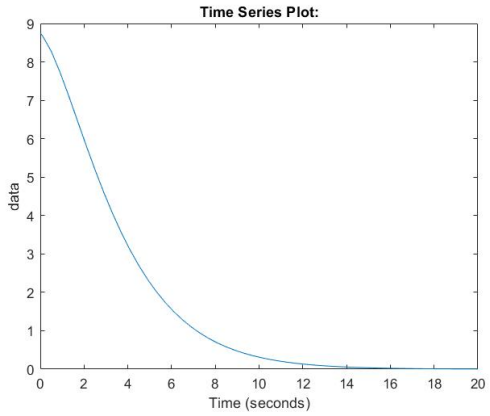

Rysunek 16: Wykres zmiennych stanu (po lewej) oraz sterowania (po prawej) dla bieguna: $s_b$ = -0.25



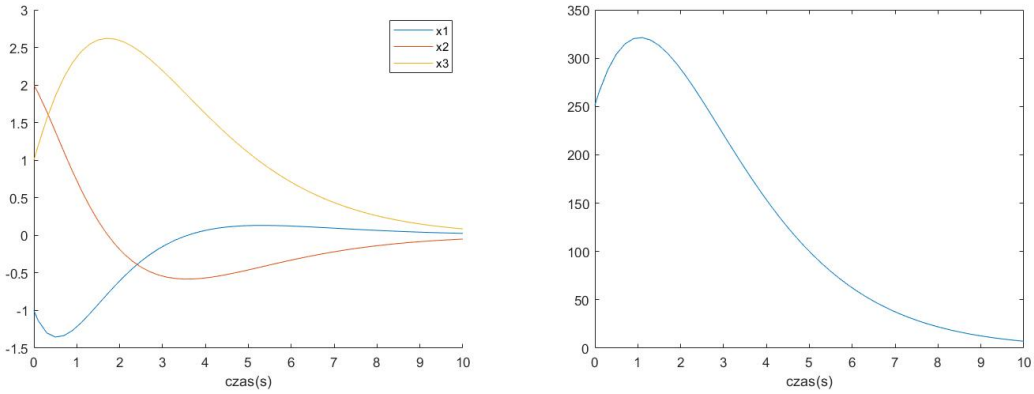

Rysunek 17: Wyjście obiektu dla bieguna: $s_b$ = -0.25



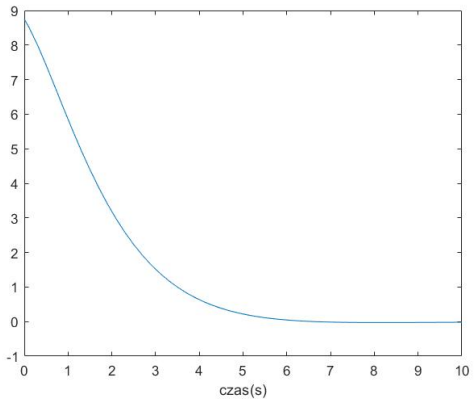

Rysunek 18: Wykres zmiennych stanu (po lewej) oraz sterowania (po prawej) dla bieguna: $s_b$ = -0.5



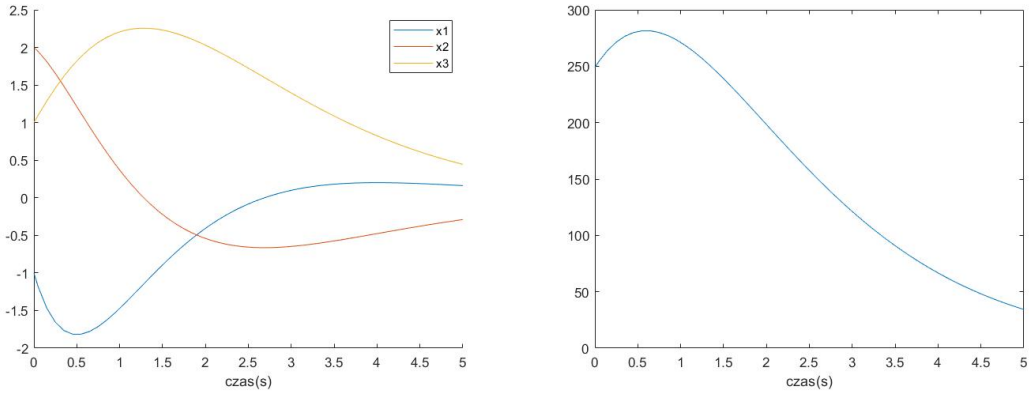

Rysunek 19: Wyjście obiektu dla bieguna: $s_b$ = -0.5



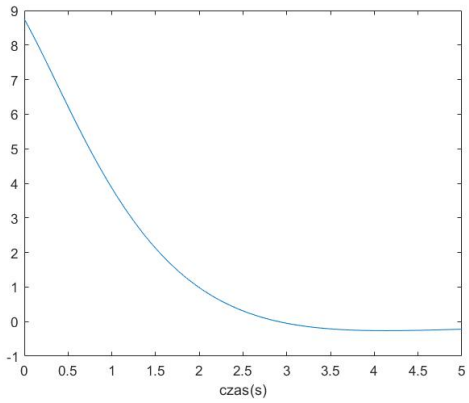

Rysunek 20: Wykres zmiennych (po lewej) oraz sterowania (po prawej) dla bieguna: $s_b$ = -0.75



