<table>
<tr><td colspan="2">Politechnika Poznańska
Wydział Automatyki, Robotyki i Elektrotechniki</td><td></td></tr>
<tr><td>AiR Sem. 6</td><td>Identyfikacja Systemów</td><td>2022/23 (s.letni)</td></tr>
<tr><td>Skład osobowy:
Jakub Wicher 147589</td><td>Identyfikacja systemu HILSys</td><td>Data przesłania.:
29.05.23</td></tr>
<tr><td>Grupa A6 (L9)</td><td>Sprawozdanie z projektu</td><td></td></tr>
</table>

1 Określenie celu modelowania eksperymentalnego.

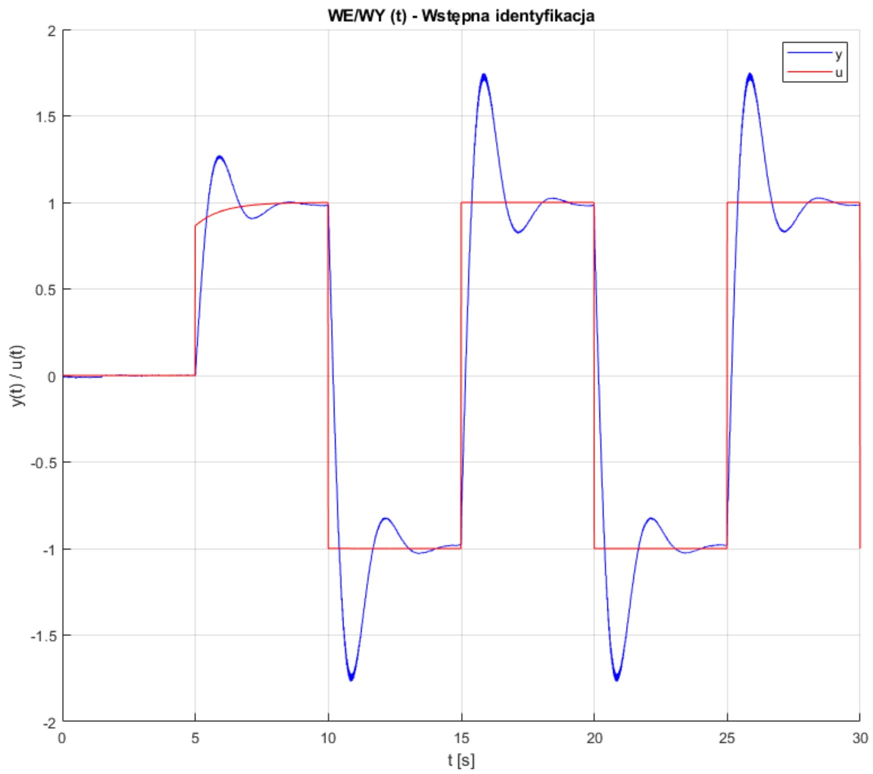
Celem modelowania eksperymentalnego jest uzyskanie symulatora wyjaśniającego odpowiedź systemu z dokładnością na poziomie $J_{FIT} > 95\%$.

2 Zebranie wiedzy dostępnej o systemie a priori (tj. przed eksperymentem identyfikacyjnym).

- źródłem danych jest pojedyncza sekcja systemu HILSys (proces czasu ciągłego),
- system posiada cechy takie jak:
  - po odjęciu składowej stałej od odpowiedzi y dynamika procesu jest (prawie) liniowa,
  - proces ma stałe parametry,
- dane numeryczne znajdują się w pliku HILSys.mat,
- dane zebrane z okresem próbkowania $T_p = 0.01$ s,
- pierwsze 150 próbek pomiarowych sygnału y bez odjętej składowej stałej,
- sygnał pobudzający u jest znany dokładnie (brak zakłóceń pomiarowych w sygnale u).

3 Zebranie wiedzy dodatkowej (na podstawie danych numerycznych).

Za pomocą środowiska Matlab zostały wczytane dane z pliku HILSys.mat, a następnie, na jednym wykresie, wykreślone zostały przebiegi sygnałów $u(t)$ oraz $y(t)$.



