Raport do rozwiązania projektu z przedmiotu metody numeryczne

Mateusz Marzec

Listopad 2021r.

Abstract

Ten raport podzielony jest na cztery części zgodnie z częściami projektu. Każda z nich, zgodnie z wymaganiami projektu, zawiera opis modelu matematycznego, modelu numerycznego, implementację modelu, wyniki oraz ich weryfikację.

Chapter 1

Część 1

1.1 Opis modelu matematycznego

Model matematyczny danego układu elektrycznego jest szczegółowo opisany w treści zadania - jest to układ trzech liniowych równań różniczkowych pierwszego rzędu ze zmiennymi będącymi: prądem w pierwszym obwodzie, prądem w drugim obwodzie oraz napięciem na kondensatorze. Przybliżone rozwiązanie tego układu zostanie wykonane metodami numerycznymi zgodnie z treścią zadania. Wyniki zostaną zweryfikowane inną metodą numeryczną (patrz kolejne rozdziały) oraz metodą półanalityczną zaprezentowaną poniżej.

Dany układ równań różniczkowych można rozwiązać metodą operatorową. Przyjmując $\mathcal{L}i_1(t) = I_1(s), \mathcal{L}i_2(t) = I_2(s), \mathcal{L}u_c(t) = U(s) oraz \mathcal{L}e(t) = E(s)$, układ przyjmuje postać:

$$\begin{cases} E(s) = U(s) + R_1 I_1(s) + L_1(sI_1(s)) + M(sI_2(s)) \\ 0 = R_2 I_2(s) + L_2(sI_2(s)) + M(sI_2(s)) \\ I_1(s) = C(sU(s)) \end{cases} \tag{1.1}$$

Żeby nie komplikować zagadnienia, weryfikacja tą metodą zostanie wykonana przyjmując $e(t) = \sin(t)$ (tak jak zostały podane przykładowe rozwiązania w treści projektu dla tego podstawienia!), a więc $E(s) = \frac{1}{(s-i)(s+i)}$ (i oznacza jednostkę urojoną, nie mylić z prądem). Podstawiając liczby (i opuszczając jednostki, gdyż wszystkie są zgodne z SI), dochodzimy do następujących równań (szczegóły obliczeń na końcu raportu):

$$\begin{cases} U(s) = E(s)\frac{s+2}{1.436s^3+3.05s^2+1.1s+2} \\ I_1(s) = E(s)\frac{0.5s^2+s}{1.436s^3+3.05s^2+1.1s+2} \\ I_2(s) = E(s)\frac{0.08s^3+0.16s^2}{1.436s^4+5.922s^3+7.2s^2+4.2s+4} \end{cases} \tag{1.2}$$

Aby rozwiązać te równania, należy znaleźć bieguny powyższych funkcji. Niestety jest to zbyt skomplikowane, aby zrobić to analitycznie. Tutaj więc pojawia się element "pół-" powyższego, jak dotąd analitycznego modelu. Korzystając z wbudowanej w MATLAB funkcji `roots()`, przybliżone bieguny powyższych funkcji wyglądają następująco:

funkcja	s_1	s_2	s_3	s_4	s_5	s_6
$I_2(s)$	$-i$	$+i$	$-0.81i - 0.02$	$0.81i - 0.02$	-2.08	-2.0
$I_1(s)$	$-i$	$+i$	$-0.81i - 0.02$	$0.81i - 0.02$	-2.08	-
$U(s)$	$-i$	$+i$	$-0.81i - 0.02$	$0.81i - 0.02$	-2.08	-

Wszystkie bieguny są (na szczęście!) biegunami prostymi. To umożliwia skorzystanie z twierdzenia o rozkładzie, dzięki czemu dochodzimy do następujących równań (dla przejrzystości pozwoliłem sobie na zaokrąglenie współczynników):

$$\begin{aligned} u(t) =& -0.003e^{-2.078t} + 3.522e^{-0.023t}\sin(0.818t) - 2.879\sin(t) \\ &+ 0.45e^{-0.023t}\cos(0.818t) - 0.446\cos(t) \\ i_1(t) =& 0.003e^{-2.078t} - 0.224e^{-0.023t}\sin 0.818t + 0.223\sin(t) \\ &+ 1.436e^{-0.023t}\cos(0.818t) - 1.439\cos(t) \\ i_2(t) =& 0.013e^{-2.078t} - 0.087e^{-0.023t}\sin(0.818t) + 0.1\sin(t) \\ &+ 0.019e^{-0.023t}\cos(0.818t) - 0.032\cos(t) \end{aligned} \tag{1.3}$$

Na kartce wyliczyłem jedynie funkcję $u(t)$. Upewniwszy się, że oprogramowanie WolframAlpha robi to poprawnie, pozostałe funkcje "wyprowadziłem" korzystając z tegoż narzędzia. Mając wyprowadzone jawnie funkcje opisujące model, można weryfikować wyniki modelu numerycznego we wszystkie strony.

1.2 Opis modelu numerycznego

Model numeryczny obejmuje wykorzystanie prostej metody Eulera, oraz jej wariantu, zwanego w treści zadania "metodą ulepszoną", a w literaturze określanej jako *midpoint method*. Według metody prostej:

$$\begin{cases} i_1(t + \Delta t) \approx i_1(t) + i_1'(i_1, i_2, u_c, e, t)\Delta t \\ i_2(t + \Delta t) \approx i_2(t) + i_2'(i_1, i_2, u_c, e, t)\Delta t \\ u_c(t + \Delta t) \approx u_c(t) + u_c'(i_1)\Delta t \end{cases} \tag{1.4}$$

Zaczynając od wartości początkowych dla $t = 0$ (podanych w zadaniu) i kolejno w każdym obrocie pętli powtarzając powyższe obliczenia aż $t = 30s$, uzyskuję przybliżone numeryczne rozwiązanie układu równań.

Wariant metody Eulera jest liczony analogicznie, jedyną różnicą jest inna postać czynnika po prawej stronie przed Δt.

Aby sprawdzić poprawność uzyskanych wyników, rozwiązuję układ równań w sposób przybliżony posługując się metodą Rungego-Kutty. Ta metoda jest, podobnie do powyższych, metodą iteracyjną; jedyną różnicą jest bardziej skomplikowany sposób wyliczania przyrostu wartości funkcji. Ponieważ treść projektu poleca jedynie dokonać weryfikacji wyniku dowolną metodą, nie muszę implementować tej metody samodzielnie i nie będę przytaczał tutaj wzorów, dostępnych powszechnie w literaturze. Zamiast tego posłużę się funkcją `ode45()`, która w MATLABIE jest implementacją powyższej metody.

Aby nie komplikować rozwiązania projektu, weryfikację wyników przeprowadzam jedynie dla wymuszenia $e = \sin(t)$. Jest oczywiste, że analogiczna może zostać przeprowadzona dla dowolnej funkcji $e(t)$.

1.3 Opis implementacji

1.3.1 Metoda prosta Eulera

Kod wyliczający całą część 1. przedstawiam poniżej. Wyjaśnienia są obecne w formie komentarzy. Aby uruchomić skrypt, należy wpisać *Czesc1(n)*, gdzie n wynosi 1- dla wymuszenia pulsacyjnego, 2- dla wymuszenia $e = 240\sin(t)$, 3- dla wymuszenia $e = 210\sin(2\pi \cdot 5)$, 4- dla wymuszenia $e = 120\sin(2\pi \cdot 50)$, 5- dla wymuszenia $e = \sin(t)$.

```
function [result] = Czesc1(n) %n=1,2,3,4,5 dla kolejnych wymuszeń.
n=5 dla e=sin(t)
    global licznikWykresow;
    licznikWykresow = 0;
    R1 = 0.1;
    R2 = 10;
    C = 0.5;
    L1 = 3;
    L2 = 5;
    M = 0.8;
    di1dt = @(i1,i2,uC,E) ((1/((L1/M)-(M/L2)))*(((-R1/M)*i1)+((R2/L2)*i2)
    -((1/M)*uC)+((1/M)*E)));
    di2dt = @(i1,i2,uC,E) ((1/((M/L1) - (L2/M)))*(((-R1/L1)*i1)+((R2/M)*i2)
    -((1/L1)*uC)+((1/L1)*E)));
    %di1dt = @(i1,i2,uC,E) ((-1/M) * (R2*i2 + L2 * di2dt(i1,i2,uC,E)));
    duCdt = @(i1) ((1/C)*i1);
    global h;
```

```
    h = 0.0001;
    global tmax;
    tmax = 300;
    t = 0:h:tmax;
    E = zeros(1, length(t));
    i1 = zeros(1, length(t));
    i2 = zeros(1, length(t));
    uC = zeros(1, length(t));
    i = 1;
    E(1) = fE(n, t(1)); %funkcja fE liczy E(t) dla roznych (n)
     podpunktow tego zadania
    while (t(i) < tmax) %implementacja pętli opisanej powyżej
        E(i+1) = fE(n, t(i+1));
        i1(i+1) = i1(i) + (h * di1dt(i1(i),i2(i),uC(i),E(i)));
        i2(i+1) = i2(i) + (h * di2dt(i1(i),i2(i),uC(i),E(i)));
        uC(i+1) = uC(i) + (h * duCdt(i1(i)));
        i = i + 1;
    end
    %odtąd rysowanie wykresów
    Rysuj(t, uC, E, i1, i2);
    LiczIRysujAnal(n);%liczy i rysuje weryfikację półanalityczną
    LiczIRysujMacierze(n);%liczy i rysuje weryfikację drugą metodą numeryczną
    result = 'wykonano';
end

function [E] = fE(n, t)
    global h;
    switch (n(1))
        case 1
            t2 = (1 / h) * t(1);
            T = (1 / h) * 3;
            div = rem(t2, T);
            if (div < (T/2))    %dobieram takie h żeby T/2 było całkowite
                E = 120;
            else
                E = 0;
            end
        case 2
            E = 240*sin(t(1));
        case 3
            E = 210*sin(2 * pi * 5 * t(1));
```

```
        case 4
            E = 120*sin(2 * pi * 50 * t(1));
        case 5
            E = sin(t(1));
        otherwise
            E = 1000;
    end
end

function [result] = LiczIRysujAnal(n)   %liczy i rysuje met.półanalityczną
    global h;
    global tmax;
    t = 0:h:tmax;
    i1Anal = zeros(1,length(t));
    i2Anal = zeros(1,length(t));
    uAnal = zeros(1,length(t));
    if (n ~= 5)
        return;
    end
    for i=1:length(t)
        uAnal(i) = -0.0029943*exp(-2.0779*t(i)) +
         3.52233*exp(-0.023*t(i))*sin(0.8184*t(i)) - 2.87857*sin(t(i)) +
          0.449056*exp(-0.023*t(i))*cos(0.8184*t(i)) - 0.446062*cos(t(i)) + 0;

        i1Anal(i) = 0.00311093*exp(-2.0779*t(i)) -
         0.224261*exp(-0.023*t(i))*sin(0.8184*t(i)) + 0.223031*sin(t(i)) +
          1.43617*exp(-0.023*t(i))*cos(0.8184*t(i)) - 1.43928*cos(t(i)) + 0;

        i2Anal(i) = 0.0132769*exp(-2.0779*t(i)) -
         0.0870447*exp(-0.023*t(i))*sin(0.8184*t(i)) + 0.0992511*sin(t(i)) +
          0.0185062*exp(-0.023*t(i))*cos(0.8184*t(i)) - 0.0317831*cos(t(i)) + 0;

    end
    RysujAnal(t, uAnal, i1Anal, i2Anal);
    result = 'obliczono analitycznie';
end

function [result] = Rysuj(t, uC, E, i1, i2) %rysuje obliczenia met. Eulera
    global licznikWykresow;
    licznikWykresow = licznikWykresow + 1;
    figure(licznikWykresow);
```

```
    hold on;
    plot(t, i1, 'DisplayName', 'i1(t)');
    plot(t, i2, 'DisplayName', 'i2(t)');
    %plot(t, uC, 'DisplayName', 'uC(t)');
    yline(0);
    xlabel('t');
    ylabel('i');
    %ylabel('i(t) lub u(t)');
    title('i(t) - Euler');
    %title('i(t) i u(t) - Euler');
    legend;
    hold off;
    licznikWykresow = licznikWykresow + 1;
    figure(licznikWykresow);
    hold on;
    plot(t, E, 'DisplayName', 'E(t)');
    plot(t, uC, 'DisplayName', 'uC(t)');
    yline(0);
    xlabel('t');
    ylabel('u');
    title('u(t) - Euler');
    legend;
    hold off;
    result = 'narysowano';
end

function [result] = RysujAnal(t, uAnal, i1Anal, i2Anal)
    global licznikWykresow;
    licznikWykresow = licznikWykresow + 1;
    figure(licznikWykresow);
    hold on;
    plot(t, i1Anal, 'DisplayName', 'i1(t)');
    plot(t, i2Anal, 'DisplayName', 'i2(t)');
    plot(t, uAnal, 'DisplayName', 'E(t)');
    yline(0);
    xlabel('t');
    %ylabel('i');
    ylabel('i(t) lub u(t)');
    %title('i(t) - półanalitycznie');
    title('i(t) lub u(t) - półanalitycznie');
    legend;
```

```
    hold off;
    licznikWykresow = licznikWykresow + 1;
    figure(licznikWykresow);
    hold on;
    plot(t, uAnal, 'DisplayName', 'E(t)');
    yline(0);
    xlabel('t');
    ylabel('u');
    title('u(t) - półanalitycznie');
    legend;
    hold off;
    result = 'narysowano';
end

function [result] = LiczIRysujMacierze(n)   %liczy i rysuje met. Rungego-Kutty
    global licznikWykresow;
    global tmax;
    global h;
    if (n(1) == 5)
        [t,y] = ode45(@odefun, [0 tmax], [0 0 0]);
        licznikWykresow = licznikWykresow + 1;
        figure(licznikWykresow);
        plot(t, y);
        yline(0);
        xlabel('t');
        ylabel('i(t) lub u(t)');
        title('i(t) i u(t) - Runge-Kutta');
        legend('i1(t)','i2(t)','u(t)');
    end
    result = 'wykonano obliczenia macierzowe';
end

function dydt = odefun(t, y)%dydt to macierz będąca de facto układem równań
    R1 = 0.1;
    R2 = 10;
    C = 0.5;
    L1 = 3;
    L2 = 5;
    M = 0.8;
    dydt = [(1/(L1/M - M/L2))*((-R1/M)*y(1) + (R2/L2)*y(2) - (1/M)*y(3)
     + (1/M)*sin(t));
```

```
        (1/(M/L1 - L2/M))*((-R1/L1)*y(1) + (R2/M)*y(2) - (1/L1)*y(3)
         + (1/L1)*sin(t));
        (1/C)*y(1)];
end
```