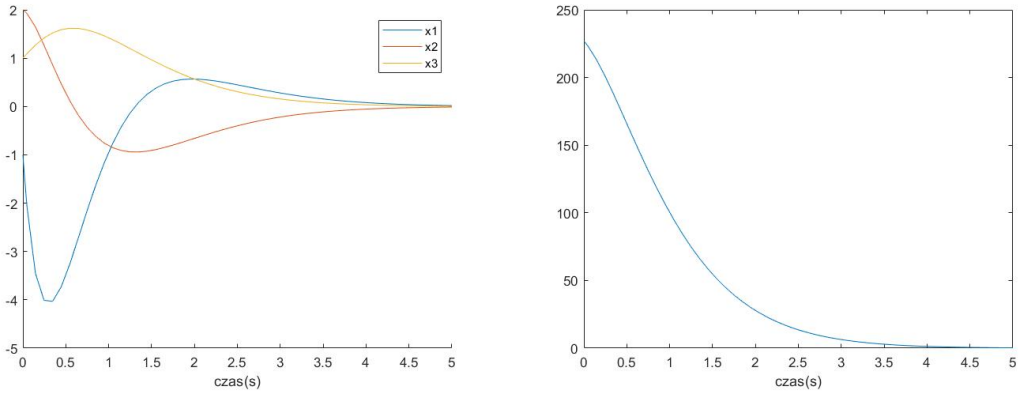

Rysunek 21: Wyjście obiektu dla bieguna: $s_b$ = -0.75



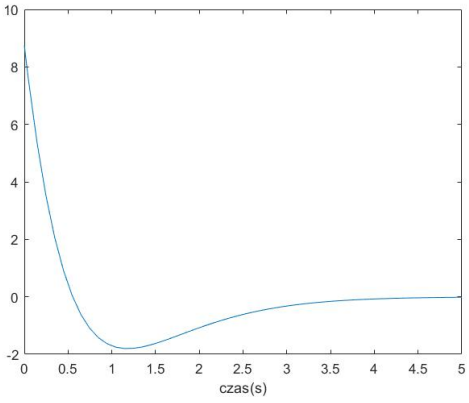

Rysunek 22: Wykres zmiennych (po lewej) oraz sterowania (po prawej) dla bieguna: $s_b$ = -1.0



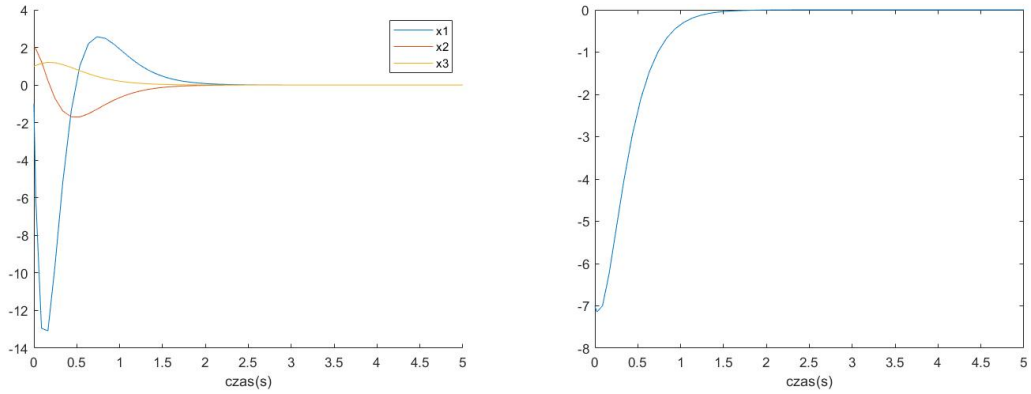

Rysunek 23: Wyjście obiektu dla bieguna: $s_b$ = -1.0



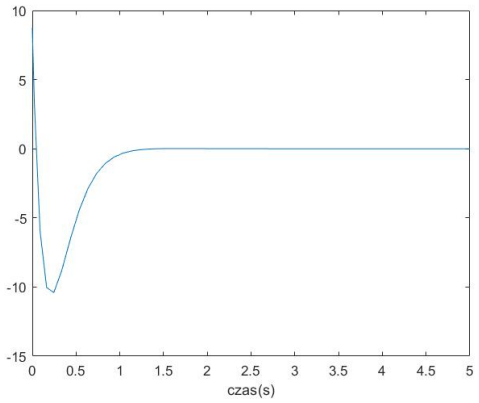

Rysunek 24: Wykres zmiennych (po lewej) oraz sterowania (po prawej) dla bieguna: $s_b$ = -2.0




Rysunek 25: Wyjście obiektu dla bieguna: $s_b$ = -2.