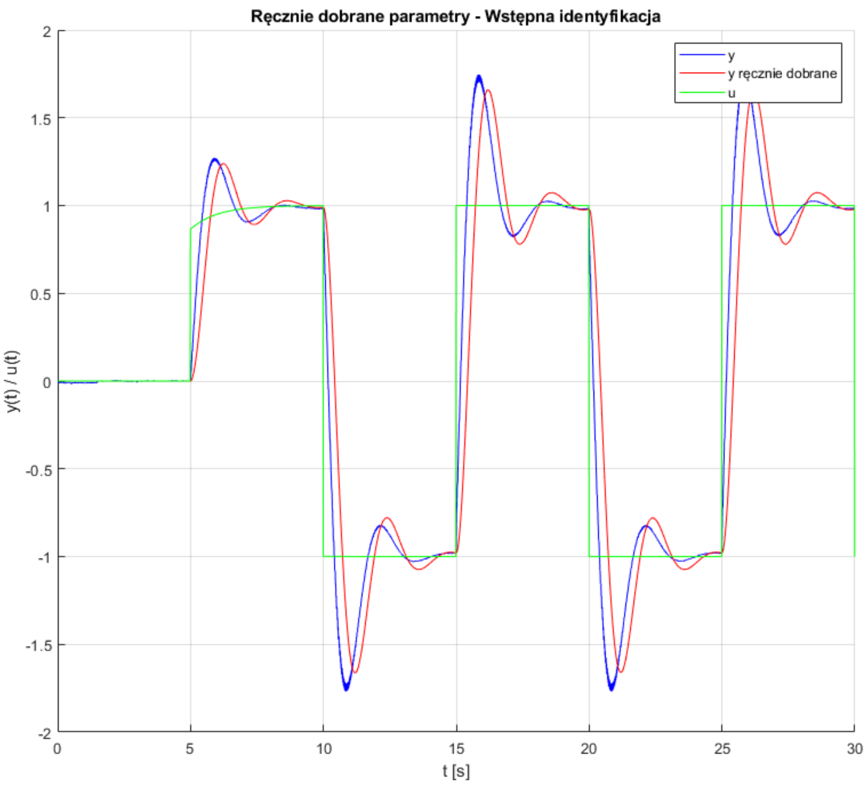
Rysunek 1: Przebieg sygnałów wejściowego (u) / wyjściowego (y) na podstawie danych z pliku HILSys.mat

Na podstawie powyższego przebiegu, można stwierdzić, że badany obiekt na odpowiedź zbliżoną do odpowiedzi obiektu oscylacyjnego. Aby potwierdzić przypuszczenia stworzono obiekt oscylacyjny ciągły z parametrami ω_n oraz ξ dobranymi metodą prób i błędów. Następnie obiekt zdyskretyzowano, za pomocą metody zero-order hold (ZOH). Za pomocą funkcji *lsim* przeprowadzono symulację odpowiedzi obiektu na ten sam sygnał wejściowy (u) i wykreślono przebiegi porównujące stworzony obiekt (y ręcznie dobrane) z rzeczywistym (y).

```
k = 1;
wn = 2.78;
ksi = 0.33;
Gs = k*wn^2/(s^2 + 2*ksi*wn*s + wn^2);
Gz = c2d(Gs, Tp);
y_RDP = lsim(Gz, u, t);

figure;
hold on;
title('Recznie dobrane parametry - Wstepna identyfikacja');
plot(t, y, 'b', t, y_RDP, 'r', t, u, '-g');
legend('y', 'y recznie dobrane', 'u');
xlabel('t [s]');
ylabel('y(t) / u(t)');
grid on;
```

Listing 1: Kod realizujący symulację oraz wyświetlenie odpowiedzi obiektu



Rysunek 2: Przebiegi wygenerowane przez skrypt.

Można zauważyć, że przebiegi są do siebie mocno zbliżone, więc przypuszczenie było prawidłowe.

4 Wybór wariantu i struktury modelu.

Do realizacji zadania sokrzystano z wariantu 3. z instrukcji - identyfikacja typu least squares (LS), model dynamiczny czasu dyskretnego, identyfikacja typu BLACK-BOX. Przyjęta struktura modelu to model dyskretny typu ARMA, który przedstawia się następująco:

$$G(z) = \frac{Y(z)}{U(z)} = \frac{(b_0 + b_1 \cdot z^{-1}) \cdot z^{-1}}{1 + a_1 \cdot z^{-1} + a_2 \cdot z^{-2}} \tag{1}$$

z równania 1 wyznaczono wzór na $y(n \cdot T_p)$:

$$y(n \cdot T_p) = -a_1 \cdot y((n-1) \cdot T_p) - a2((n-2) \cdot T_p) + b_0 \cdot u((n-1) \cdot T_p) + b1 \cdot u((n-2) \cdot T_p) \tag{2}$$

gdzie:

- T_p - to okres próbkowania sygnału,
- n - to numer próbki,
- a_1, a_2, b_0, b_1 - to szukane parametry,
- $u(n \cdot T_p)$ - to sygnał wejściowy w chwili $(n \cdot T_p)$,
- $y(n \cdot T_p)$ - to sygnał wyjściowy w chwili $(n \cdot T_p)$.