1.3.2 Wariant metody Eulera - midpoint method

Implementacja wariantu metody Eulera przedstawiona jest poniżej w postaci osobnej funkcji. Należy uruchomić skrypt Czesc1_2.m w analogiczny sposób, jak w implementacji metody prostej.

```
function [i1_2, i2_2, uC_2, E_2, t_2] = MidPointEuler(n)
    R1 = 0.1;
    R2 = 10;
    C = 0.5;
    L1 = 3;
    L2 = 5;
    M = 0.8;
    global h;
    global tmax;
    t = 0:h:tmax;
    di1dt = @(i1,i2,uC,E) ((1/((L1/M)-(M/L2)))*(((-R1/M)*i1)+((R2/L2)*i2)
    -((1/M)*uC)+((1/M)*E)));
    di2dt = @(i1,i2,uC,E) ((1/((M/L1) - (L2/M)))*(((-R1/L1)*i1)+((R2/M)*i2)
    -((1/L1)*uC)+((1/L1)*E)));
    duCdt = @(i1) ((1/C)*i1);
    E = zeros(1,length(t));
    i1 = zeros(1,length(t));
    i2 = zeros(1, length(t));
    uC = zeros(1, length(t));
    i = 1;
    E(1) = fE(n, t(1));
    while (t(i) < 30)
        E(i+1) = fE(n, t(i+1));
        i1c = di1dt(i1(i),i2(i),uC(i),E(i));
        i1(i+1) = i1(i) + (h * di1dt(i1(i)+(h/2)*i1c,i2(i)+(h/2)*i1c, uC(i)
        +(h/2)*i1c, E(i)+(h/2)*i1c));
        i2c = di2dt(i1(i),i2(i),uC(i),E(i));
        i2(i+1) = i2(i) + (h * di2dt(i1(i)+(h/2)*i2c,i2(i)+(h/2)*i2c,uC(i)
        (h/2)*i2c,E(i)+(h/2)*i2c));
        uC2 = duCdt(i1(i));
```

```
        uC(i+1) = uC(i) + (h * duCdt(i1(i)+(h/2)*uC2));
        i = i + 1;
    end
    i1_2 = i1;
    i2_2 = i2;
    uC_2 = uC;
    E_2 = E;
    t_2 = t
end
```

1.4 Wyniki i ich weryfikacja

1.4.1 Metoda prosta Eulera

Stosując wymuszenie pulsacyjne, gdzie $e(t)$ wynosi okresowo 120V lub 0V, uzyskuję przebiegi pokazane na rysunkach 1.1 i 1.2.

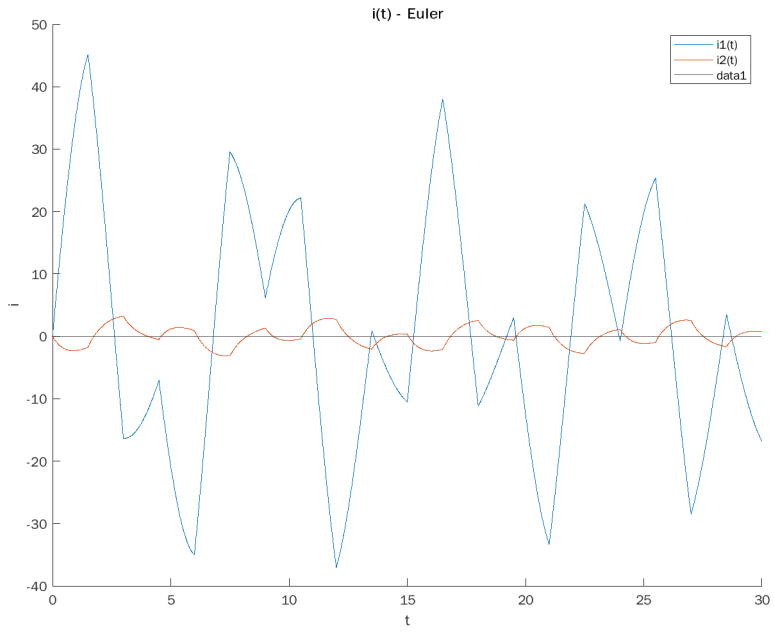


Figure 1.1: i(t) dla wymuszenia pulsacyjnego

Stosując wymuszenie, gdzie $e(t) = 240\sin(t)$, uzyskuję przebiegi pokazane na rysunkach 1.3 i 1.4.

Stosując wymuszenie, gdzie $e(t) = 210\sin(2\pi \cdot 5)$, uzyskuję przebiegi pokazane na rysunkach 1.5 i 1.6.

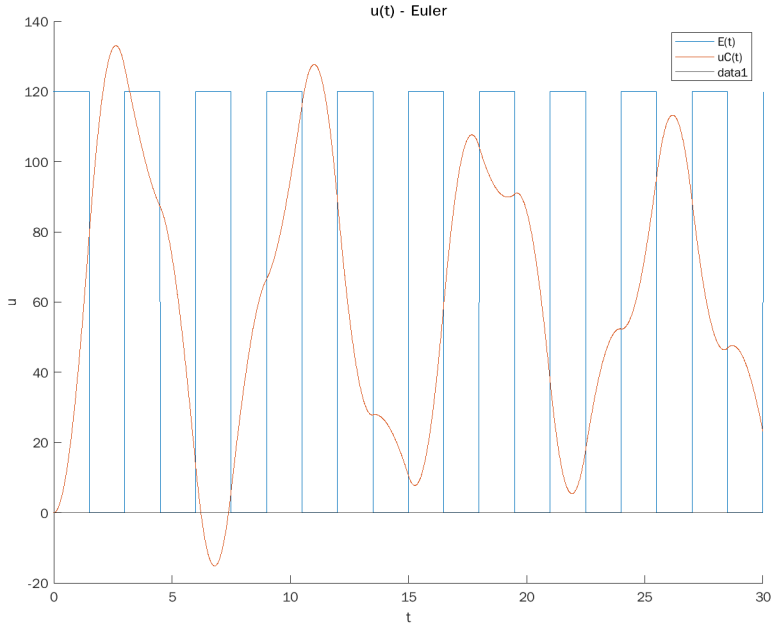


Figure 1.2: u(t) dla wymuszenia pulsacyjnego

Stosując wymuszenie, gdzie $e(t) = 120\sin(2\pi \cdot 50)$, uzyskuję przebiegi pokazane na rysunkach 1.7, 1.8 i 1.9.

A teraz, celem weryfikacji wyników, prezentuję przebiegi (wszystkie trzy zmienne na jednym wykresie) dla wymuszenia $e = \sin(t)$ i trzech metod: Eulera, Rungego-Kutty i półanalitycznej - rysunki 1.10, 1.11 i 1.12.

Celem dalszej weryfikacji wyników prezentuję analogiczne przebiegi, lecz wydłużając czas symulacji do 300s (aby upewnić się, że osiągany jest stan stacjonarny) - rysunki 1.13, 1.14 i 1.15.

Jak widać na powyższych przebiegach, wszystkie trzy metody dają przebiegi o podobnym kształcie oraz wszystkie są zbieżne od stanu stacjonarnego. Co więcej, obie metody numeryczne: Eulera i Rungego-Kutty dają niemal identyczne wyniki. Można to uznać za pozytywną weryfikację metody Eulera.

Ciekawą obserwacją jest nieco inna wysokość amplitud w przypadku metody półanalitycznej. Możliwym wyjaśnieniem tej obserwacji jest fakt, że błąd dwóch metod numerycznych popełniany jest w tym samym miejscu (iteracje), zaś błąd metody półanalitycznej zależy od numerycznie wyliczonych biegunów transformat.

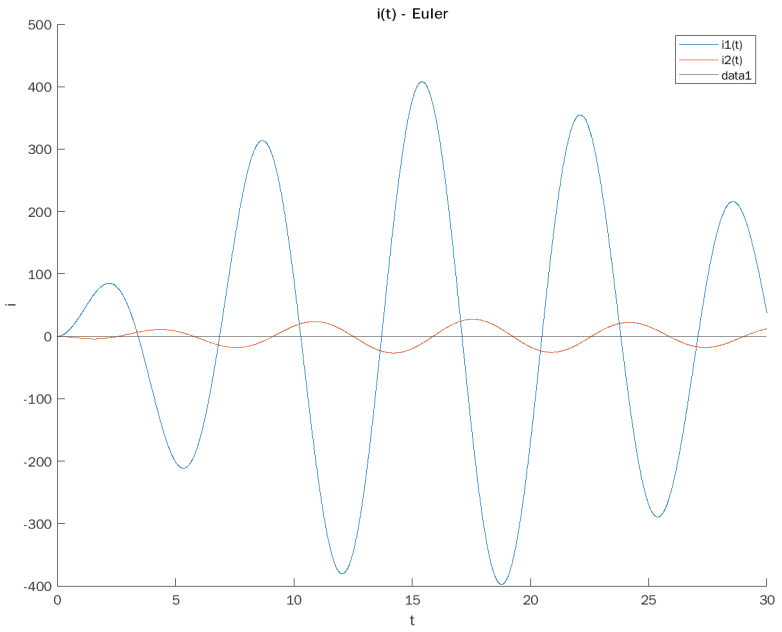


Figure 1.3: i(t) dla wymuszenia $e(t) = 240\sin(t)$

1.4.2 Wariant metody Eulera - midpoint method

Stosując wymuszenie pulsacyjne, gdzie $e(t)$ wynosi okresowo 120V lub 0V, uzyskuję przebiegi pokazane na rysunkach 1.16 i 1.17.

Stosując wymuszenie, gdzie $e(t) = 240\sin(t)$, uzyskuję przebiegi pokazane na rysunkach 1.18 i 1.19.

Stosując wymuszenie, gdzie $e(t) = 210\sin(2\pi \cdot 5)$, uzyskuję przebiegi pokazane na rysunkach 1.20 i 1.21.

Stosując wymuszenie, gdzie $e(t) = 120\sin(2\pi \cdot 50)$, uzyskuję przebiegi pokazane na rysunkach 1.22 i 1.23.

Widać, że obie metody Eulera - metoda prosta i jej wariant - dają podobne wyniki. Na rysunku 1.24 - celem weryfikacji - przedstawiłem bezwzględną różnicę pomiędzy prądami wyliczonymi przez obie metody przy przebiegu wydłużonym do 300s dla wymuszenia $e(t) = \sin(t)$.

Widać, że bezwlęgna różnica pomiędzy obiema metodami w przypadku prądów nie jest duża i zmniejsza się wraz ze stabilizacją układu. Jest to prawdopodobnie spowodowane zmniejszającą się dynamiką przebiegu prądów wraz z postępującą stabilizacją układu.

