Politechnika Świętokrzyska w Kielcach

Wydział Elektrotechniki, Automatyki i Informatyki
Katedra Systemów Informatycznych

Kierunek:	**Laboratorium:**	
Elektrotechnika	**Metody estymacji parametrów i sygnałów**	
Grupa dziekańska:	**Temat ćwiczenia:**	**Wykonali:**
1EDXXA	**Identyfikacja parametrów modelu matematycznego silnika elektrycznego za pomocą metody simpleksów.**	
Data wykonania:	**Data oddania:**	**Ocena i podpis:**
16, 23.01.2023	27.01.2023	

Spis treści

1. WSTĘP 1
2. MODEL SILNIKA DC 2
3. SKRYPT IDENTYFIKUJĄCY PARAMETRY SILNIKA DC 2
3.1. Symulacja pracy silnika – rozwiązywanie równań różniczkowych 2
3.2. Funkcja obliczająca wskaźnik jakości i funkcja ograniczeń parametrów 3
3.3. Dopasowywanie parametrów silnika 4
3.4. Symulacje pracy silnika 4
3.5. Porównanie wykresów wartości rzeczywistych z punktem początkowym 4
3.6. Porównanie wykresów wartości rzeczywistych z optymalnymi 5
4. WYNIKI SYMULACJI 5
5. WNIOSKI 7

1. Wstęp

Ćwiczenie zostało wykonane przy użyciu środowiska Octave. Napisano skrypty wykorzystujące metodę wędrującego simpleksu do dopasowania parametrów modelu silnika do rzeczywistych parametrów silnika na podstawie określonego kryterium jakości. Z powodu braku danych empirycznych pozwalających na estymację parametrów w rzeczywistych warunkach, dane zostały wygenerowane na podstawie znanych parametrów, co również ułatwiło ocenę efektywności metody.

2. Model silnika DC

Do wykonania symulacji pracy silnika DC wykorzystano model matematyczny opisujący silnik transmitancją 2-go rzędu. Model ten uwzględnia inercję wynikającą z bezwładności wału silnika oraz inercję wynikającą z obecności indukcyjności w obwodzie twornika. Poniżej przedstawiono równania opisujące w ten sposób silnik.

$$\begin{cases} u(t) = R\dfrac{di(t)}{dt} + c\omega(t) \\ J\dfrac{d\omega(t)}{dt} = ci(t) - M_{st} \end{cases} \tag{1}$$

$$\begin{cases} \dfrac{di(t)}{dt} = \dfrac{1}{R} \cdot \big(u(t) - c\omega(t)\big) \\ \dfrac{d\omega(t)}{dt} = \dfrac{1}{J} \cdot (ci(t) - M_{st}) \end{cases} \tag{2}$$

3. Skrypt identyfikujący parametry silnika DC

3.1. Symulacja pracy silnika – rozwiązywanie równań różniczkowych

```
function [w, i] = simulate(t, x0, params)
    x = x0;
    [t, y] = ode45(@(t, x) f(t, x, params), t, x);
    w = y(:, 2);
    i = y(:, 1);
end

function [dx] = f(t, x, params)
    Pn = params(1);
    Un = params(2);
    In = params(3);
    nn = params(4);
    R  = params(5);
    L  = params(6);
    J  = params(7);

    U  = Un;
    wn = nn * pi/30;
    Mn = Pn * 1000 / wn;
    Mst = Mn;
    c = (Un - R*In) / wn;

    i = x(1);
    w = x(2);
    dx = [-1/L * R * i - c/L * w + U/L; c/J * i - Mst/J];
end
```

Listing 3.1 - Rozwiązywanie równań różniczkowych.