Niech Θ będzie szukanym wektorem parametrów a_1 ... b_1 (bez a_0, ponieważ jest znane, równe 1)

$$\Theta = [a_1, a_2, b_0, b_1]^T \tag{3}$$

wtedy, na podstawie algorytmu metody LS, wektor Θ można wyznaczyć korzystając z następującej zależności:

$$\Theta = \Phi^{\dagger} \cdot y_N \tag{4}$$

gdzie:

- y_N - to pionowy wektor pomiarów (wyjścia),
- Φ - to macierz składająca się z próbek wejścia wyjścia w odpowiednich chwilach czasowych, wynikająca ze wzoru 2.

Ponieważ najwyższe opóźnienie w układzie wynosi 2. próbki, wektor y_N powinien zacząć się od próbki 3..

5 Realizacja algorytmu LS.

Wczytanie danych i rozdzielenie ich na dane do identyfikacji i walidacji:

```
Tp = 0.01; % s
t = (0.006:0.01:29.995+0.01)'; % sam wygenerowalem bo potrzebuje stale Tp.

procent_valid = 0.3;
length_valid = floor(0.3 * length(t));

y_ident = y(1 : length(t) - length_valid);
u_ident = u(1 : length(t) - length_valid);
t_ident = t(1 : length(t) - length_valid);

y_valid = y(length(t) - length_valid + 1 : end);
u_valid = u(length(t) - length_valid + 1 : end);
t_valid = t(length(t) - length_valid + 1 : end);
```

Listing 2: Przygotowanie danych

5.1 Sposób dla symulacji statycznej.

```
max_delay = 2;

phi = [
        -y_ident(2:end-1), ...
        -y_ident(1:end-2), ...
        +u_ident(2:end-1), ...
        +u_ident(1:end-2)
    ];

theta = pinv(phi) * y_ident(max_delay + 1 : end);

% a0 = 1;
a1 = theta(1);
a2 = theta(2);
b0 = theta(3);
b1 = theta(4);

y_est = [
    y_valid(1);
    y_valid(2);
    ( ...
        -a1 * y_valid(2 : end - 1) ...
        -a2 * y_valid(1 : end - 2) ...
        +b0 * u_valid(2 : end - 1) ...
        +b1 * u_valid(1 : end - 2) ...
    )
];
```

Listing 3: Implementacja metody LS - statycznie

5.2 Metoda iteracyjna (do zastosowania przy symulacji 'real time').

```
max_delay = 2;
N = length(t_ident) - max_delay;
param_vec_len = 4;
phi = zeros(N, param_vec_len);

for i = 1:N
   phi(i, :) = [
       -y_ident(max_delay + i - 1), ...
       -y_ident(max_delay + i - 2), ...
       +u_ident(max_delay + 1 - 1), ...
```

```
        +u_ident(max_delay + i - 2);
    ];
end

theta = pinv(phi) * y_ident(max_delay + 1 : end);

y_est = zeros(size(y));
for i = 1:N
    % y(t) = -a1*y(t-1) -a2*y(t-2) +b0*u(t-1) +b1*u(t-2)
    y_est(max_delay + i) = ...
        -a1 * y_valid(max_delay + i - 1) ...
        -a2 * y_valid(max_delay + i - 2) ...
        +b0 * u_valid(max_delay + i - 1) ...
        +b1 * u_valid(max_delay + i - 2);
end
```