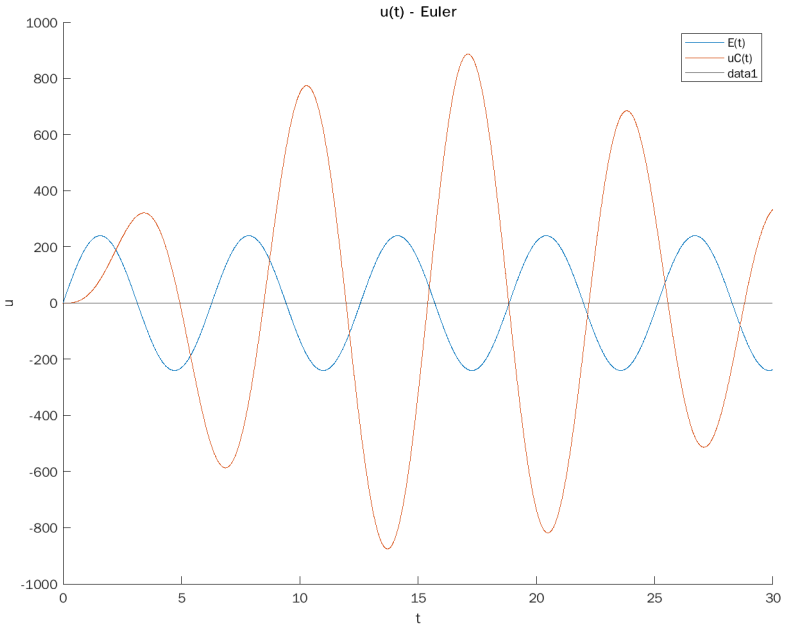


Figure 1.4: u(t) dla wymuszenia $e(t) = 240\sin(t)$

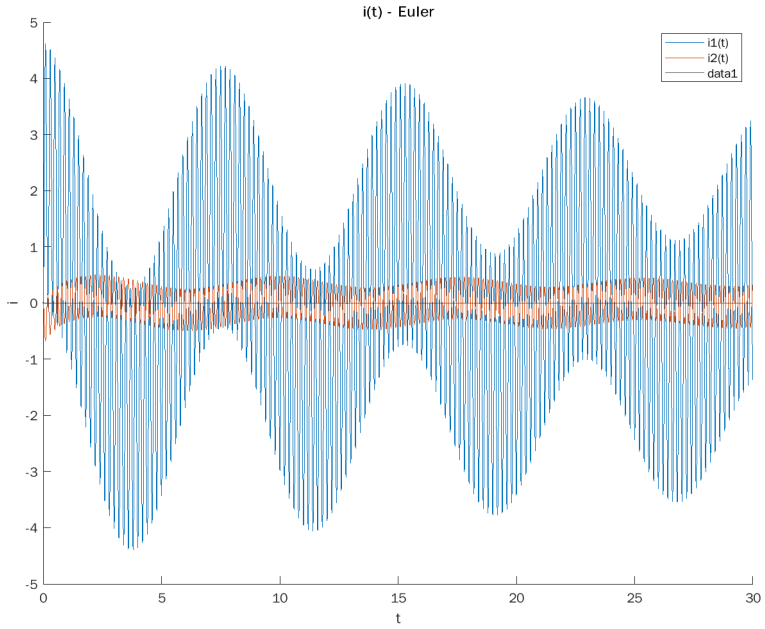


Figure 1.5: i(t) dla wymuszenia $e(t) = 210\sin(2\pi \cdot 5)$

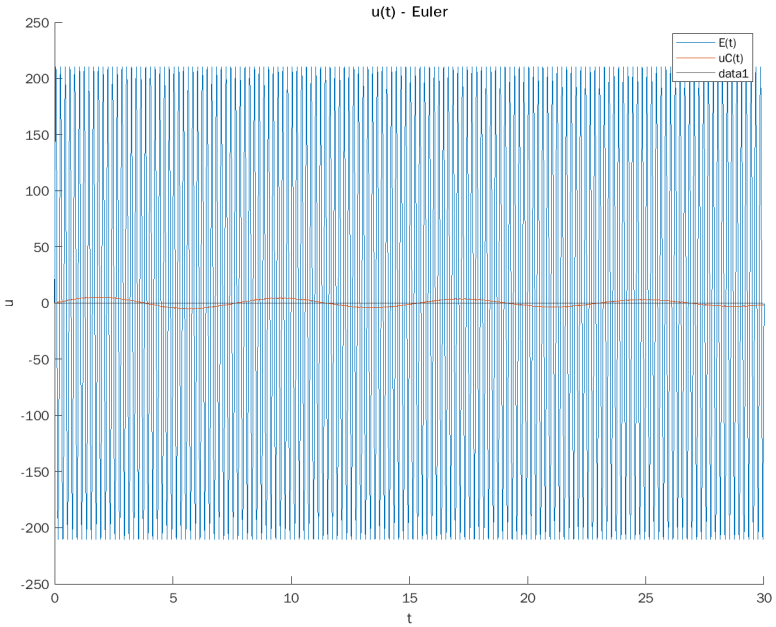


Figure 1.6: u(t) dla wymuszenia $e(t) = 210\sin(2\pi \cdot 5)$

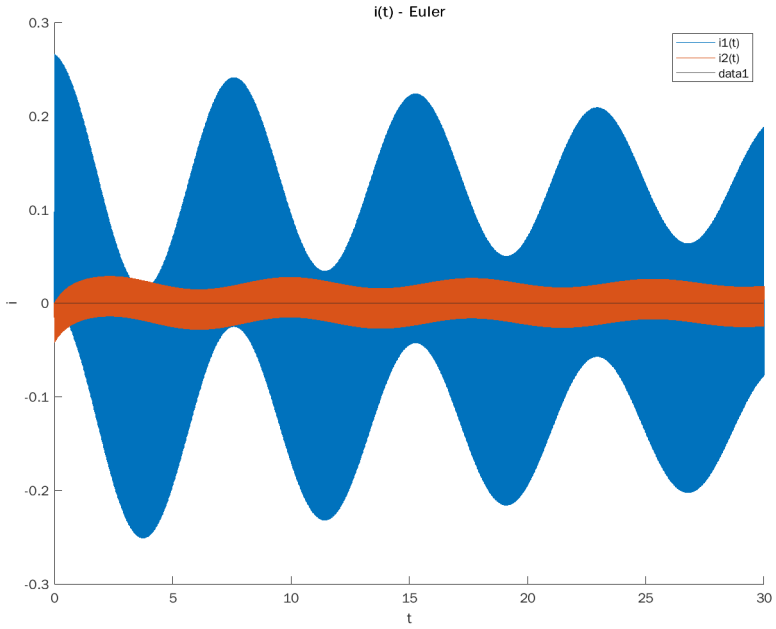


Figure 1.7: i(t) dla wymuszenia $e(t) = 120\sin(2\pi \cdot 50)$

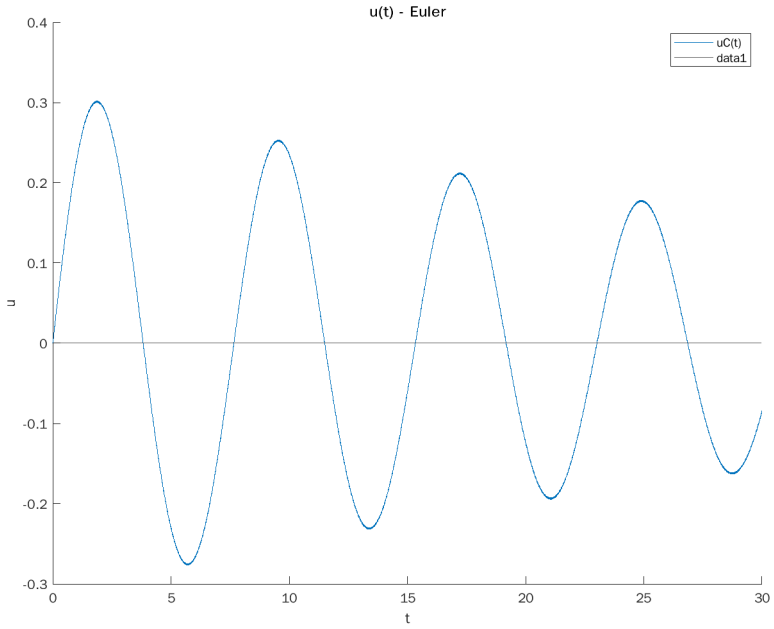


Figure 1.8: $u_c(t)$ dla wymuszenia $e(t) = 120\sin(2\pi \cdot 50)$

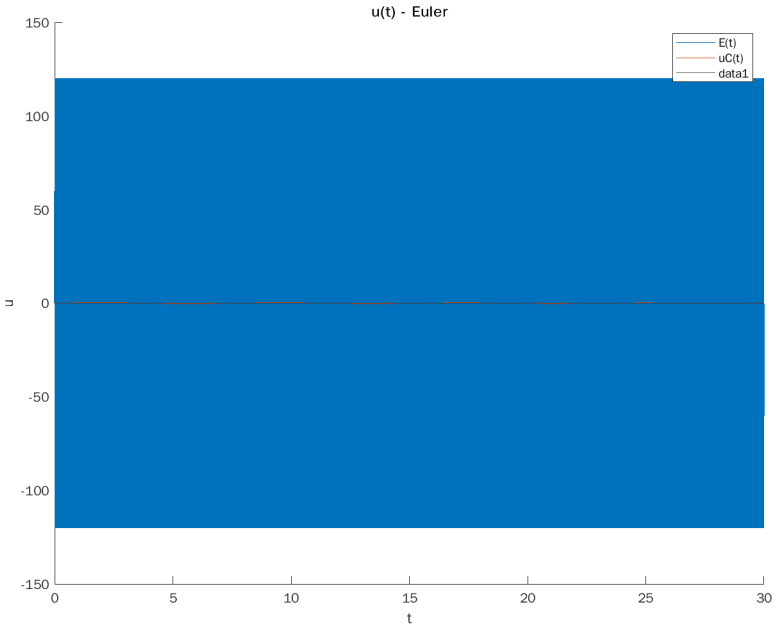


Figure 1.9: u(t) dla wymuszenia $e(t) = 120\sin(2\pi \cdot 50)$

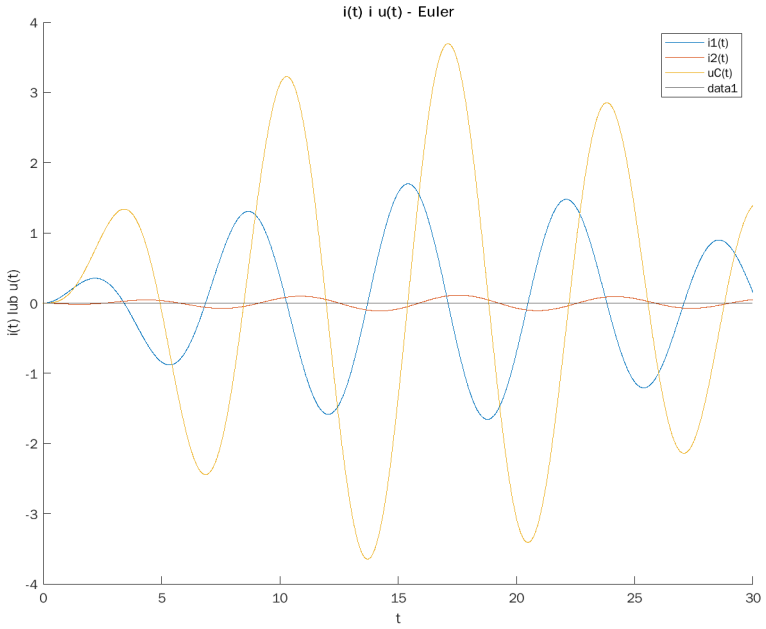


Figure 1.10: przebiegi dla wymuszenia $e(t) = \sin(t)$ - metodą Eulera

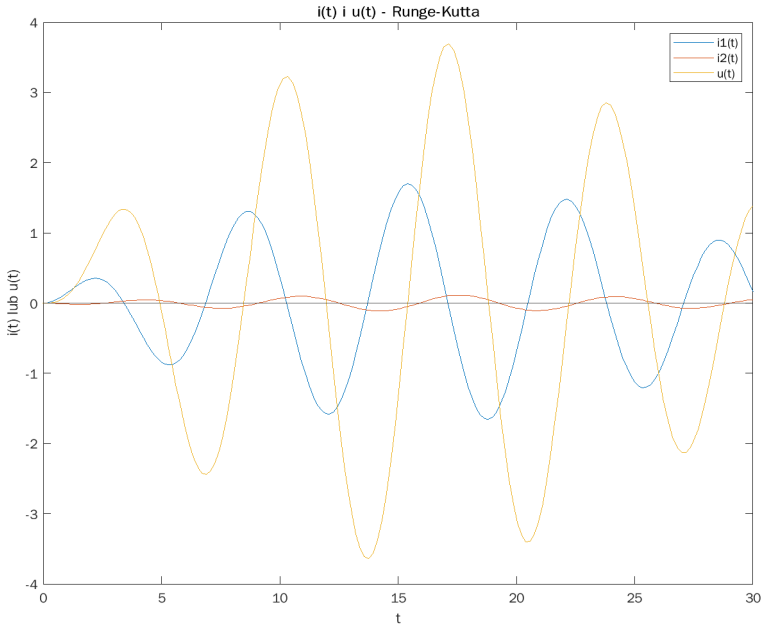


Figure 1.11: przebiegi dla wymuszenia $e(t) = \sin(t)$ - metodą Rungego-Kutty

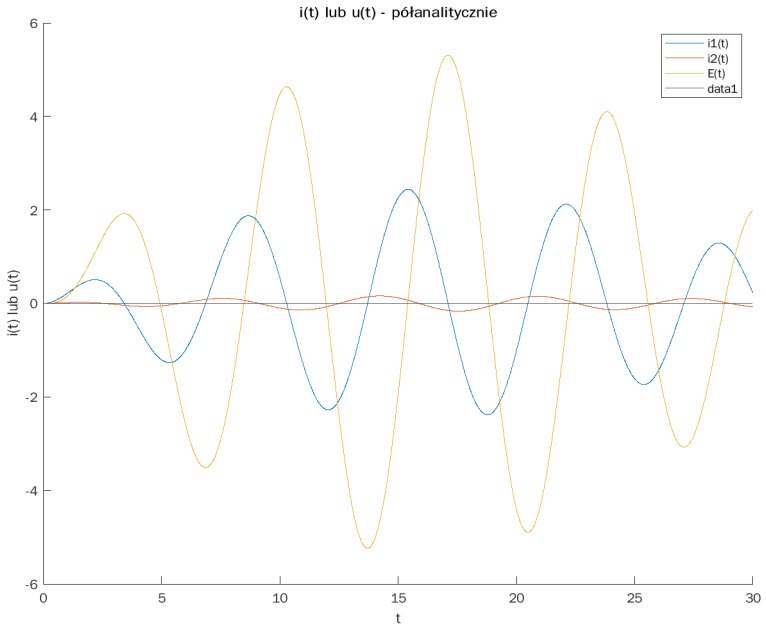


Figure 1.12: przebiegi dla wymuszenia $e(t) = \sin(t)$ - metodą półanalityczną

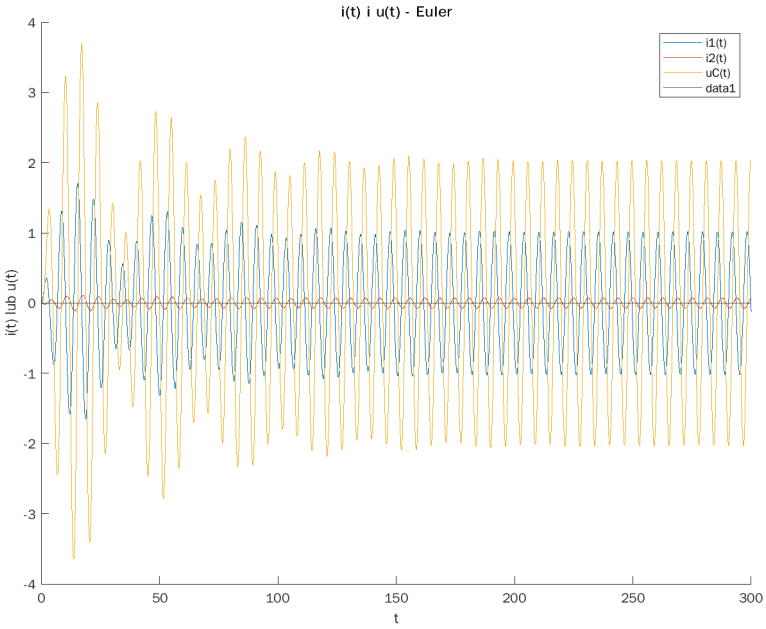


Figure 1.13: przebiegi do 300s dla wymuszenia $e(t) = \sin(t)$ - metodą Eulera

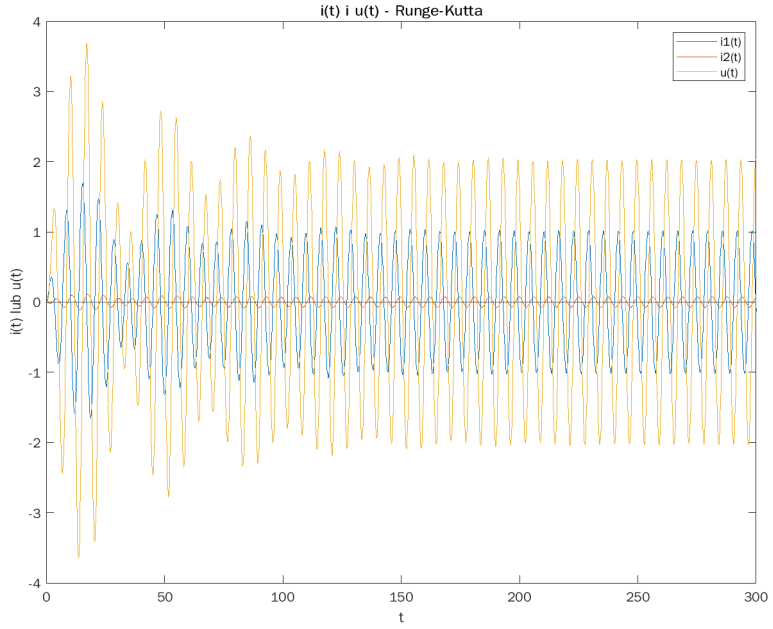


Figure 1.14: przebiegi do 300s dla wymuszenia $e(t) = \sin(t)$ - metodą Rungego-Kutty