Rysunek 26: Wykres zmiennych (po lewej) oraz sterowania (po prawej) dla bieguna: $s_b$ = -5.0




Rysunek 27: Wyjście obiektu dla bieguna: $s_b$ = -5.0

Dla biegunów bliskich zera (lecz mniejszych od zera) obiekt osiagą warunek końcowy bardzo wolno. Dla coraz mniejszych biegunów (większych patrząc na wartość bezwzględną bieguna) szybkość zbieżności zmiennych stanu jest coraz szybszy, lecz w pewnym momencie dzieje się to kosztem szybkości i wartości zmian sygnału sterującego. Z przedstawionych powyżej wykresów wg mnie najlepszym jest wykres dla bieguna $s_b$ = -2.0. Przesterowanie nie jest tak dużo oraz sterowanie nie zmienia się gwałtowanie i nie osiąga dużych wartości. Próbowałem jeszcze wartości z przedziału (-2, -1) natomiast nie były one lepsze od bieguna $s_b$ = -2.0.

# 8 Zadanie 7

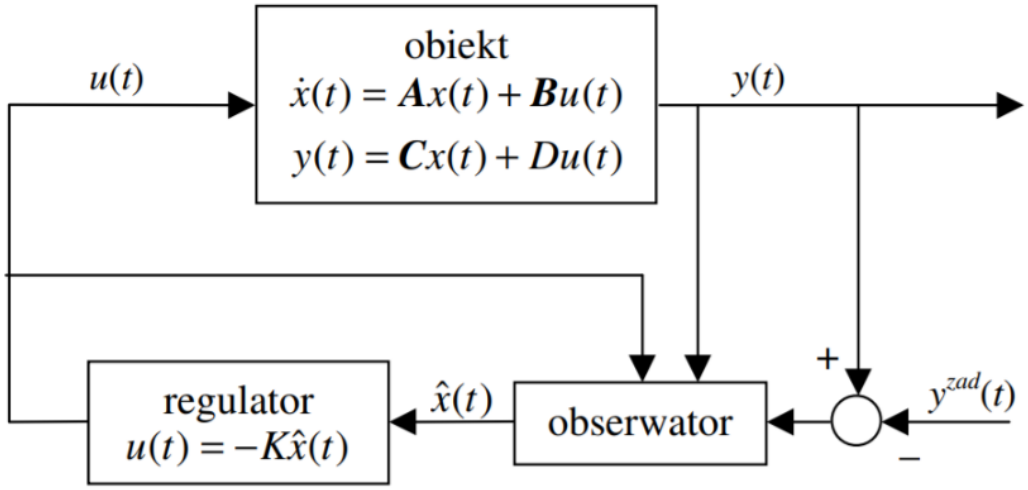
Równania obserwatora pełnego rzędu wyznaczyłem za pomocą programu Matlab. W napisanym skrypcie korzystałem z następujących funkcji:

- *acker(A, C, [bieguny]*
- *coeffs(L, s, 'all')*

Funckje zostały opisane w punkcie 7 (Zadanie 6).
Dane wartości współczynników L są następujące:

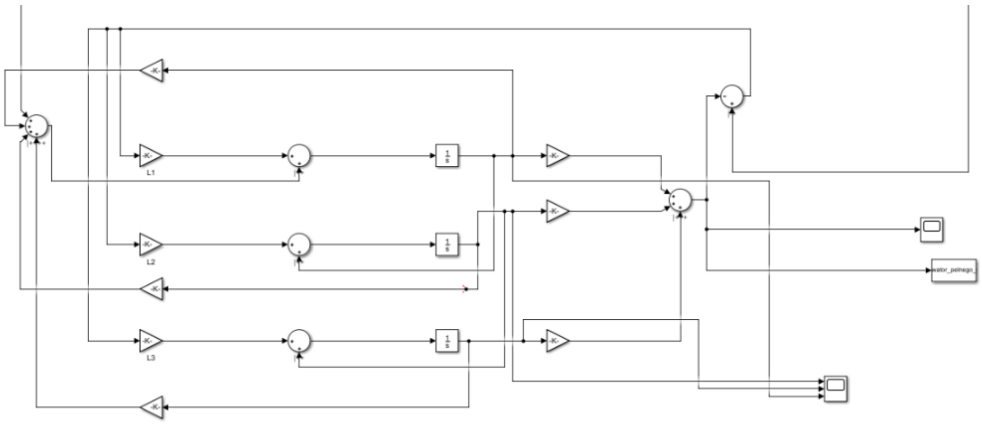
$$\begin{cases} l_1 = -\frac{22628*{s_o}^3}{10395} - \frac{79228*{s_o}^2}{3465} - \frac{287708*s_o}{3465} - \frac{1168108}{10395} \\ l_2 = \frac{6448*{s_o}^3}{10395} + \frac{22628*{s_o}^2}{3465} + \frac{79228*s_o}{3465} + \frac{287708}{10395} \\ l_3 = -\frac{1808*{s_o}^3}{10395} - \frac{6448*{s_o}^2}{3465} - \frac{22628*s_o}{3465} - \frac{79228}{10395} \end{cases} \tag{6}$$