Listing 4: Implementacja metody LS - iteracyjnie

6 Realizacja algorytmu LS.

Wczytanie danych i rozdzielenie ich na dane do identyfikacji i walidacji:

```
Tp = 0.01; % s
t = (0.006:0.01:29.995+0.01)'; % sam wygenerowalem bo potrzebuje stale Tp.

procent_valid = 0.3;
length_valid = floor(0.3 * length(t));

y_ident = y(1 : length(t) - length_valid);
u_ident = u(1 : length(t) - length_valid);
t_ident = t(1 : length(t) - length_valid);

y_valid = y(length(t) - length_valid + 1 : end);
u_valid = u(length(t) - length_valid + 1 : end);
t_valid = t(length(t) - length_valid + 1 : end);
```

Listing 5: Przygotowanie danych

6.1 Sposób dla symulacji statycznej.

```
max_delay = 2;

phi = [
        -y_ident(2:end-1), ...
        -y_ident(1:end-2), ...
        +u_ident(2:end-1), ...
        +u_ident(1:end-2)
    ];

theta = pinv(phi) * y_ident(max_delay + 1 : end);

% a0 = 1;
a1 = theta(1);
a2 = theta(2);
b0 = theta(3);
b1 = theta(4);

y_est = [
    y_valid(1);
    y_valid(2);
    ( ...
        -a1 * y_valid(2 : end - 1) ...
        -a2 * y_valid(1 : end - 2) ...
        +b0 * u_valid(2 : end - 1) ...
        +b1 * u_valid(1 : end - 2) ...
    )
];
```

Listing 6: Implementacja metody LS - statycznie

6.2 Metoda iteracyjna (do zastosowania przy symulacji 'real time').

```
max_delay = 2;
N = length(t_ident) - max_delay;
param_vec_len = 4;
phi = zeros(N, param_vec_len);

for i = 1:N
   phi(i, :) = [
       -y_ident(max_delay + i - 1), ...
       -y_ident(max_delay + i - 2), ...
       +u_ident(max_delay + 1 - 1), ...
       +u_ident(max_delay + i - 2);
   ];
end

theta = pinv(phi) * y_ident(max_delay + 1 : end);

y_est = zeros(size(y));
for i = 1:N
    % y(t) = -a1*y(t-1) -a2*y(t-2) +b0*u(t-1) +b1*u(t-2)
    y_est(max_delay + i) = ...
       -a1 * y_valid(max_delay + i - 1) ...
       -a2 * y_valid(max_delay + i - 2) ...
       +b0 * u_valid(max_delay + i - 1) ...
       +b1 * u_valid(max_delay + i - 2);
end
```

Listing 7: Implementacja metody LS - iteracyjnie

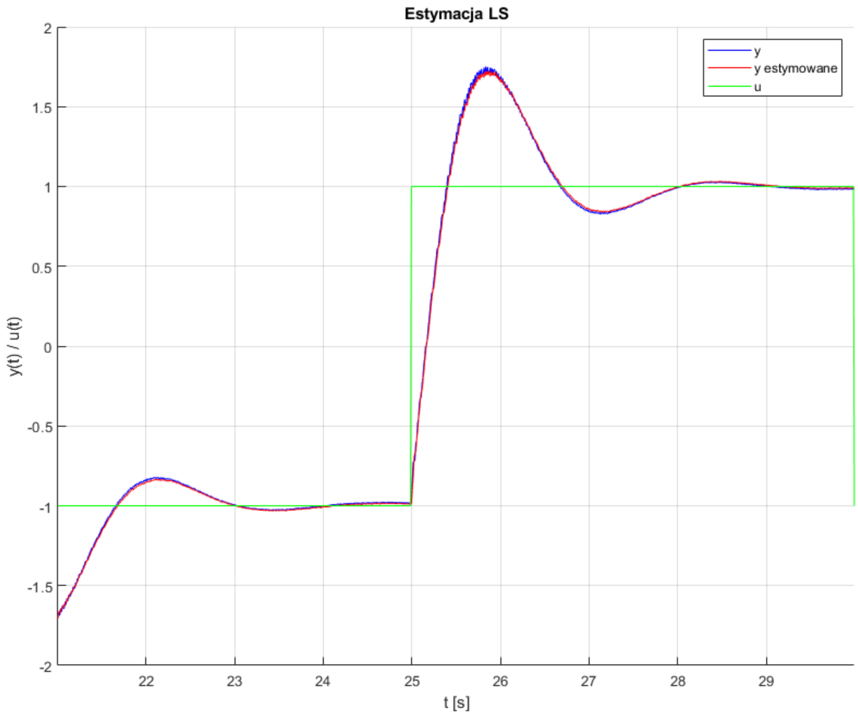
7 Wyświetlenie i analiza wyników.