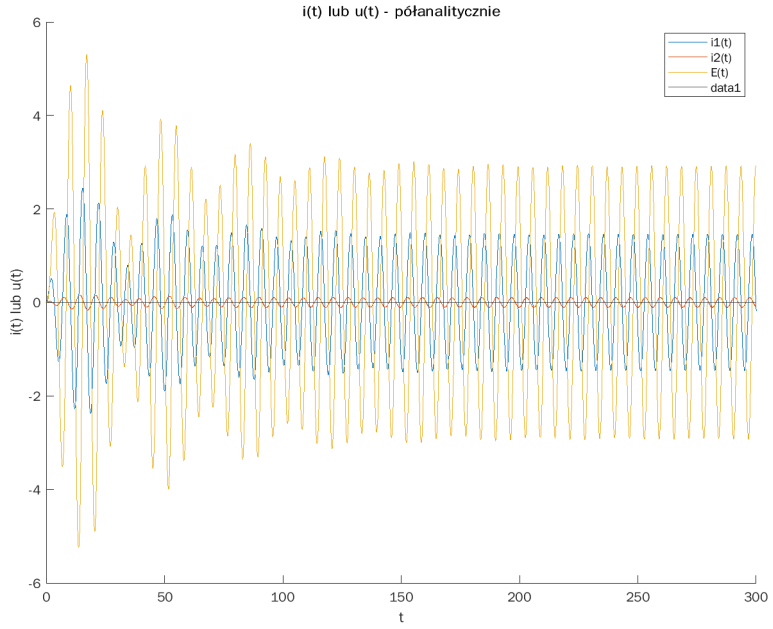


Figure 1.15: przebiegi do 300s dla wymuszenia $e(t) = \sin(t)$ - metodą półanalityczną

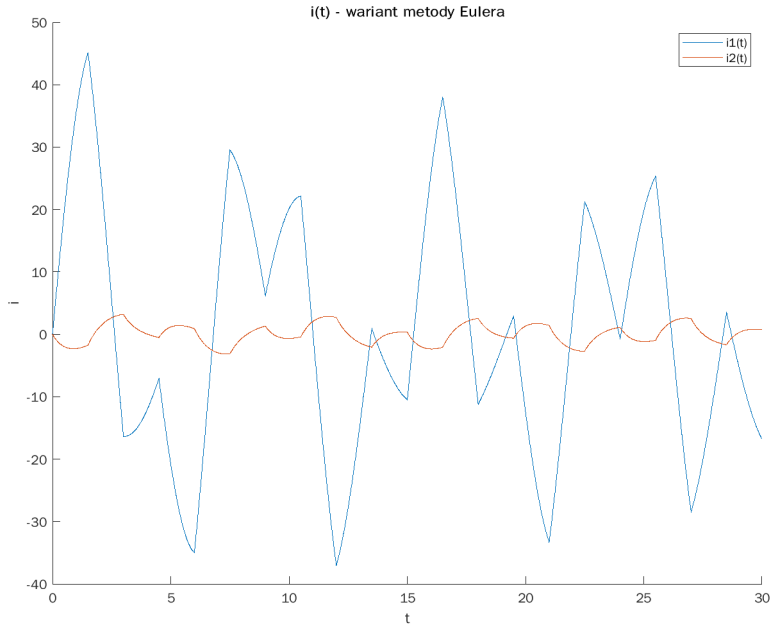


Figure 1.16: i(t) dla wymuszenia pulsacyjnego

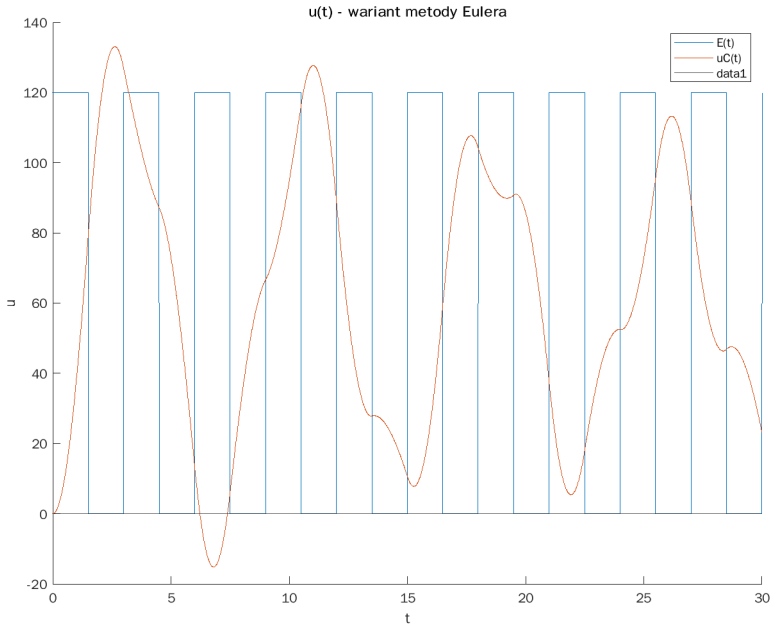


Figure 1.17: u(t) dla wymuszenia pulsacyjnego

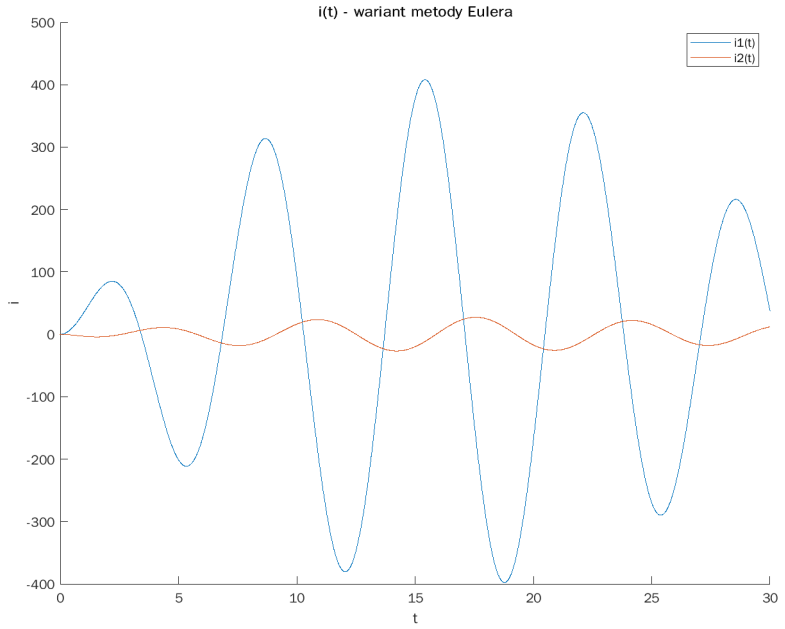


Figure 1.18: i(t) dla wymuszenia $e(t) = 240\sin(t)$

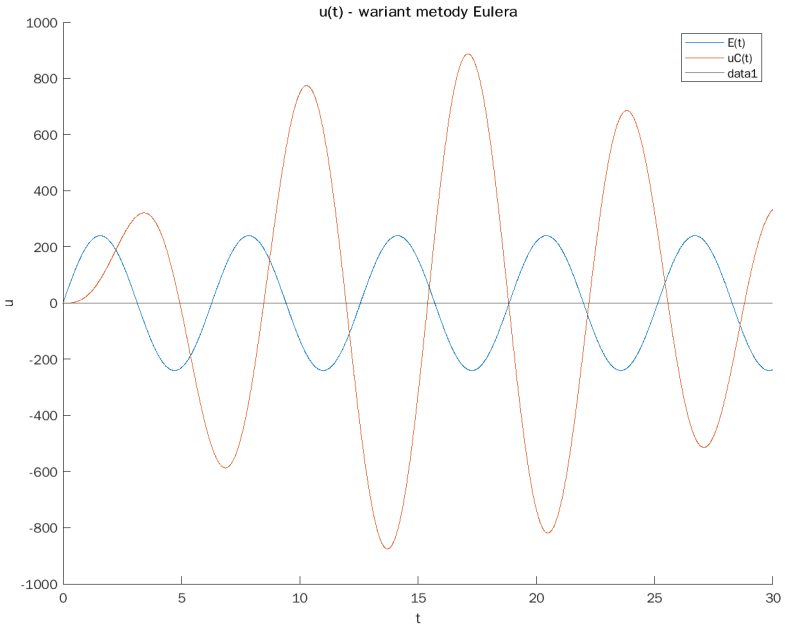


Figure 1.19: u(t) dla wymuszenia $e(t) = 240\sin(t)$

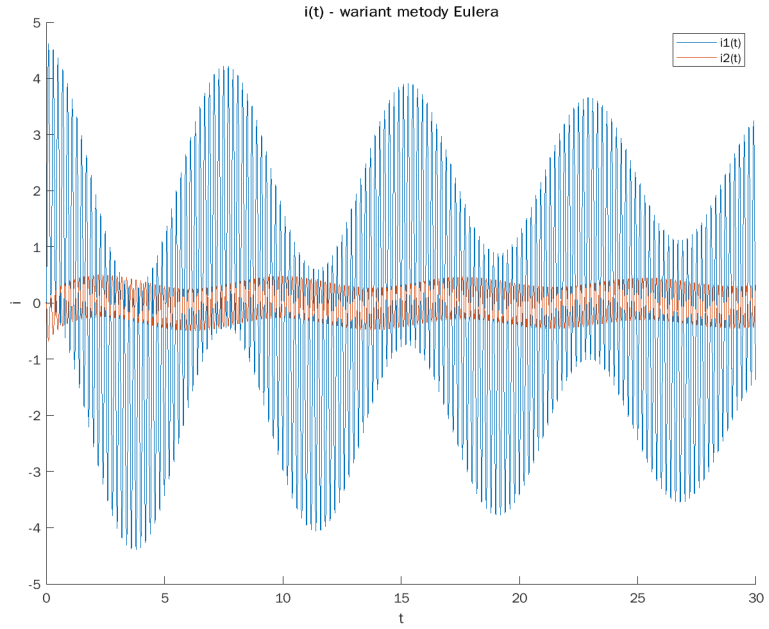


Figure 1.20: i(t) dla wymuszenia $e(t) = 210\sin(2\pi \cdot 5)$

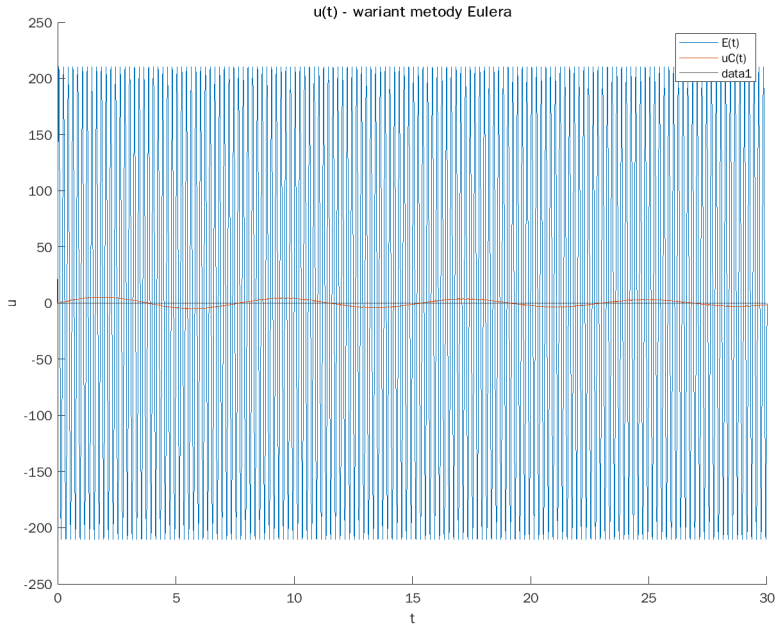


Figure 1.21: u(t) dla wymuszenia $e(t) = 210\sin(2\pi \cdot 5)$

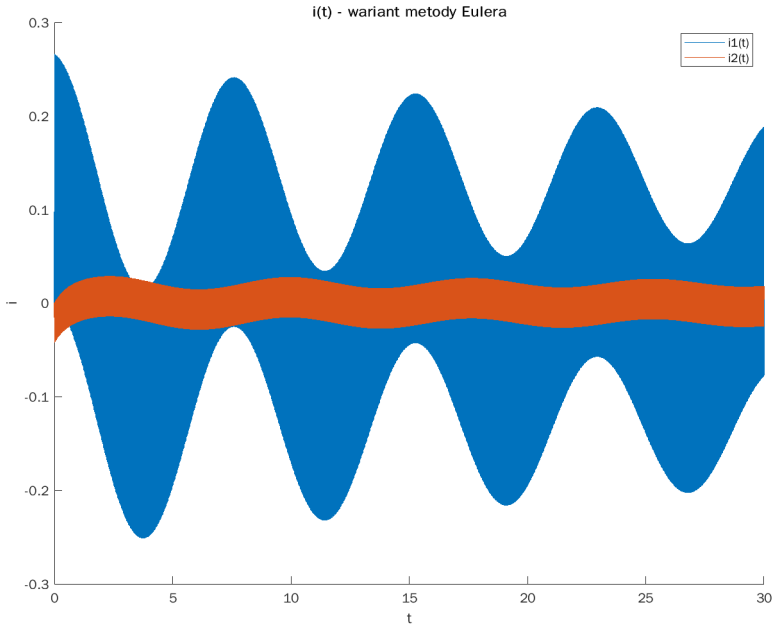


Figure 1.22: i(t) dla wymuszenia $e(t) = 120\sin(2\pi \cdot 50)$

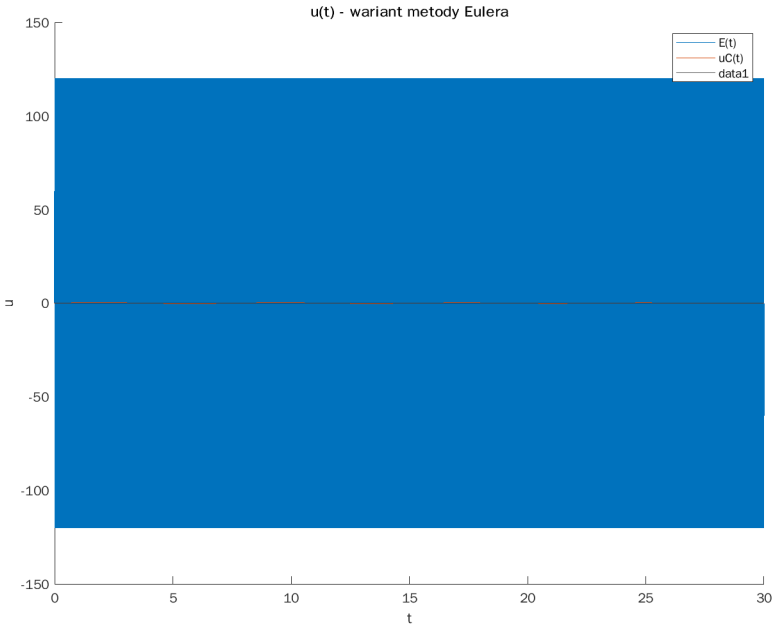


Figure 1.23: u(t) dla wymuszenia $e(t) = 120\sin(2\pi \cdot 50)$

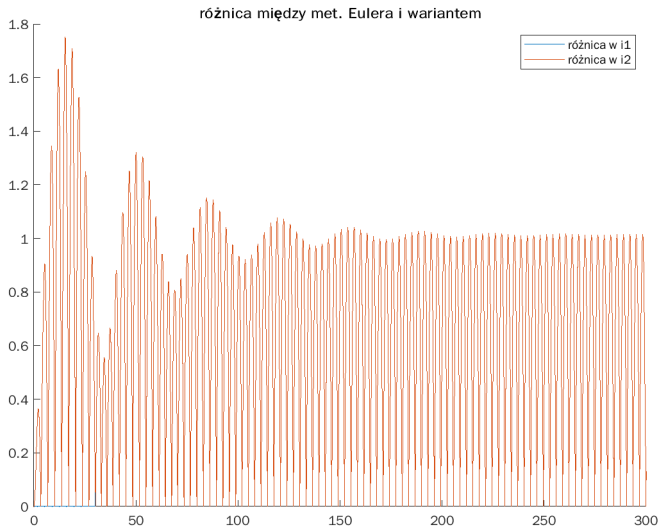


Figure 1.24: różnica między prądami dla dwóch metod Eulera dla wymuszenia $e(t) = \sin(t)$