Rysunek ogólnej struktury układu regulacji z obserwatorem jest zamieszony poniżej:

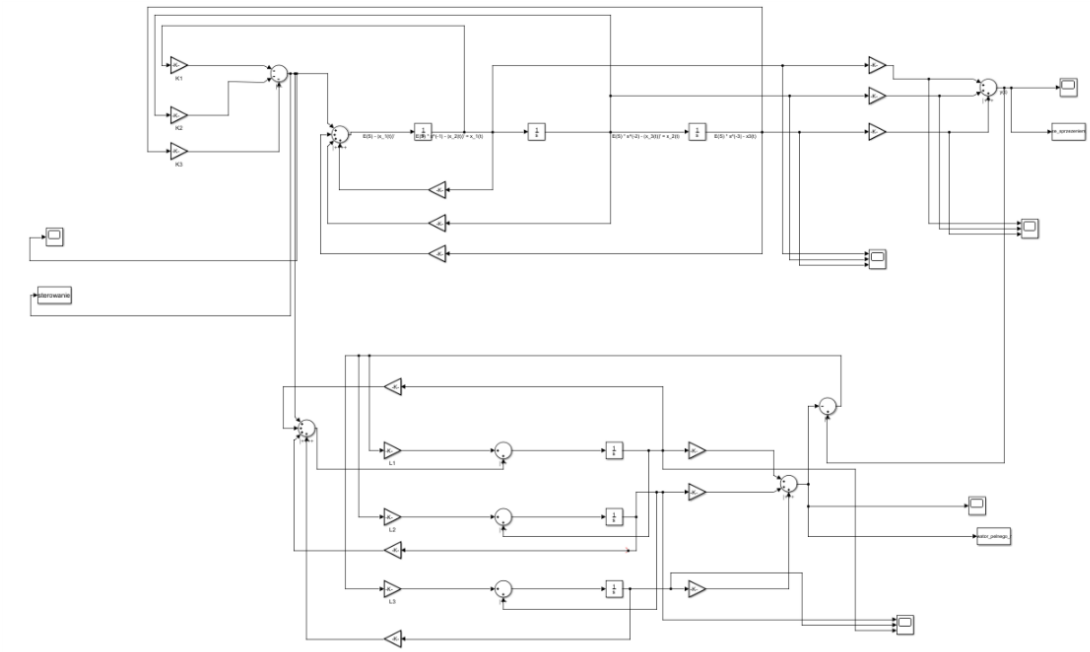



Rysunek 28: Ogólna struktura układu regulacji z obserwatorem

Rysunki struktury obserwatora i układu regulacji z obserwatorem są zamieszczone poniżej:



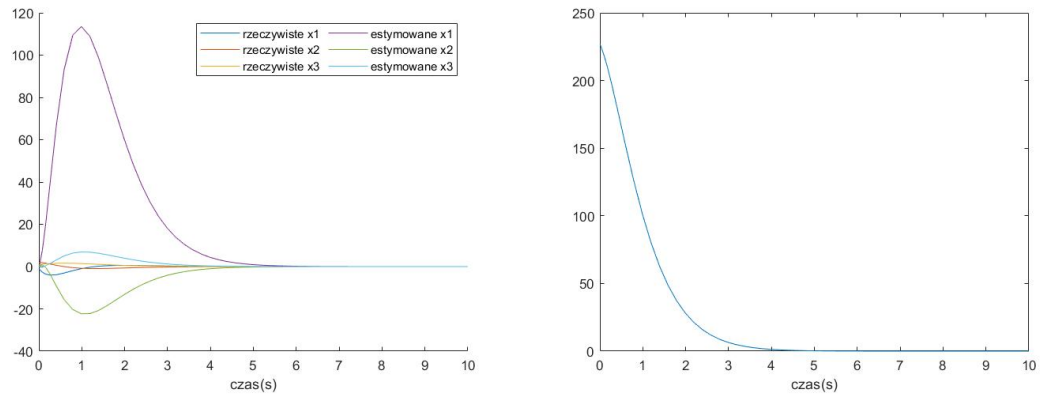
Rysunek 29: Struktura obserwatora pełnego rzędu o potrójnym biegunie