Z użyciem otrzymanych parametrów obliczono odpowiedź obiektu na wymuszenie i porównano z sygnałem rzeczywistym. Następnie obliczono wskaźniki jakości i potwierdzono dokładność estymacji. Następnie pobudzono obiekt innym sygnałem i sprawdzono jego zachwoanie. Na koniec zbadano jego stabilność i przedstawiono końcowy wynik identyfikacji.

7.1 Wyświetlenie przebiegów.

```
figure;
hold on;
title('Estymacja LS');
plot(t, y, 'b', t_valid, y_est, 'r', t, u, '-g');
xlim([t_valid(1), t_valid(end)])
legend('y', 'y estymowane', 'u');
xlabel('t [s]');
ylabel('y(t) / u(t)');
grid on;
```

Listing 8: Wyświetlenie wyników



Rysunek 3: Przebieg sygnałów wejściowego i wyjściowego rzeczywistego i estymowanego

7.2 Sprawdzenie precyzji obliczonych estymat.

Obliczono następujące wskaźniki jakości:

- $J_{FIT} = (1 - \frac{||y-\hat{y}||}{||y-\bar{y}||}) \cdot 100\%$
- $MSE = \frac{1}{N}\Sigma(y - \hat{y})^2$
- $MAE = \frac{1}{N}\Sigma|y - \hat{y}|$

gdzie:

- y - wartość zmierzona
- $\hat{y}$ - wartość estymowana
- $\bar{y}$ - średnia wartość zmierzona

```
%% Wskazniki
n = length(y_valid);
avg_error = sum(abs(y_valid - y_est))/n;
mean_squared_error = sum((y_valid - y_est).^2)/n;
J_fit = (1 - norm(y_valid - y_est)/norm(y_valid - mean(y_valid)*ones(size(y_valid)))) * 100;
```

Listing 9: Wyświetlenie wyników

i otrzymano następujące wartości:

J_{FIT}	98.42
MSE	2.80
MAE	0.01

Tabela 1: Wartości wskaźników jakości

Spełnione zostało założenie projektu - wskaźnik $J_{FIT} > 95\%$.

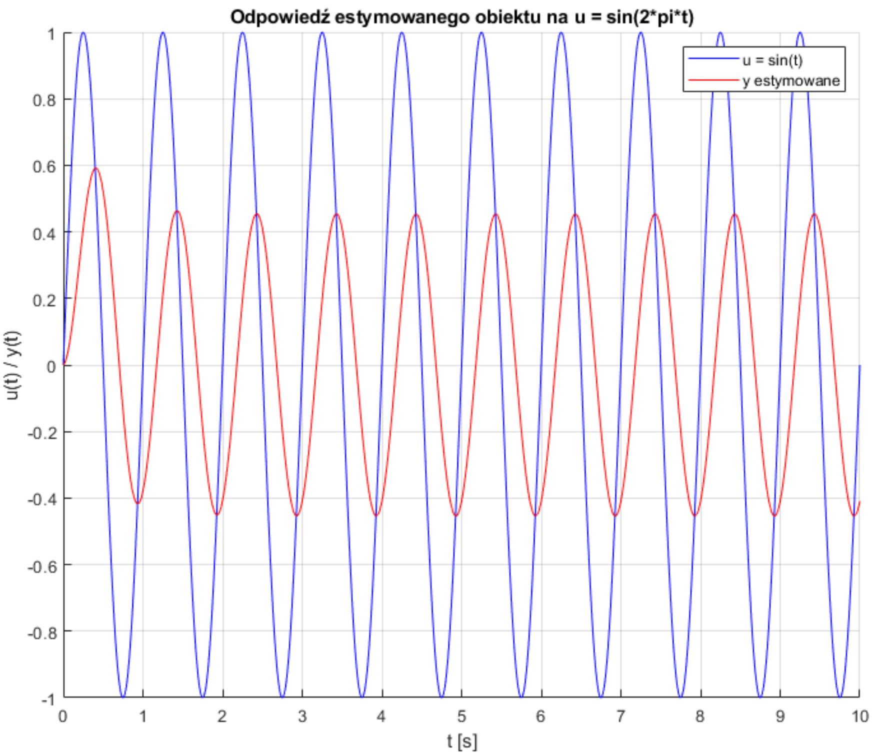
7.3 Pobudzenie innym sygnałem.