Chapter 2

Część2

2.1 Opis modelu matematycznego

W tym przypadku oryginalny układ równań opisujący obwód w poprzedniej części jest zmodyfikowany (skomplikowany) o dołożone - czwarte - równanie o poniższej postaci:

$$M(t) = f(u_{L1}(t)) = f(L_1 i_1'(t)) \tag{2.1}$$

Funkcja $M(t)$ nie jest znana. Zamiast niej dysponujemy jedynie kilkoma wartościami M dla odpowiednich napięć u_{L1}. W związku z powyższym nie jest możliwe analityczne rozwiązanie obwodu. *Sensu stricto*, ponieważ nie mam bladego pojęcia, jak wygląda funkcja $M(u_{L1})$ poza podanymi węzłami, przybliżenia numeryczne - oprócz błędu tego co w poprzedniej części, będą posiadały błąd nieznajomości przebiegu $M(u_{L1})$ poza węzłami, które teoretycznie może być dowolne. Celem przybliżenia numerycznego zakładam więc, że funkcja $M(u_{L1})$ "zachowuje się przyzwoicie", tj. intuicyjnie domyślam się, jakie przyjmuje wartości poza węzłami (a być może opierając się na znajomości budowy transformatorów). Istnieje nieskończenie wiele metod pozwalających na oszacowanie $M(u_{L1})$ poza węzłami, obliczenia wykonam dla tych podanych w treści zadania.

2.2 Opis modelu numerycznego

Wiadomo, że $u_{L1}(t)$ przyjmuje wartości zarówno ujemne, jak i dodatnie. Podane wartości dotyczą jedynie dodatnich napięć, domyślam się, że dla ujemnych są identyczne.

Pierwsza z metod to interpolacja wielomianowa, współczynniki wyznaczę metodą Lagrange'a - uzyskam wielomian siódmego stopnia (jest osiem wezłów).

W miejscach na lewo od najmniejszego węzła i na prawo od największego, wartości będą *ekstrapolacją*. Druga metoda to interpolacja funkcjami sklejanymi, zastosuję funkcje sklejane pierwszego stopnia, tj. połączenie węzłów odcinkami. Ekstrapolacja będzie tu po prostu przedłużeniem pierwszego lub ostratniego odcinka. Trzecia i czwarta metoda to aproksymacja wielomianowa trzeciego i piątego stopnia, która daje w wyniku wielomian stopnia mniejszego niż w metodzie pierwszej, lecz łatwiejszy do obliczenia, jednak nieprzechodzący przez wszystkie węzły. Krótko mówiąc, metody aproksymacyjne z założenia będą błędne, lecz łatwiejsze do obliczenia niż interpolacja. Natomiast gdyby przyjąć, że podane wartości funkcji $M(u_{L1})$ w węzłach są przybliżone, nie będzie strasznym błędem znalezienie aproksymacji, która dokładnie przez te węzły nie przechodzi.

2.3 Opis implementacji

Kod implementujące powyższe metody numeryczne jest zaprezentowany poniżej. Jego objaśnienia są zawarte w komentarzach. Aby uruchomić skrypt, należy wpisać *Czesc2(n)*, gdzie n ma wynosić 3- dla pierwszego wymuszenia i 4- dla drugiego wymuszenia, bądź 5- dla metody weryfikacyjnej.

```
function [result] = Czesc2(n) %n=3,4,(5)
    wezly = [20,50,100,150,200,250,280,300];
    wartosci = [0.46,0.64,0.78,0.68,0.44,0.23,0.18,0.18];
    global h;
    h = 0.001;
    global tmax;
    tmax = 30;
    t = 0:h:tmax;
    global sp;
    global figCounter;
    figCounter = 1;

    %Trial(wezly, wartosci); %liczy M(uL) dla roznych metod

    %wielomianowa interp (lagrange)
    sp = 1;
    [uC_lagr, i1_lagr, i2_lagr, E, R1, R2, uL1_lagr, uR2_lagr] = ObliczNumeryczn
    if (n == 5)
        [tM_lagr, yM_lagr] = LiczMacierze(wezly, wartosci, 0);%metod
        a weryfikacyjna dla e=sin(t)
        i1M_lagr = yM_lagr(:,1);
```

```
        i2M_lagr = yM_lagr(:,2);
        uM_lagr = yM_lagr(:,3);
        figure(figCounter);
        figCounter = figCounter + 1;
        hold on;
        plot(tM_lagr, yM_lagr);
        title('mac,lagrange');
        legend('i1','i2','uC');
        hold off;
    end
%     figure(figCounter);
%     figCounter = figCounter+1;
%     hold on;
%     plot(t, i1_lagr);
%     plot(t, i2_lagr);
%     plot(t, uC_lagr);
%     title('interpolacja wielomianowa');
%     hold off;

    %spline interp
    sp = 2;
    [uC_spl, i1_spl, i2_spl, E, R1, R2, uL1_spl, uR2_spl] =
    ObliczNumerycznie(n, wezly, wartosci, 0);
    if (n == 5)
        [tM_spl, yM_spl] = LiczMacierze(wezly, wartosci, 0);
        i1M_spl = yM_spl(:,1);
        i2M_spl = yM_spl(:,2);
        uM_spl = yM_spl(:,3);
        figure(figCounter);
        figCounter = figCounter + 1;
        hold on;
        plot(tM_spl, yM_spl);
        title('mac,spline');
        legend('i1','i2','uC');
        hold off;
    end
%     figure(figCounter);
%     figCounter = figCounter+1;
%     hold on;
%     plot(t, i1_spl);
%     plot(t, i2_spl);
```

```
%     plot(t, uC_spl);
%     title('interpolacja sklejana');
%     hold off;

    %aprox 3
    sp = 3;
    A3 = AproksymacjaStopien(wezly, wartosci, 3);%najpierw współczynniki
    [uC_apr3, i1_apr3, i2_apr3, E, R1, R2, uL1_apr3, uR2_apr3] = ObliczNumeryczn
    if (n == 5)
        [tM_apr3, yM_apr3] = LiczMacierze(wezly, wartosci, A3);
        i1M_apr3 = yM_apr3(:,1);
        i2M_apr3 = yM_apr3(:,2);
        uM_apr3 = yM_apr3(:,3);
        figure(figCounter);
        figCounter = figCounter + 1;
        hold on;
        plot(tM_apr3, yM_apr3);
        title('mac,apr3');
        legend('i1','i2','uC');
        hold off;
    end
%     figure(figCounter);
%     figCounter = figCounter+1;
%     hold on;
%     plot(t, i1_apr3);
%     plot(t, i2_apr3);
%     plot(t, uC_apr3);
%     title('interpolacja aproks. 3 st.');
%     hold off;

    %aprox 5
    sp = 4;
    A5 = AproksymacjaStopien(wezly, wartosci, 5);%najpierw współczynniki
    [uC_apr5, i1_apr5, i2_apr5, E, R1, R2, uL1_apr5, uR2_apr5] = ObliczNumeryczn
    if (n == 5)
        [tM_apr5, yM_apr5] = LiczMacierze(wezly, wartosci, A5);
        i1M_apr5 = yM_apr5(:,1);
        i2M_apr5 = yM_apr5(:,2);
        uM_apr5 = yM_apr5(:,3);
        figure(figCounter);
        figCounter = figCounter + 1;
```

```
        hold on;
        plot(tM_apr5, yM_apr5);
        title('mac,apr5');
        legend('i1','i2','uC');
        hold off;
    end
%     figure(figCounter);
%     figCounter = figCounter+1;
%     hold on;
%     plot(t, i1_apr5);
%     plot(t, i2_apr5);
%     plot(t, uC_apr5);
%     title('interpolacja aproks. 5 st.');
%     hold off;

    % poniższe funkcje rysują wykresy
    RysowanieRoznic(t, i1_lagr, i1_spl, i1_apr3, i1_apr5, 'i1');
    RysowanieRoznic(t, i2_lagr, i2_spl, i2_apr3, i2_apr5, 'i2');
    RysowanieRoznic(t, uC_lagr, uC_spl, uC_apr3, uC_apr5, 'uC');
    RysowanieRoznic(t, uR2_lagr, uR2_spl, uR2_apr3, uR2_apr5, 'uR2');
    RysowanieRoznic(t, uL1_lagr, uL1_spl, uL1_apr3, uL1_apr5, 'uL1');

    result = 'wykonano';
end

function [uC_2, i1_2, i2_2, E_2, r1, r2, uL1_2, uR2_2] =
ObliczNumerycznie(n, wezly, wartosci, A)
    global h;
    global tmax;
    R1 = 0.1;
    R2 = 10;

    C = 0.5;
    L1 = 3;
    L2 = 5;
    M = 0.8;

    di1dt = @(i1,i2,uC,E,mm) ((1/((L1/mm)-(mm/L2)))*(((-R1/mm)*i1)
    +((R2/L2)*i2)-((1/mm)*uC)+((1/mm)*E)));
    di2dt = @(i1,i2,uC,E,mm) ((1/((mm/L1) - (L2/mm)))*(((-R1/L1)*i1)
    +((R2/mm)*i2)-((1/L1)*uC)+((1/L1)*E)));
```

```
    %di1dt = @(i1,i2,uC,E) ((-1/M) * (R2*i2 + L2 * di2dt(i1,i2,uC,E)));
    duCdt = @(i1) ((1/C)*i1);

    t = 0:h:tmax;
    E = zeros(1, length(t));
    i1 = zeros(1, length(t));
    i2 = zeros(1, length(t));
    uC = zeros(1, length(t));
    uL1 = zeros(1, length(t));
    uR2 = zeros(1, length(t));
    i = 1;
    E(1) = fE(n, 0); %funkcja fE liczy E(t) dla roznych (n) podpunktow
     tego zadania (na dole)
    while (t(i) < tmax)
        mm = fM(wezly, wartosci, uL1(i), A);% to daje M
        E(i+1) = fE(n, t(i+1));
        poch = di1dt(i1(i),i2(i),uC(i),E(i),mm);
        i1(i+1) = i1(i) + (h * poch);
        i2(i+1) = i2(i) + (h * di2dt(i1(i),i2(i),uC(i),E(i),mm));
        uC(i+1) = uC(i) + (h * duCdt(i1(i)));
        uR2(i+1) = i2(i+1)*R2;
        uL1(i+1) = abs(L1 * poch);
        i = i + 1;
    end
    uC_2 = uC;
    i1_2 = i1;
    i2_2 = i2;
    E_2 = E;
    r1 = R1;
    r2 = R2;
    uL1_2 = uL1;
    uR2_2 = uR2;
end

function [E] = fE(n, t)%funkcja wyliczająca wymuszenie
    global h;
    switch (n(1))
        case 1
            t2 = (1 / h) * t(1);
            T = (1 / h) * 3;
            div = rem(t2, T);
```

```
            if (div < (T/2))    %dobieram takie h żeby T/2 było całkowite
                E = 120;
            else
                E = 0;
            end
        case 2
            E = 240*sin(t(1));
        case 3
            E = 240*sin(2 * t(1));
        case 4
            E = 120*sin(2 * t(1));
        case 5
            E = sin(t(1));
        otherwise
            E = 1000;
    end
end

function [t, y] = LiczMacierze(wezly, wartosci, A)
    global tmax;
    global h;
    [t,y] = ode45(@(t,y) odefun(t,y,wezly,wartosci, A), [0 tmax], [0 0 0]);
end

function dydt = odefun(t, y, wezly, wartosci, A)
    R1 = 0.1;
    R2 = 10;
    C = 0.5;
    L1 = 3;
    L2 = 5;
    M = 0.8;

    dydt = zeros(3,1);
    dydt(1) = (1/((L1/fM(wezly, wartosci, abs(L1*dydt(1)), A)) -
    (fM(wezly, wartosci, abs(L1*dydt(1)), A)/L2)))*((-R1/fM(wezly,
     wartosci, abs(L1*dydt(1)), A))*y(1) + (R2/L2)*y(2) - (1/fM(
     wezly, wartosci, abs(L1*dydt(1)),A))*y(3) + (1/fM(wezly, wartosci, abs(L1*d
    dydt(2) =  (1/(fM(wezly, wartosci, abs(L1*dydt(1)),A)/L1 -
    L2/fM(wezly, wartosci, abs(L1*dydt(1)),A)))*((-R1/L1)*y(1) +
    (R2/fM(wezly, wartosci, abs(L1*dydt(1)),A))*y(2) - (1/L1)*y(3) +
    (1/L1)*sin(t));
```

```
    dydt(3) = (1/C)*y(1);

end

function [M] = fM(wezly, wartosci, uL, A)%w zależności od sposobu inter/apr odpo
    global sp;
    if (sp == 1)
        M = Lagrange(wezly, wartosci, uL);
    elseif (sp == 2)
        M = Spline(wezly, wartosci, uL);
    elseif (sp == 3 || sp == 4)
        M = LiczAproksymacje(uL, A);
    else
        M = 0.8;
    end
end

function [y] = Lagrange(wezly, wartosci, x)%klasyczna funkcja
licząca współczynniki met.Lagrange'a
    wynik = 0;
    for i=1:length(wezly)
        iloczyn = 1;
        for j=1:length(wezly)
            if (j == i)
                continue;
            end
            iloczyn = iloczyn * ((x-wezly(j)) / (wezly(i)-wezly(j)));
        end
        wynik = wynik + (iloczyn * wartosci(i));
    end
    y = wynik;
end

function [a,b,skr] = ZnajdzPrzedzialSpline(wezly, x)%kwalifikuje uL
do przedziału
    a=1;
    b=1;
    skr=0;
    if (x(1) < wezly(1))
        a = 1;
        b = 2;
```

```
        skr = 0;
        return;
    end
    if (x(1) >= wezly(length(wezly)))
        a = length(wezly);
        b = a;
        skr = 1;
        return;
    end
    for i=1:length(wezly)
        if (x(1) == wezly(i))
            a = i;
            b = i;
            skr = 1;
            return;
        end
        if (x(1) > wezly(i) && x(1) < wezly(i+1))
            a = i;
            b = i + 1;
            skr = 0;
            return;
        end
    end
end

function [y] = Spline(wezly, wartosci, x)
    [a, b, skr] = ZnajdzPrzedzialSpline(wezly, x);%aby wyliczyć
    M trzeba 'zakwalifikować' uL do odpowiedzniego przedziału
    if (skr == 1)
        y = wartosci(a);
        return;
    end
    rozw = zeros(2, 1);
    left = [wezly(a), 1; wezly(b), 1];
    right = [wartosci(a); wartosci(b)];
    rozw = linsolve(left, right);%rozwiązuje prosty układ 2x2
    A = rozw(1);
    B = rozw(2);
    y = A * x(1) + B;
end
```

```
function [A] = AproksymacjaStopien(wezly, wartosci, stopien)
    st = stopien(1);
    M = zeros(length(wezly), st+1);%macierz funkcji psi(wezel)
    for i=1:length(wezly)
        for j=1:(st+1)
            M(i, j) = wezly(i)^(j-1);
        end
    end
    Y = transpose(wartosci);
    A = zeros(st+1, 1);
    left = transpose(M) * M;
    right = transpose(M) * Y;
    A = linsolve(left, right);
end

function [y] = LiczAproksymacje(x, A)%mnozy współczynniki przez
x i sumuje
    y = 0;
    for i=1:length(A)
        y = y + A(i)*x(1)^(i-1);
    end
end

function [result] = Trial(wezly, wartosci)
    global figCounter;
    x = 0:1:400;
    yLagrange = zeros(1, length(x));
    ySpline = zeros(1, length(x));
    yApr3 = zeros(1, length(x));
    yApr5 = zeros(1, length(x));
    A3 = AproksymacjaStopien(wezly, wartosci, 3);
    A5 = AproksymacjaStopien(wezly, wartosci, 5);
    for i=1:length(x)
        yLagrange(i) = Lagrange(wezly, wartosci, x(i));
        ySpline(i) = Spline(wezly, wartosci, x(i));
        yApr3(i) = LiczAproksymacje(x(i), A3);
        yApr5(i) = LiczAproksymacje(x(i), A5);
    end
    figure(figCounter);
    figCounter = figCounter + 1;
    hold on;
```

```
    plot(x, yLagrange);
    plot(x, ySpline);
    plot(x, yApr3);
    plot(x, yApr5);
    title('M(uL)');
    legend('lagr','spl','apr3','apr5');
    hold off;
    result = 'wykonano';
end

function [result] = RysowanieRoznic(t, a1, a2, a3, a4, nazwa)
    global figCounter;
    figure(figCounter);
    figCounter = figCounter + 1;
    hold on;
    plot(t, a1);
    plot(t, a2);
    plot(t, a3);
    plot(t, a4);
    title(nazwa);
    xlabel('t');
    ylabel(nazwa);
    legend('inter. wielom.','inter. sklej.', 'aproks. 3st.',
    'aproks. 5st.');
    result = 'wykonano';
end
```