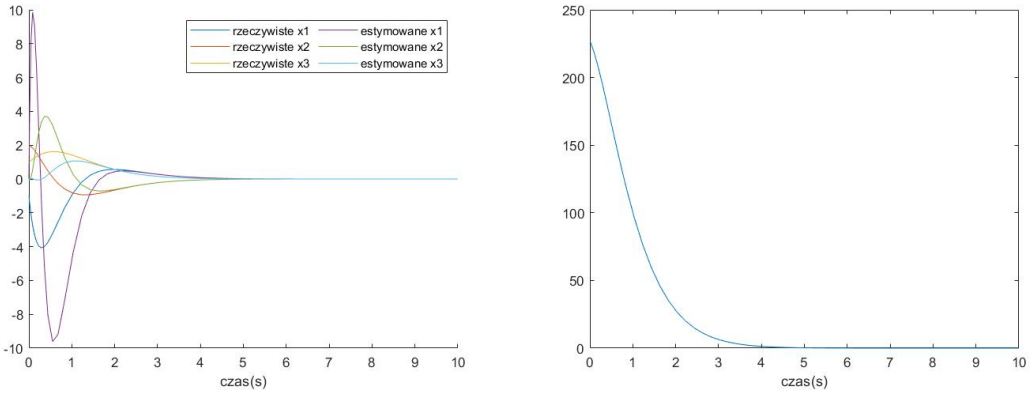
Rysunek 30: Struktura układu regulacji z obserwatora pełnego rzędu o potrójnym biegunie

# 9 Zadanie 8

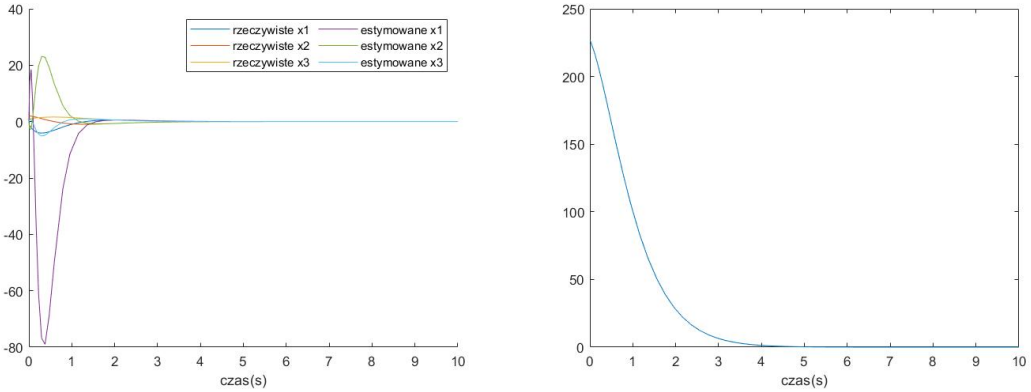
W tej części testuję działanie obserwatora przy regulatorze korzystającym z mierzonego stanu. Do symulacji przyjąłem warunek początkowy obserwatora zerowy oraz niezerowy warunek począkowy dla obiektu (równy [-1, 2, 1]).



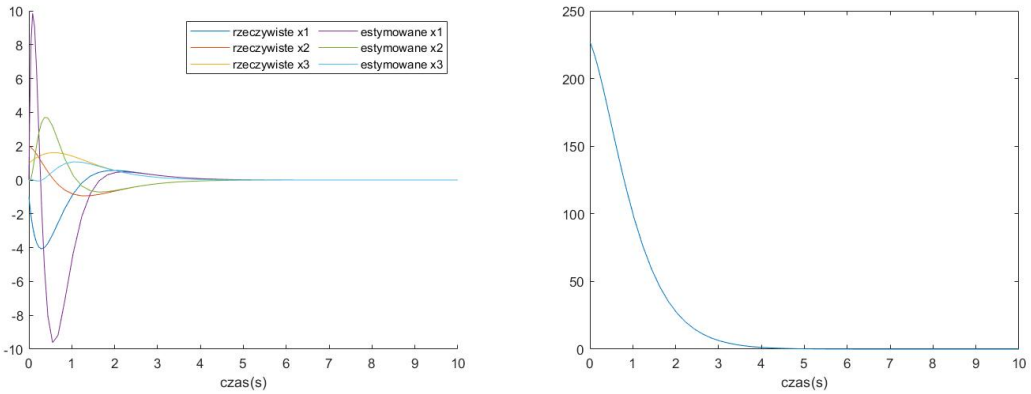

Rysunek 31: Wykres zmiennych stanu (po lewej) oraz sterowania (po prawej) dla bieguna: $s_o$ = -2.0