3.2. Funkcja obliczająca wskaźnik jakości i funkcja ograniczeń parametrów

```
function [Q] = quality(params_test)

    Q = qbounds(params_test);
    if Q != 0
        return;
    endif

    t = linspace(0, 0.3, 100);
    [wreal, ireal] = simulate(t, [0, 0], [90.5, 600, 212, 2310, 0.122,
0.00135, 0.38]);
    [wtest, itest] = simulate(t, [0, 0], params_test);
    Qw = 1 / size(wreal, 1) * trapz((wreal - wtest).^2);
    Qi = 1 / size(ireal, 1) * trapz((ireal - itest).^2);
    Q = sqrt(Qw^2 + 0.1 * Qi^2);
End

function Q = qbounds(params)
    Pn = params(1);
    Un = params(2);
    In = params(3);
    nn = params(4);
    R  = params(5);
    L  = params(6);
    J  = params(7);

    qmax = 1e10;

    Q = 0;

    if Pn < 0 Q = qmax; endif
    if Un < 0 Q = qmax; endif
    if In < 0 Q = qmax; endif
    if nn < 0 Q = qmax; endif
    if R < 0  Q = qmax; endif
    if L < 0  Q = qmax; endif
    if J < 0  Q = qmax; endif

    if Pn > 10000 Q = qmax; endif
    if Un > 10000 Q = qmax; endif
    if In > 10000 Q = qmax; endif
    if nn > 10000 Q = qmax; endif
    if  R > 10000 Q = qmax; endif
    if  L > 10000 Q = qmax; endif
    if  J > 10000 Q = qmax; endif
end
```

Listing 3.2 - Funkcja obliczająca wskaźnik jakości i funkcja ograniczeń parametrów

3.3. Dopasowywanie parametrów silnika

```
params_orig = [90.5, 600, 212, 2310, 0.122, 0.00135, 0.38];
params_init = params_orig * 0.6;
options = optimset('Display', 'iter', 'MaxFunEvals', 4000, 'MaxIter', 2000,
'TolFun', 1e-10, 'TolX', 1e-10);
params_opt = fminsearch(@quality, params_init, options);
display(params_opt);
display(params_orig);
```

Listing 3.3 - Dopasowywanie parametrów silnika

3.4. Symulacje pracy silnika

```
t = linspace(0, 0.3, 1000);
[wreal, ireal] = simulate(t, [0, 0], params_orig);
[winit, iinit] = simulate(t, [0, 0], params_init);
[wopt, iopt] = simulate(t, [0, 0], params_opt);
```

Listing 3.4 - Symulacje pracy silnika

3.5. Porównanie wykresów wartości rzeczywistych z punktem początkowym

```
figure(1);
subplot(2, 1, 1);
plot(t, ireal, t, iopt, 'k');
grid on;
title('Prad twornika - oryginal vs optymalizacja');
xlabel('t [s]');
ylabel('i [A]');
legend('Oryginal', 'Optymalizacja');

subplot(2, 1, 2);
plot(t, wreal * 30/pi, t, wopt * 30/pi, 'k');
grid on;
title('Predkosc obrotowa - oryginal vs optymalizacja');
xlabel('t [s]');
ylabel('w [obr/min]');
legend('Oryginal', 'Optymalizacja');
```