2.4 Wyniki i ich weryfikacja

Aby dobrze przedstawić wyniki. w pierwszej kolejności przedstawię wykres $M(u_{L1})$ oszacowanej przez wszystkie cztery metody (rysunek 2.1).

Widać od razu, że wszystkie cztery metody dają zasadniczo bardzo zbliżone do siebie wyniki. Różnice dotyczą przedziałów *ekstrapolacji*, tj poniżej i powyżej skrajnych węzłów - w przedziale poniżej tylko dla interpolacji funkcjami sklejanymi, w przedziale powyżej dla każdej z metod. Krótko mówiąc, jeśli mają być jakieś różnice w symulacjach, to będą spowodowane znalezieniem się funkcji $u_{L1}(t)$ w tych (skrajnych) przedziałach.

Zatem przedstawiam wykres funkcji $|u_{L1}(t)|$ dla obu wymuszeń w tej części projektu podanych w treści zadania - rysunek 2.2 i 2.3.

Widać, że dla pierwszego wymuszenia u_{L1} dochodzi do 380V, podczas

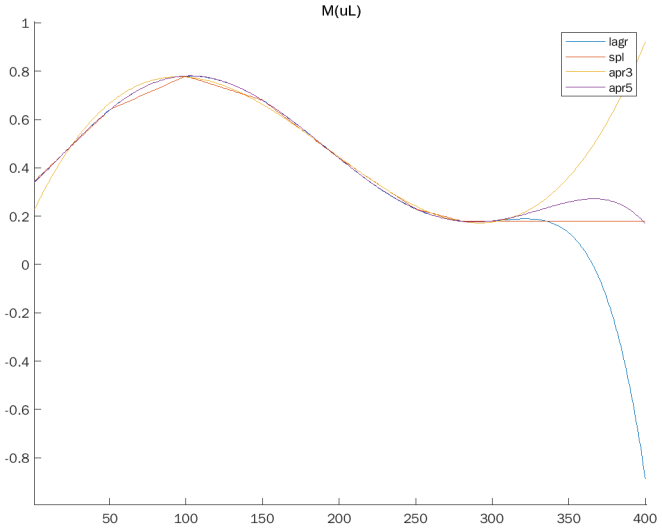


Figure 2.1: przybliżenie $M(u_{L2})$ dla różnych metod interpolacyjnych i aproksymacyjnych

gdy dla drugiego wymuszenia u_{L1} nie przekracza 200V. Ponieważ szacunki $M(u_{L1})$ zaczynają się "rozjeżdżać" dla u_{L1} przekraczającego 300V, należy oczekiwać, że w drugim wymuszeniu przebiegi prądów i napięć będą bardziej zbliżone do siebie niż w pierwszym.

Przebiegi prądów i napięć dla wymuszenia $e(t) = 240\sin(2t)$ są przedstawione na rysunkach 2.4, 2.5, 2.6 i 2.7.

Przebiegi prądów i napięć dla wymuszenia $e(t) = 120\sin(2t)$ są przedstawione na rysunkach 2.8, 2.9, 2.10 i 2.11.

Porównanie tych rysunków potwierdza postawioną chwilę temu tezę. Dla wymuszenia z o połowę mniejszą amplitudą i mniejszą amplitudą napięcia na cewce różnice są mniejsze dzięki "nierozjeżdżaniu się" funkcji szacujących sprzężenie M. Co więcej, widać też, że niezależnie od wymuszenia, oszacowanie sprzężenia metodą funkcji sklejanych daje troszkę inne wyniki od pozostałych funkcji. Jest to prawdopodobnie spowodowane "odejściem" tej funkcji szacującej od pozostałych dla przedziału poniżej pierwszego węzła.

A teraz weryfikacja wyników - analizując powyższy obwód metodą Rungego-Kutty. Jak wspomniałem wcześniej, wyliczenie (pół)analityczne tego obwodu wobec braku znajomości $M(t)$ jest niemożliwe. Weryfikacja - jak w poprzedniej części - ma miejsce dla wymuszenia $e(t) = \sin(t)$.

Przebiegi i_1, i_2, u_c (pominąłem u_R z powodu banalnie prostej zależności od i_2) dla każdej z metod liczonej dla metody prostej Eulera (trzy przebiegi na każdżą metodę interpolacyjną/aproksymacyjną), jak również (dla porównania) analogiczne przebiegi dla każdej z metod liczonej dla metody

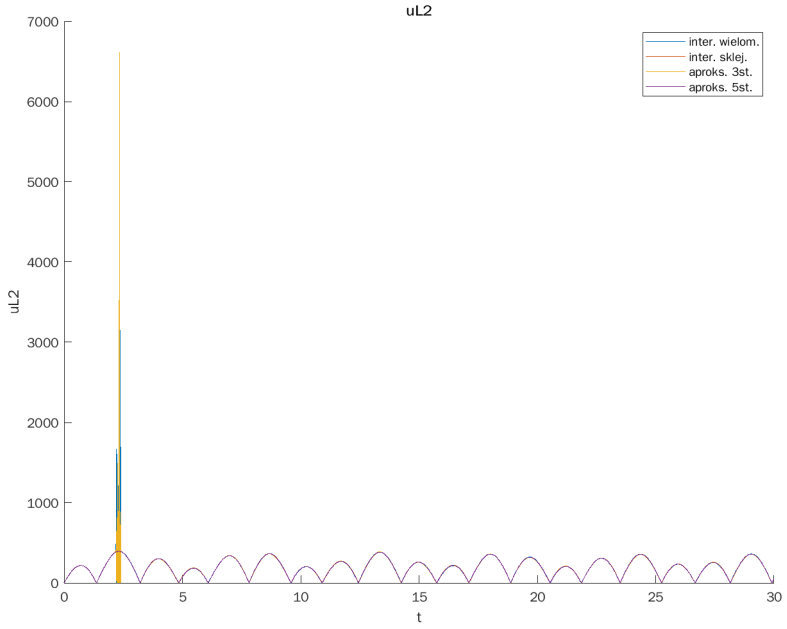


Figure 2.2: $u_{L1}(t)$ dla wymuszenia $e(t) = 240\sin(2t)$

Rungego-Kutty (tytuł z przedrostkiem 'mac') - celem weryfikacji - są przedstawione na rysunkach 2.12, 2.13, 2.14, 2.15, 2.16, 2.17, 2.18, 2.19.

Brak jest istotnych różnic między tymi wszystkimi przebiegami, co pozytywnie weryfikuje poprzednie wyliczenia.