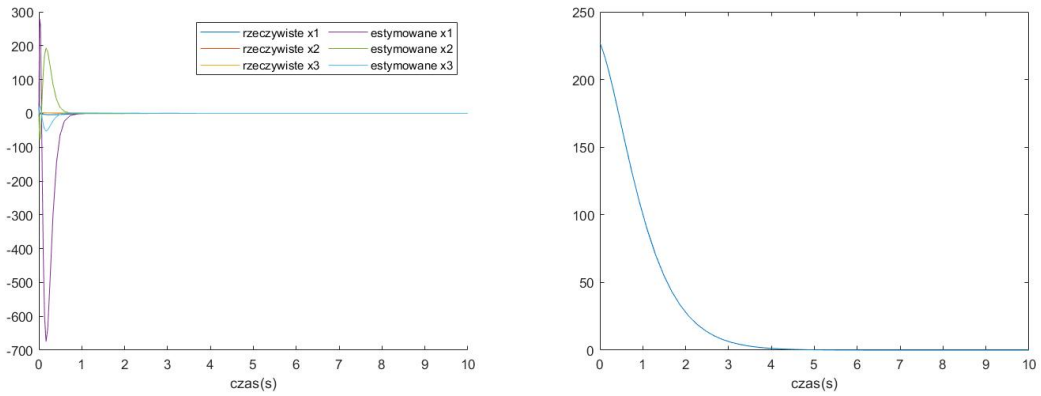

Rysunek 32: Wykres zmiennych stanu (po lewej) oraz sterowania (po prawej) dla bieguna: $s_o$ = -5.0




Rysunek 33: Wykres zmiennych stanu (po lewej) oraz sterowania (po prawej) dla bieguna: $s_o$ = -7.0




Rysunek 34: Wykres zmiennych stanu (po lewej) oraz sterowania (po prawej) dla bieguna: $s_o$ = -10.0




Rysunek 35: Wykres zmiennych stanu (po lewej) oraz sterowania (po prawej) dla bieguna: $s_o$ = -15.0

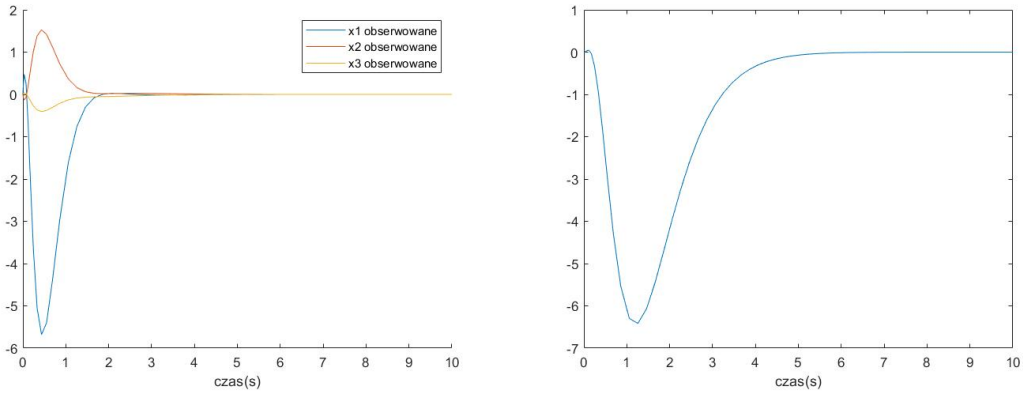
Parametry obserwatora obliczałem numerycznie przy pomocy skryptu Matlaba:

```
% bieguny obserwatora

s_o2 = -2;
s_o3 = -5;
s_o4 = -7;
s_o5 = -10;
s_o6 = -15;


L_2 = acker(A', C', [ s_o2, s_o2, s_o2 ]);
L_3 = acker(A', C', [ s_o3, s_o3, s_o3 ]);
L_4 = acker(A', C', [ s_o4, s_o4, s_o4 ]);
L_5 = acker(A', C', [ s_o5, s_o5, s_o5 ]);
L_6 = acker(A', C', [ s_o6, s_o6, s_o6 ]);

% przy symulacji mozemy zmienic aktualnie wyswietlany wykres
L_x = L_6;
```

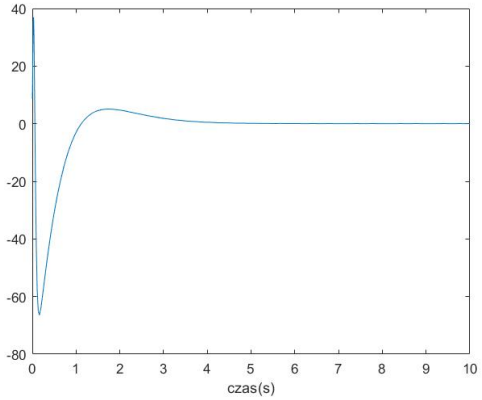
W następnym zadaniu będzie w stanie zaobserwować czy regulator korzystający ze stanu obserwowanego będzie działał prawidłowo.