Listing 3.5 - Porównanie wykresów wartości rzeczywistych z punktem początkowym

3.6. Porównanie wykresów wartości rzeczywistych z optymalnymi

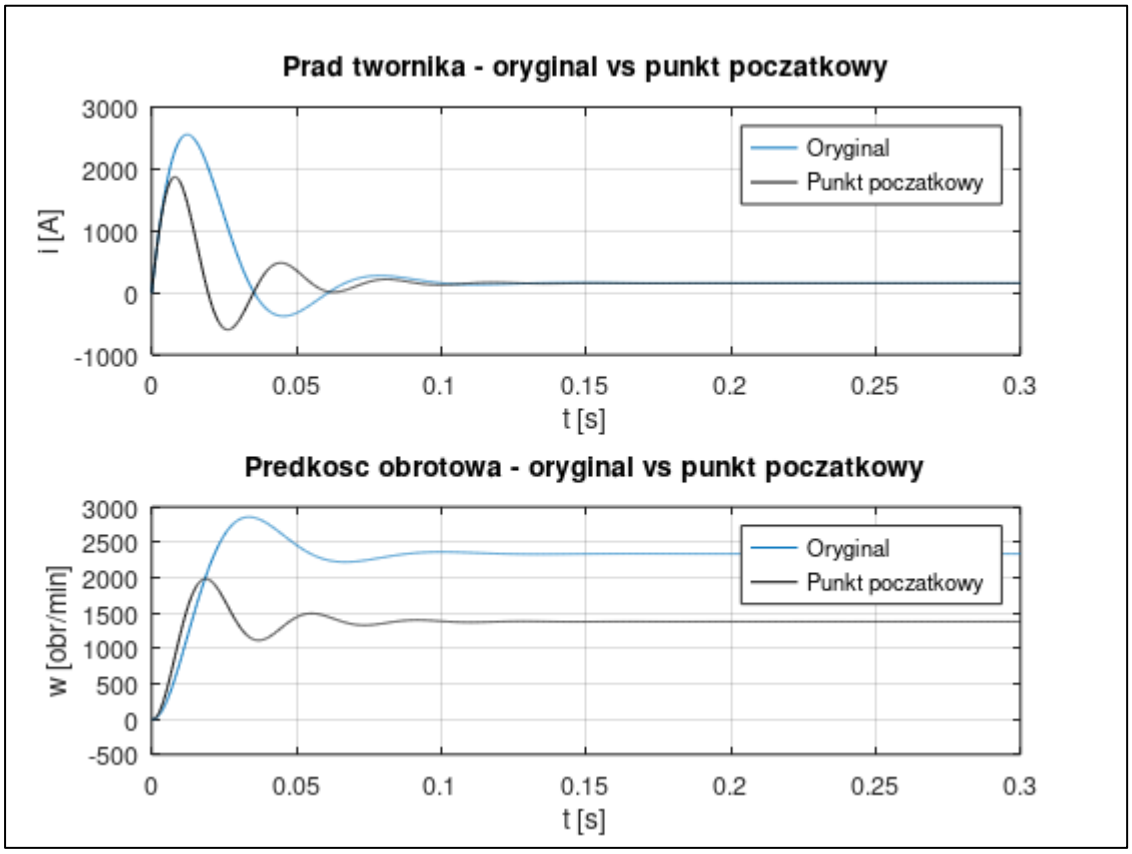
```
figure(2);
subplot(2, 1, 1);
plot(t, ireal, t, iinit, 'k');
grid on;
title('Prad twornika - oryginal vs punkt poczatkowy');
xlabel('t [s]');
ylabel('i [A]');
legend('Oryginal', 'Punkt poczatkowy');

subplot(2, 1, 2);
plot(t, wreal * 30/pi, t, winit * 30/pi, 'k');
grid on;
title('Predkosc obrotowa - oryginal vs punkt poczatkowy');
xlabel('t [s]');
ylabel('w [obr/min]');
legend('Oryginal', 'Punkt poczatkowy');

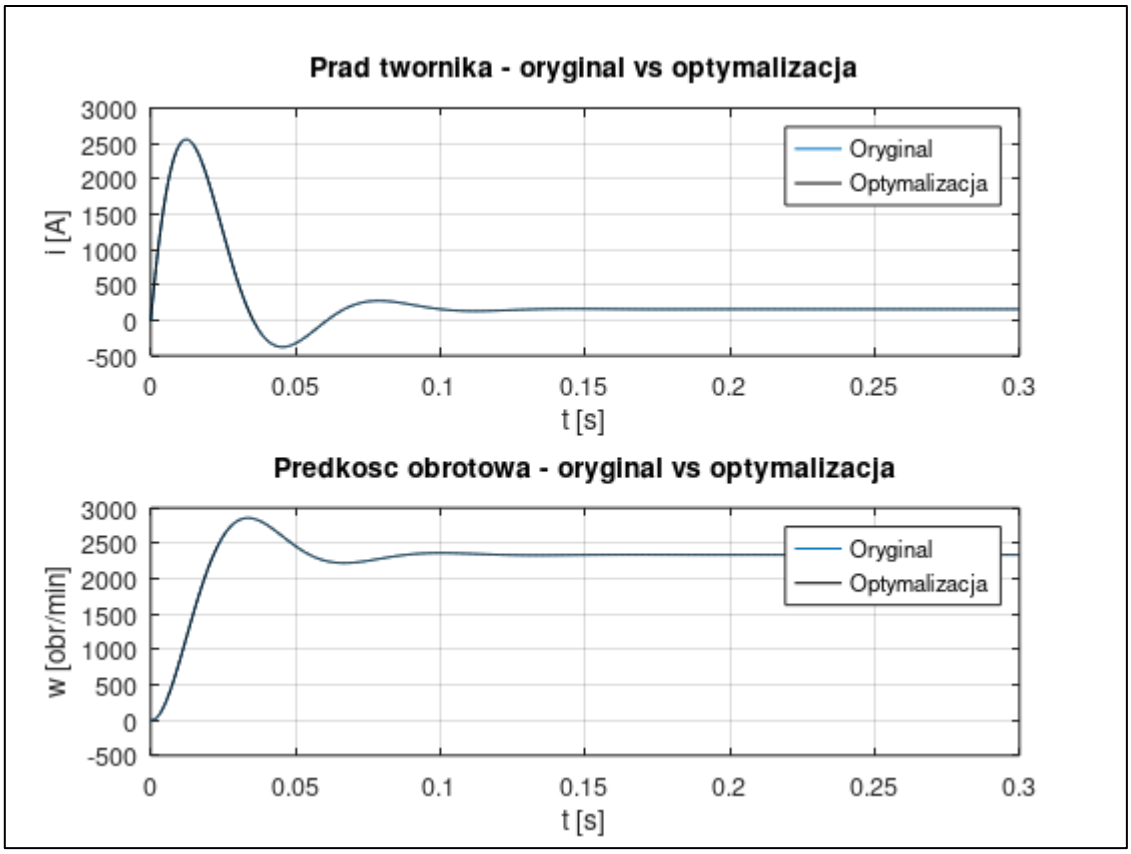
e1 = quality(params_init);
e2 = quality(params_opt);
fprintf('Blad zostal zredukowany z wartosci %g do wartosci %g, czyli o %g',
e1, e2, e1 - e2);
```