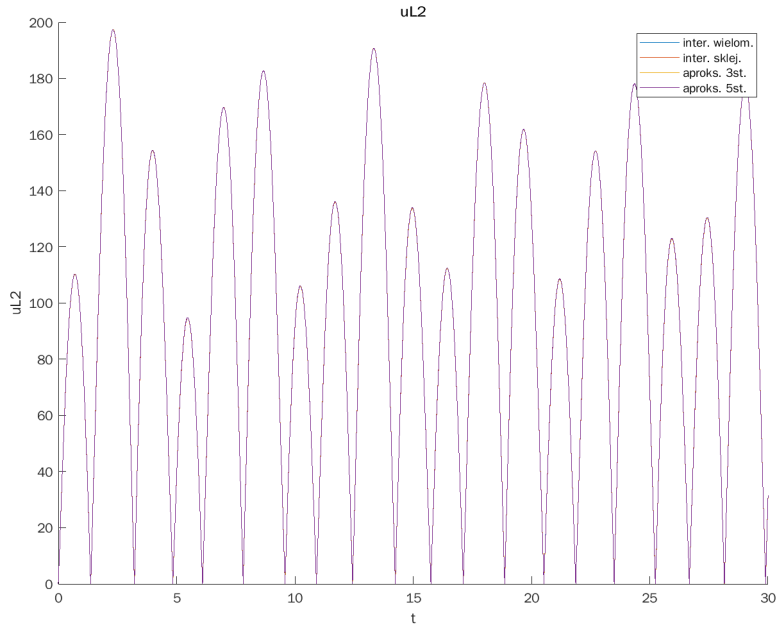


Figure 2.3: $u_{L1}(t)$ dla wymuszenia $e(t) = 120\sin(2t)$

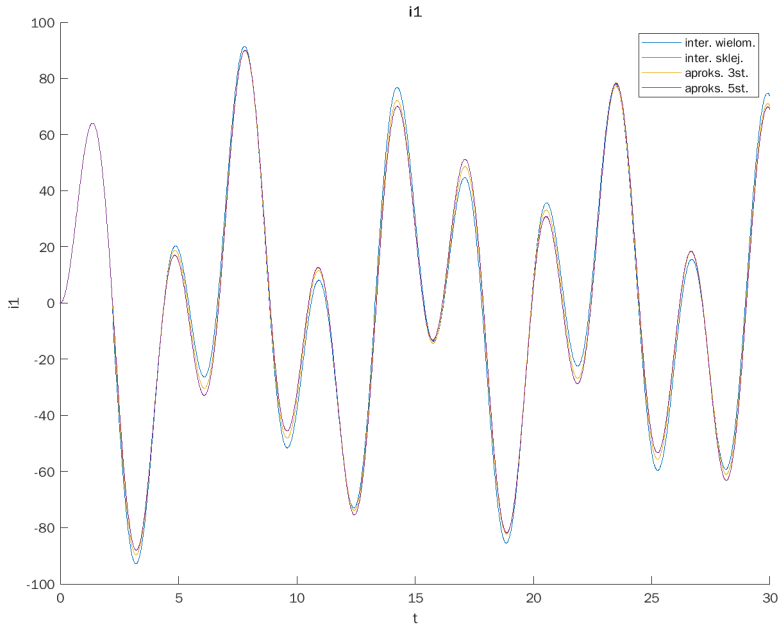


Figure 2.4: $i_1(t)$ dla wymuszenia $e(t) = 240\sin(2t)$

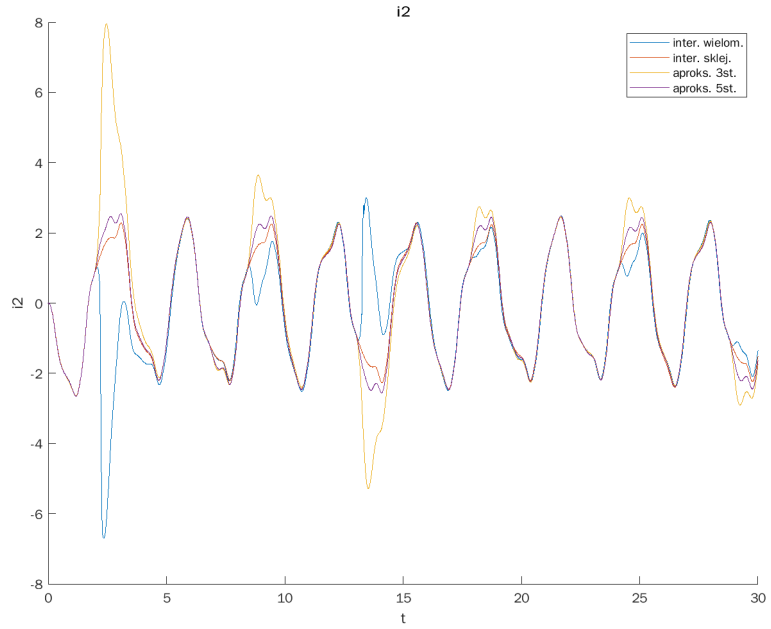


Figure 2.5: $i_2(t)$ dla wymuszenia $e(t) = 240\sin(2t)$

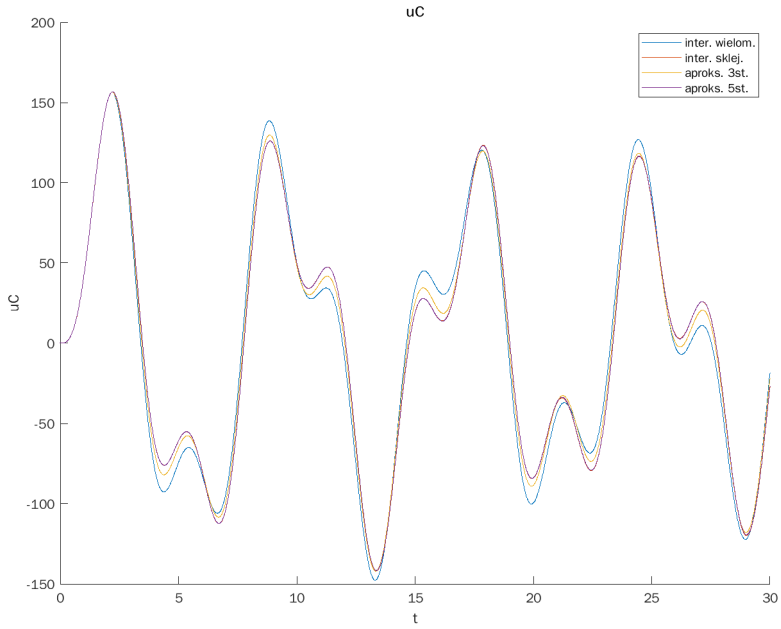


Figure 2.6: $u_C(t)$ dla wymuszenia $e(t) = 240\sin(2t)$

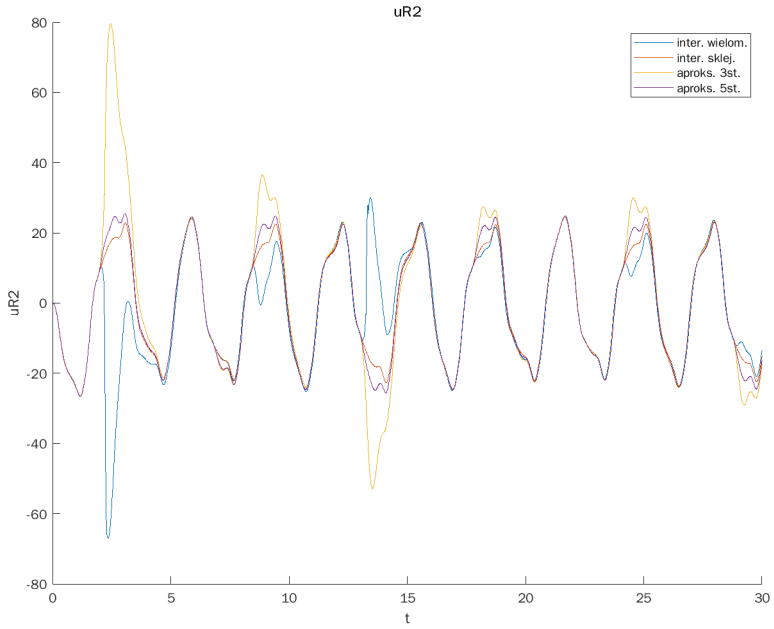


Figure 2.7: $u_{R2}(t)$ dla wymuszenia $e(t) = 240\sin(2t)$

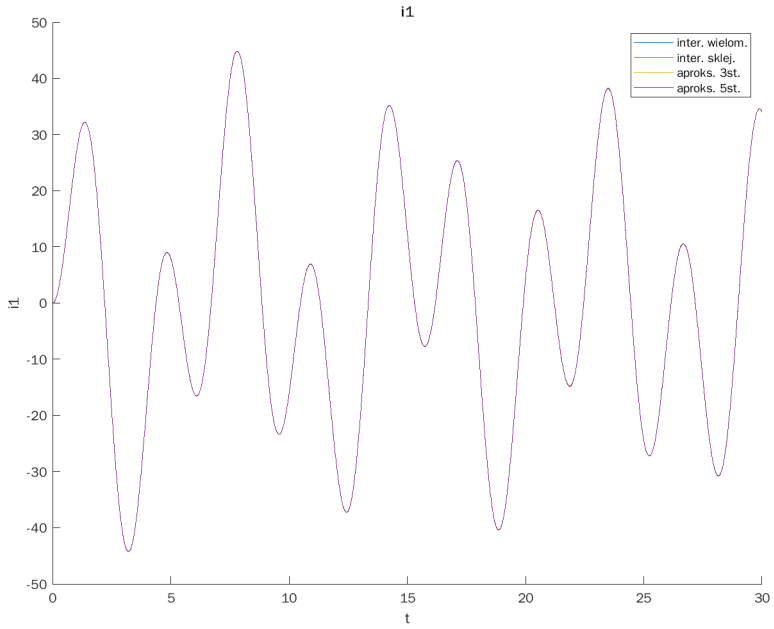


Figure 2.8: $i_1(t)$ dla wymuszenia $e(t) = 120\sin(2t)$

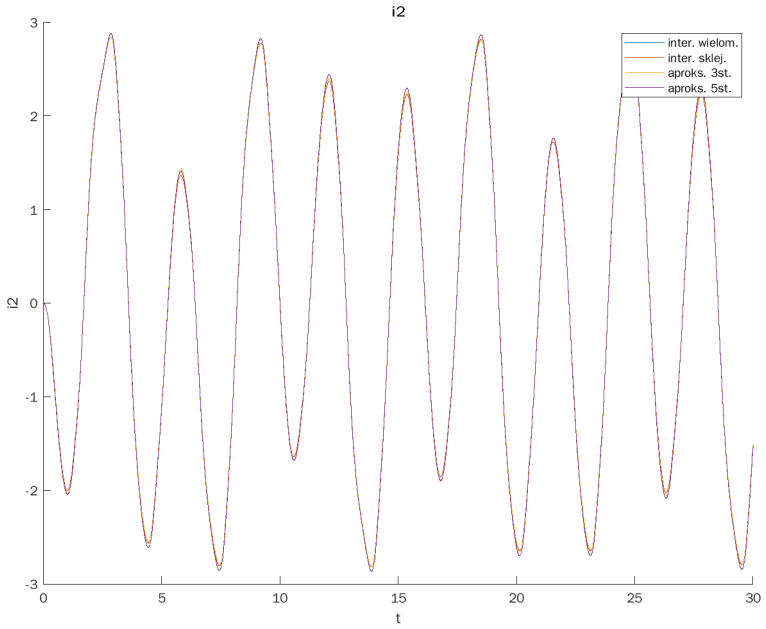


Figure 2.9: $i_2(t)$ dla wymuszenia $e(t) = 120\sin(2t)$

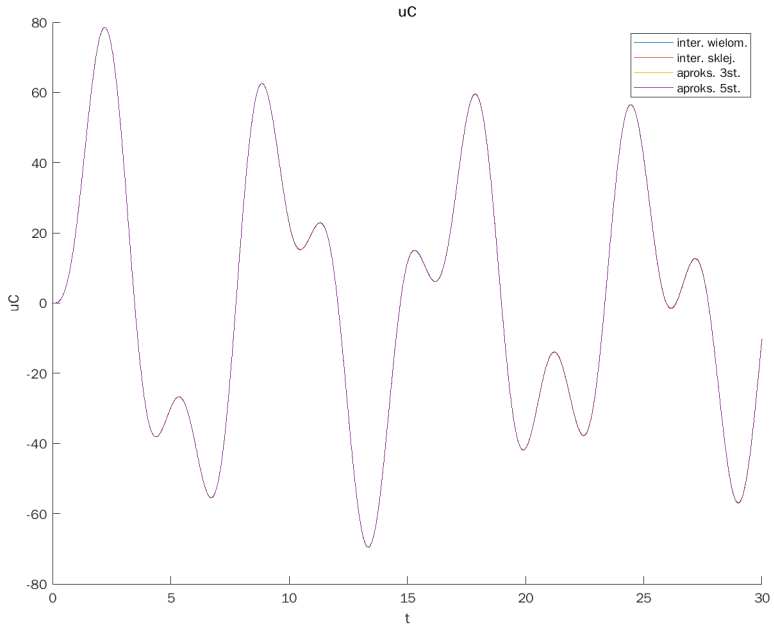


Figure 2.10: $u_C(t)$ dla wymuszenia $e(t) = 120\sin(2t)$

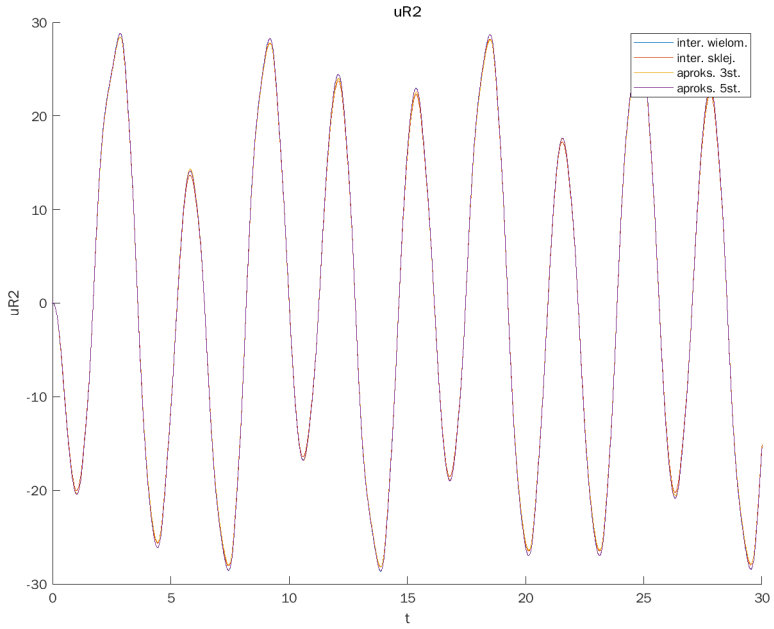


Figure 2.11: $u_{R2}(t)$ dla wymuszenia $e(t) = 120\sin(2t)$

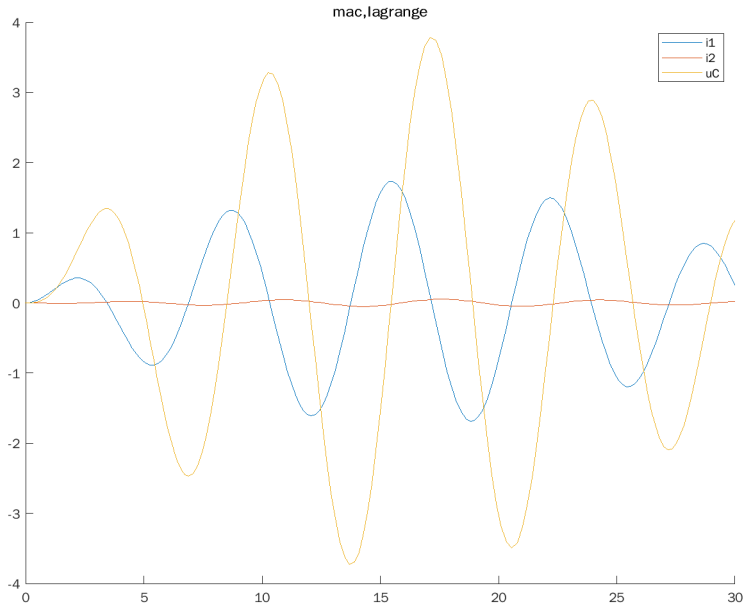


Figure 2.12: $e(t) = \sin(t)$

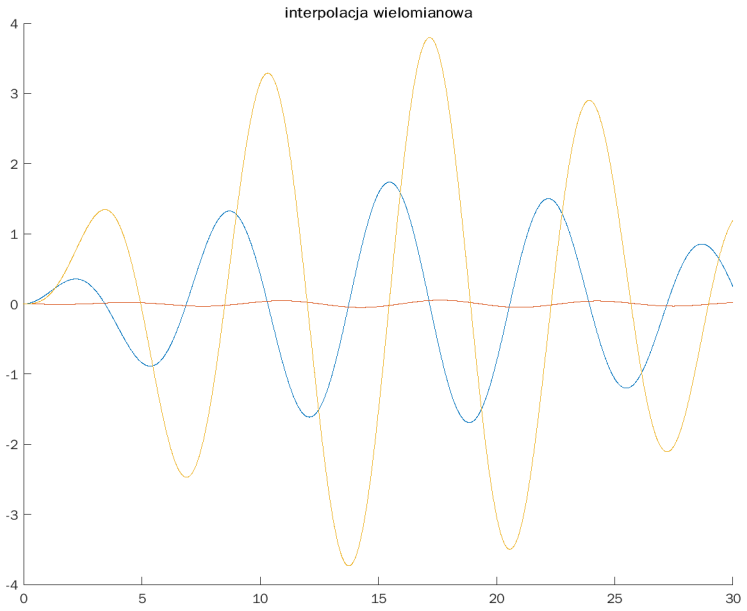


Figure 2.13: $e(t) = \sin(t)$

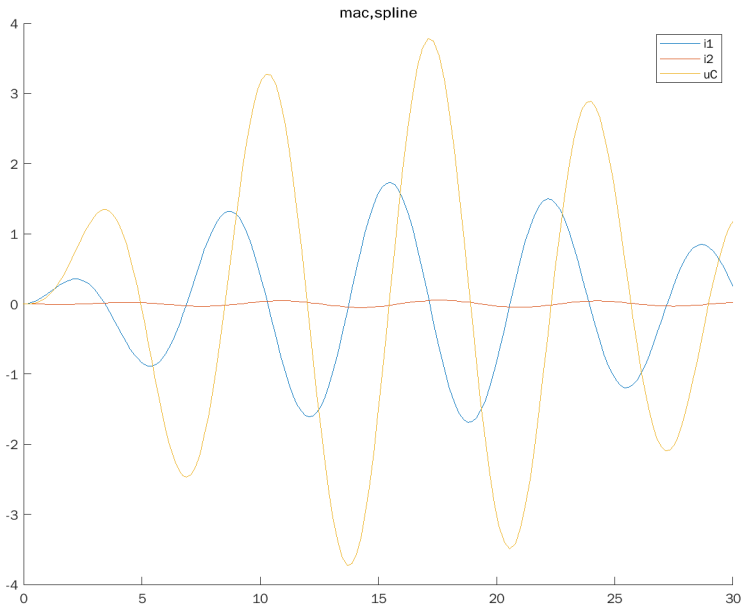


Figure 2.14: $e(t) = \sin(t)$

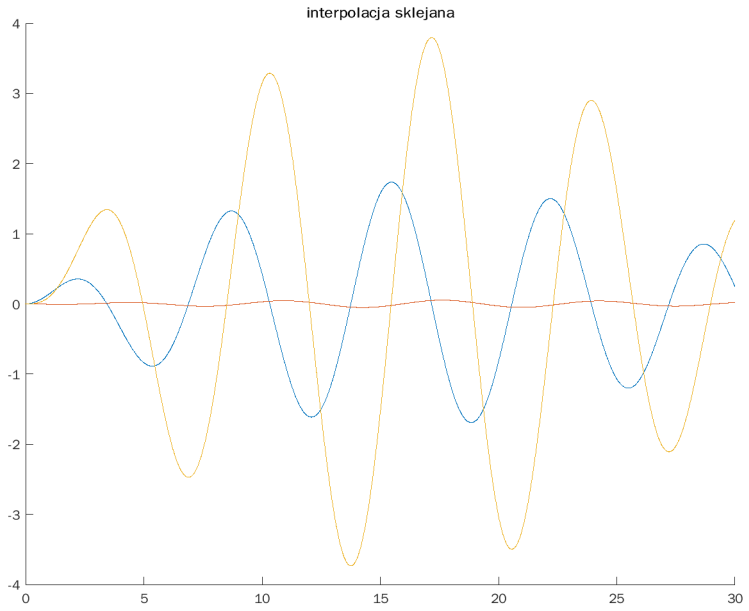


Figure 2.15: $e(t) = \sin(t)$

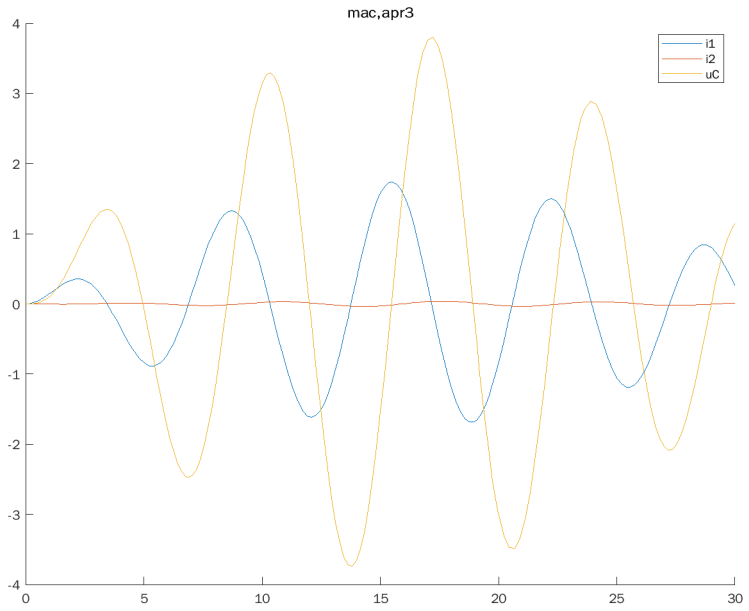


Figure 2.16: $e(t) = \sin(t)$

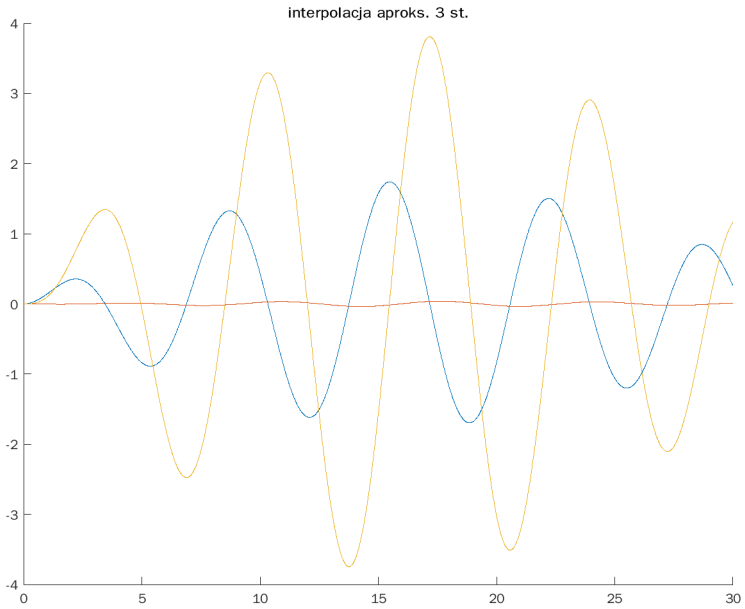


Figure 2.17: $e(t) = \sin(t)$

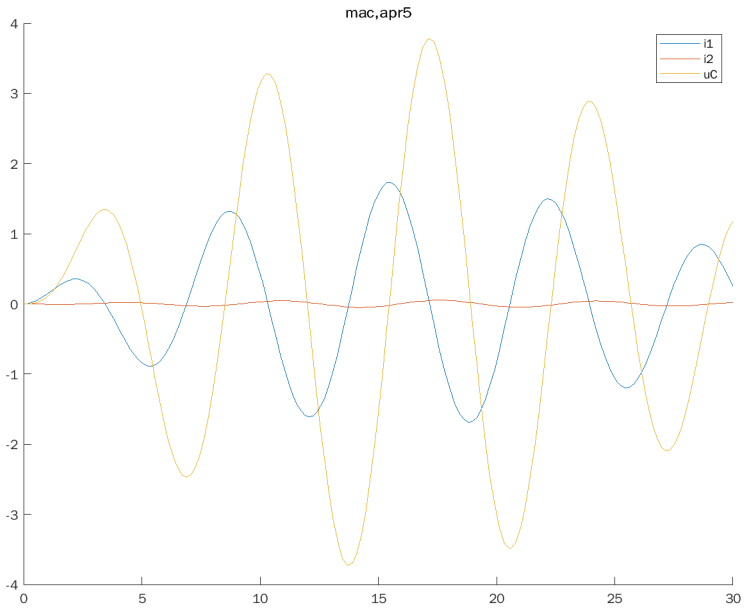


Figure 2.18: $e(t) = \sin(t)$

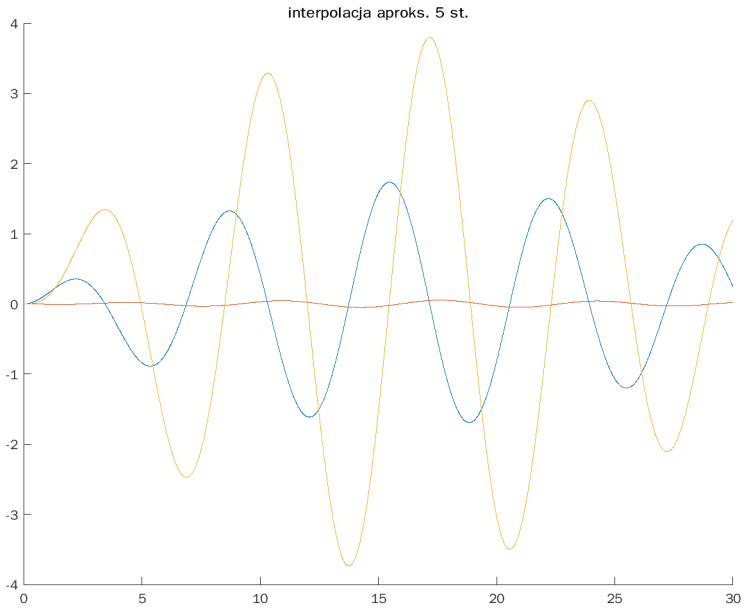


Figure 2.19: $e(t) = \sin(t)$

Chapter 3

Część 3

3.1 Opis modelu matematycznego

Ciepło wydzielone w obwodzie na opornikach w ciągu 30s wynosi jest całką taką jak ta pokazana w treści zadania. Moc czynna jest to wydzielone ciepło podzielone przez 30s (innymi słowy - śrenie ciepło wydzielone w każdej sekundzie symulacji przez pierwsze 30s). Ponieważ model półanalityczny wyprowadzony w Części 1 wylicza jawnie funkcje $i_1(t)$ oraz $i_2(t)$, jestem tutaj w stanie przeprowadzić weryfikację na kilka sposobów - nie tylko wyliczając wartość z modelu półanalitycznego, lecz również testując metodę liczenia całki.

3.2 Opis modelu numerycznego

Sposób szacowania całki metodą prostokątów i parabol jest szczegółowo przedstawiony w treści zadania i w podręczniku i nie będę go tutaj powtarzał. Dodam, że celem weryfikacji tych sposobów zamierzam zastosować sposób szacowania całki za pomocą dynamicznie dopasowywanej kwadratury - metoda zaimplementowana w funkcji `integral()` obecnej w MATLABIE.

3.3 Opis implementacji

Kod implementujący metodę jest przedstawiony poniżej. Aby uruchomić skrypt, należy wpisać *Czesc3(n)*, gdzie n wynosi 1- dla wymuszenia 0/120V, 2- dla podanego wymuszenia z częstotliwością 1, 3- dla wymuszenia z czestotliwością 5, 4- dla wymuszenia z częstotliwością 50, 5- dla wymuszenia $e(t) = \sin(t)$, 6- dla wymuszenia stałego 1V. Wyjaśnienia kodu są podane w formie

komentarzy, jak poprzednio.

```
function [result] = Czesc3(n)
    global sp;%dla zmiennego sprzeżenia.
    sp = 1;
    global h;
    h = 0.00001;
    global Ctmax;
    tmax = 30;
    t = 0:h:tmax;
    wezly = [20,50,100,150,200,250,280,300];%dla zmiennego sprzężenia
    wartosci = [0.46,0.64,0.78,0.68,0.44,0.23,0.18,0.18];
    %start liczenia pradow
    [uC, i1, i2, E, R1, R2] = ObliczNumerycznie(n);
    %[uC, i1, i2, E, R1, R2] = ObliczNumerycznie(n, wezly, wartosci, 0);
    [uCAnal, i1Anal, i2Anal, energia] = ObliczAnal(n);
    [tM, yM] = LiczMacierze(n);
    i1M = yM(:,1);
    i2M = yM(:,2);
    %koniec liczenia pradow

    %całka - prostokąt - półanalitycznie i numerycznie
    calkaProstokat1 = 0;
    calkaProstokat2 = 0;
    calkaProstokat = 0;
    calkaParabola1 = 0;