Aby potwierdzić działanie obiektu stworzono dyskretny model z podstawionymi parametrami. Następnie pobudzono go sygnałem $u(t) = sin(2\pi \cdot t)$ i wykreślono otrzymany wynik.

```
%% Odpowied  uk adu na inne wymuszenie
Ob = tf([b0, b1, 0], [1, a1, a2], Tp);
t_iw = 0:Tp:10;
u_iw = sin(2*pi*t_iw);
y_iw = lsim(Ob, u_iw, t_iw);

figure;
hold on;
title('Odpowied  estymowanego obiektu na u = sin(2*pi*t)');
plot(t_iw, u_iw, 'b', t_iw, y_iw, 'r');
legend('u = sin(t)', 'y estymowane');
xlabel('t [s]');
ylabel('u(t) / y(t)');
grid on;
```

Listing 10: Wyznaczenie odpowiedzi obiektu na sygnał sinusoidalny

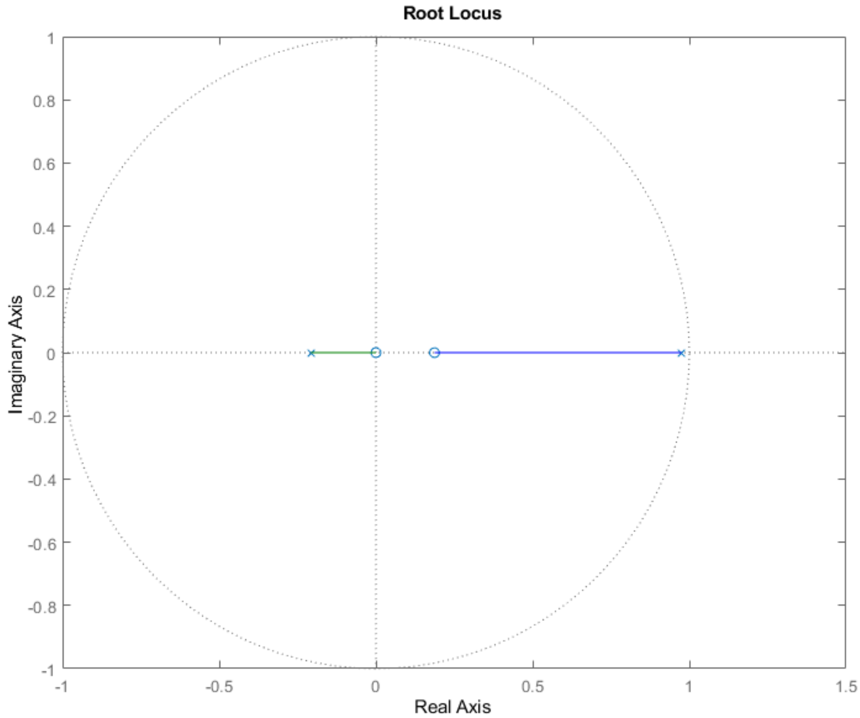


Rysunek 4: Odpowiedź obiektu na sygnał sinusoidalny

Sygnał wejściowy jest śledzony przez obiekt, co potwierdza jego poprawność.

7.4 Badanie stabilności.

Stabilność zbadano metodą za pomocą funkcji *rlocus*:



Rysunek 5: Wynik wywołania funkcji rlocus na obiekcie

Można stwierdzić, że obiekt jest stabilny - oba bieguny leżą wewnątrz okręgu jednostkowego na płaszczyźnie zmiennej zespolonej (warunek stabilności dla układów dyskretnych jest spełniony).

7.5 Wynik identyfikacji.

Analizowany obiekt można przybliżyć następującą "transmitancją":

$$G(q) = \frac{0.0451 - 0.0084 \cdot q^{-1}}{1 - 0.7670 \cdot q^{-1} - 0.2021 \cdot q^{-2}} \cdot q^{-1} \tag{5}$$

8 Podsumowanie.

Wybrany model i obliczone parametry bardzo dobrze odwzorowują obiekt rzeczywisty. Na podstawie wyznaczonych wskaźników jakości można stwierdzić, że błąd jest znikomy. Model jest dobrze zoptymalizowany (wektor parametrów Θ ma minimalny rozmiar) i łatwo rozbudowywalny (można łatwo dodać kolejny rząd obiektu, łatwo zamienić na symulacje real-time).