# 10 Zadanie 9

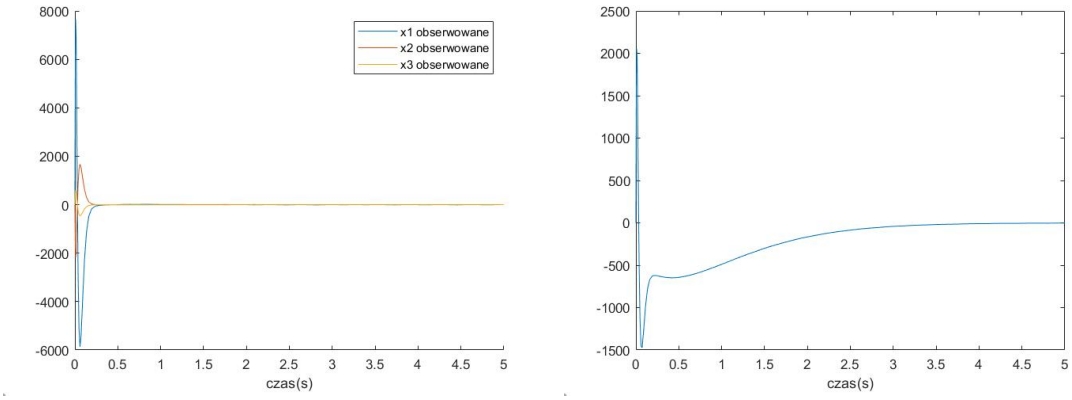
Gdy pomiar zmiennych stanu nie jest możliwy, wówczas wykorzystujemy stan obserwowany. Poniżej znajdują się wykresy dla dwóch obserwaotorów "wolnego" i "szybkiego":



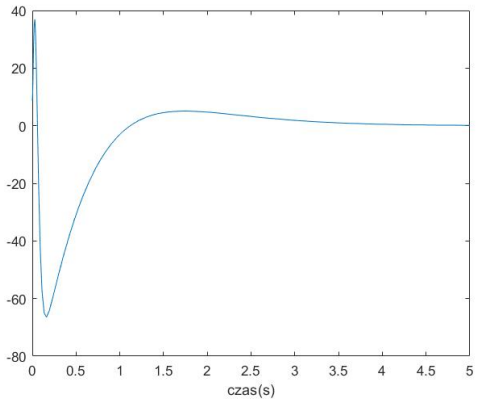

Rysunek 36: Wykres zmiennych stanu (po lewej) oraz sterowania (po prawej) dla bieguna "wolnego": $s_o$ = -5.0




Rysunek 37: Wyjście obiektu dla bieguna "wolnego": $s_o$ = -5.0




Rysunek 38: Wykres zmiennych stanu (po lewej) oraz sterowania (po prawej) dla bieguna "szybkiego": $s_o$ = -50.0




Rysunek 39: Wyjście obiektu dla bieguna "szybkiego": $s_o$ = -50.0

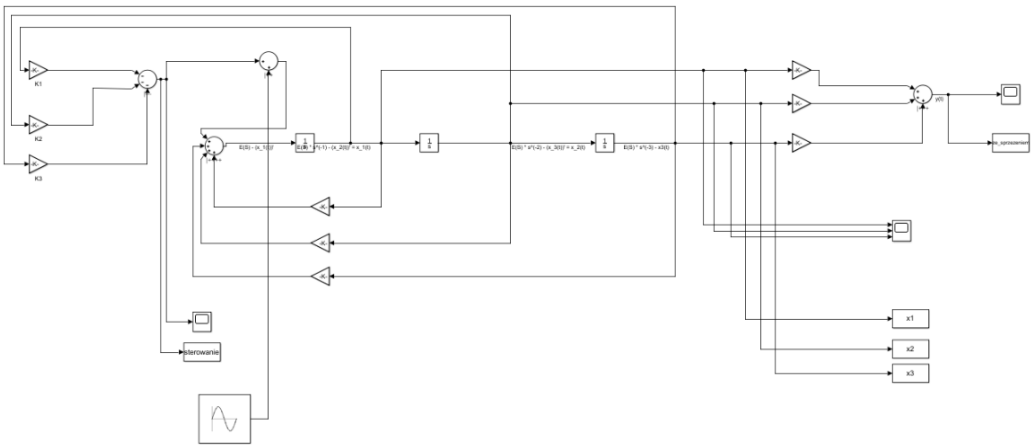
Na podstawie rysunków widzimy, iż praktyka zgadza się z teorią. Bieguny "szybkie" zdecydowanie szybciej osiągnęły wartość zadaną. Niestety patrząc na wartość sterowania - jest ona bardzo duża. Przy projektowaniu regulatorów musimy znaleźć "złoty środek" pomiędzy szybkością a wartością sterowania.