Listing 3.6 - Porównanie wykresów wartości rzeczywistych z optymalnymi

4. Wyniki symulacji



Rysunek 1 - Porównanie rzeczywistych przebiegów prędkości oraz prądu do przebiegów uzyskanych w wyniku symulacji przy parametrach z punktu początkowego



Rysunek 2 - Porównanie rzeczywistych przebiegów prędkości oraz prądu do przebiegów uzyskanych w wyniku symulacji przy parametrach z optymalizacji

```
Iter. 1542,  how = contract,  nf = 2469,  f = 7.5611e-17  (10.9%)
Iter. 1543,  how = contract,  nf = 2471,  f = 7.5611e-17  (0.0%)
Iter. 1544,  how = reflect,   nf = 2472,  f = 7.5611e-17  (0.0%)
Iter. 1545,  how = reflect,   nf = 2473,  f = 7.5611e-17  (0.0%)
Iter. 1546,  how = contract,  nf = 2475,  f = 3.8379e-17  (12.5%)
Algorithm converged.  Simplex size 9.4863e-11 <= 1.0000e-10 and step in function value 1.595
params_opt =

   5.2857e+02   4.5231e+03   1.2720e+03   1.7897e+03   9.1970e-01   1.0177e-02   2.8646e+00

params_orig =

   9.0500e+01   6.0000e+02   2.1200e+02   2.3100e+03   1.2200e-01   1.3500e-03   3.8000e-01

Blad zostal zredukowany z wartosci 69880.1 do wartosci 3.83793e-17, czyli o 69880.1>>
```

Rysunek 3 - Wyniki zapisane w konsoli Octave

5. Wnioski

Skrypt dopasował parametry do przebiegów prądu i prędkości w taki sposób, że idealnie pokrywają się z wykresami rzeczywistymi, a błąd jest bliski zeru. Warunkami początkowymi było 60% wartości oryginalnych wszystkich parametrów, którymi opisany został silnik. Optymalizacja trwała kilka minut, a algorytm optymalizacji wykonał 1546 iteracji. Wyniki optymalizacji są więc bardzo zadowalające. Niestety optymalizacja nie poszła zgodnie z planem. Dopasowane zostały parametry różniące się znacząco od rzeczywistych parametrów silnika. Jest to dowód na to, że mogą istnieć dwa różne silniki o różnych parametrach, ale charakteryzujące się takimi samymi przebiegami prędkości i prądu podczas rozruchu. Proces optymalizacji jest więc niepewny i może dawać fałszywe wyniki. Aby ulepszyć proces identyfikacji parametrów silnika, konieczne jest sprecyzowanie parametrów silnika, zakresu wartości tych parametrów, bądź manipulacja kryterium jakości. Lepsze efekty mogą również dać algorytmy genetyczne, które polegają na losowości, dzięki czemu algorytm ma szansę „trafić w punkt”.