# 11 Zadanie dodatkowe

Dodałem funkcje sinus wcelu sprawdzenia nadążania za zmianami wartości zadanej sygnału wyjściowego. Prawo regulacji postaci:

$$u(t) = -K * x(t) + (N_u + K * N_x) * y_{zad} \tag{7}$$

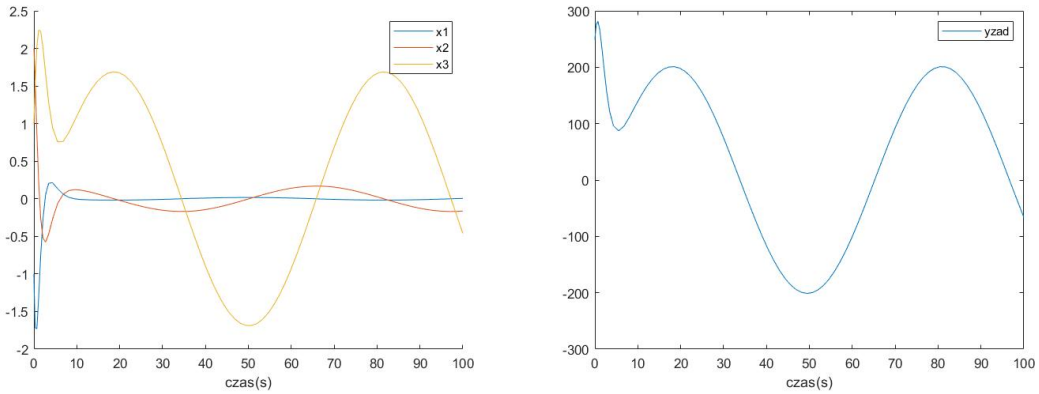
Schemat jest przedstawiony poniżej:



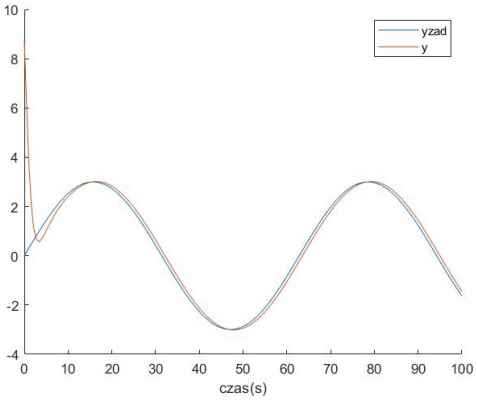

Stworzyłem następujący skrypy w programie Matlab:

```
% zadanie nadążania za zmianami wartości zadanej sygnału wyjściowego.

% u(t) = - K * x(t) + (N_u + K * N_x) *y_zad

X = inv([A, B; C, D]) * [0 0 0 1]';

Nu = X(4, 1);
Nx = X(1:3, 1);
```

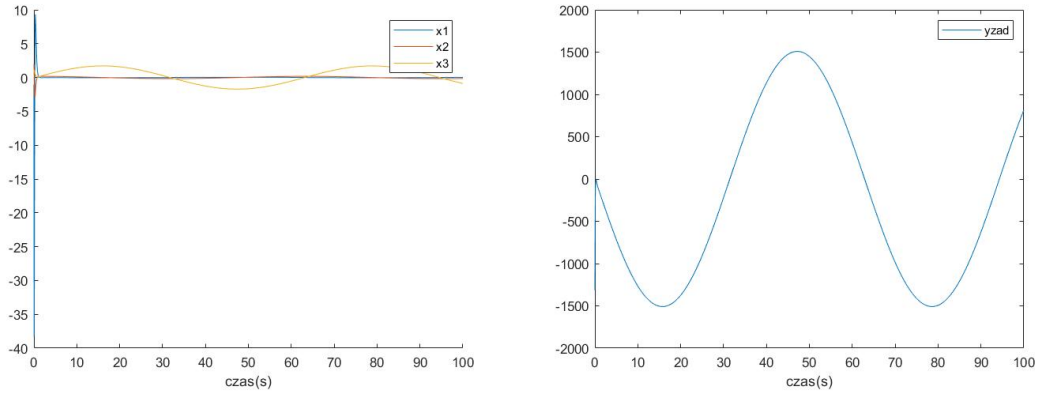
```
wyr = K_x * Nx + Nu;
```



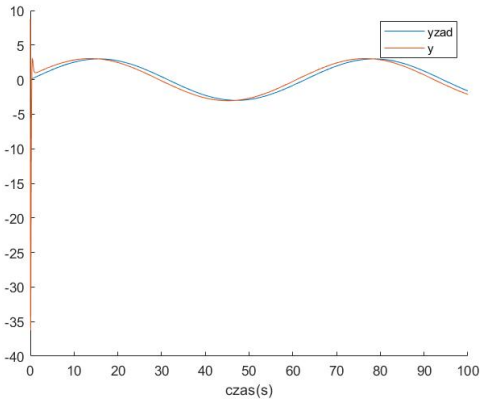

Rysunek 40: Wykres zmiennych stanu (po lewej) oraz sterowania (po prawej) dla bieguna "wolnego": $s_o$ = -2.0




Rysunek 41: Wyjście obiektu dla bieguna "wolnego": $s_o$ = -2.0




Rysunek 42: Wykres zmiennych stanu (po lewej) oraz sterowania (po prawej) dla bieguna "szybkiego": $s_o$ = -10.0




Rysunek 43: Wyjście obiektu dla bieguna "szybkiego": $s_o$ = -10.0

Ponownie praktyka potwierdza teorię - sygnał sterujący jest zdecydowanie większy (jego amplituda) dla biegunów "szybkich".