    calkaParabola2 = 0;
    calkaParabola = 0;
    calkaProstokatAnal1 = 0;
    calkaProstokatAnal2 = 0;
    calkaProstokatAnal = 0;
    for i=1:length(t)
        calkaProstokat1 = calkaProstokat1 + h*(i1(i)^2);
        calkaProstokat2 = calkaProstokat2 + h*(i2(i)^2);
        calkaProstokatAnal1 = calkaProstokatAnal1 + h*(i1Anal(i)^2);
        calkaProstokatAnal2 = calkaProstokatAnal2 + h*(i2Anal(i)^2);
    end
    calkaProstokat1 = calkaProstokat1 * R1;
    calkaProstokat2 = calkaProstokat2 * R2;
    calkaProstokat = calkaProstokat1 + calkaProstokat2;
    calkaProstokatAnal1 = calkaProstokatAnal1 * R1;
```

```
    calkaProstokatAnal2 = calkaProstokatAnal2 * R2;
    calkaProstokatAnal = calkaProstokatAnal1 + calkaProstokatAnal2;

    %całka - parabola
    i = 1;
    while (i < length(t) - 1)
        calkaParabola1 = calkaParabola1 + (h/3)*((i1(i)^2) + 4*(i1(i+1)^2)
         + (i1(i+2)^2));
        calkaParabola2 = calkaParabola2 + (h/3)*((i2(i)^2) + 4*(i2(i+1)^2)
        + (i2(i+2)^2));
        i = i + 2;
    end
    calkaParabola1 = calkaParabola1 * R1;
    calkaParabola2 = calkaParabola2 * R2;
    calkaParabola = calkaParabola1 + calkaParabola2;

    %całka - metodą weryfikacyjną
    calkaProstMac1 = 0;
    calkaProstMac2 = 0;
    calkaProstMac = 0;
    for i=1:(length(tM)-1)
        calkaProstMac1 = calkaProstMac1 + ((tM(i+1)-tM(i))*i1M(i)^2);
        calkaProstMac2 = calkaProstMac2 + ((tM(i+1)-tM(i))*i2M(i)^2);
    end
    calkaProstMac1 = calkaProstMac1 * R1;
    calkaProstMac2 = calkaProstMac2 * R2;
    calkaProstMac = calkaProstMac1 + calkaProstMac2;

    mocProstokat = calkaProstokat / tmax;
    mocParabola = calkaParabola / tmax;
    mocProstokatAnal = calkaProstokatAnal / tmax;
    mocIntegralAnal = energia / tmax;
    mocMacierze = calkaProstMac / tmax;

    result = sprintf('Energia[J]:\nprostokaty: %.3f\nparabole:
    %.3f\nnumerycznie: %.3f\nintegral(): %.3f\nmacierze: %.3f\
    nMoc[W]:\nprostokaty: %.3f\nparabole: %.3f\nnumerycznie: %.3f\nintegral(): %
    ProstokatAnal, energia, calkaProstMac, mocProstokat, mo
    cParabola, mocProstokatAnal, mocIntegralAnal, mocM
    acierze);
end
```

```
% function [uC_2, i1_2, i2_2, E_2, r1, r2] = ObliczNumerycznie(n, wezly,
% wartosci, A)%funkcja dla zmiennego sprzężenia
%     global h;
%     global tmax;
%     R1 = 0.1;
%     R2 = 10;
%
%     C = 0.5;
%     L1 = 3;
%     L2 = 5;
%     M = 0.8;
%
%     di1dt = @(i1,i2,uC,E,mm) ((1/((L1/mm)-(mm/L2)))*(((-R1/mm)*i1)
+((R2/L2)*i2)-((1/mm)*uC)+((1/mm)*E)));
%     di2dt = @(i1,i2,uC,E,mm) ((1/((mm/L1) - (L2/mm)))*(((-R1/L1)*i1)
+((R2/mm)*i2)-((1/L1)*uC)+((1/L1)*E)));
%     %di1dt = @(i1,i2,uC,E) ((-1/M) * (R2*i2 + L2 *
 di2dt(i1,i2,uC,E)));
%     duCdt = @(i1) ((1/C)*i1);
%
%     t = 0:h:tmax;
%     E = zeros(1, length(t));
%     i1 = zeros(1, length(t));
%     i2 = zeros(1, length(t));
%     uC = zeros(1, length(t));
%     uL1 = zeros(1, length(t));
%     uR2 = zeros(1, length(t));
%     i = 1;
%     E(1) = fE(n, 0); %funkcja fE liczy E(t) dla roznych
 (n) podpunktow tego zadania (na dole)
%     while (t(i) < tmax)
%         mm = fM(wezly, wartosci, uL1(i), A);% to daje M
%         E(i+1) = fE(n, t(i+1));
%         poch = di1dt(i1(i),i2(i),uC(i),E(i),mm);
%         i1(i+1) = i1(i) + (h * poch);
%         i2(i+1) = i2(i) + (h * di2dt(i1(i),i2(i),uC(i),E(i),mm));
%         uC(i+1) = uC(i) + (h * duCdt(i1(i)));
%         uR2(i+1) = i2(i+1)*R2;
%         uL1(i+1) = abs(L1 * poch);
%         i = i + 1;
```

```
%     end
%     uC_2 = uC;
%     i1_2 = i1;
%     i2_2 = i2;
%     E_2 = E;
%     r1 = R1;
%     r2 = R2;
% end
%
% function [M] = fM(wezly, wartosci, uL, A)%w zmienne M
%     global sp;
%     if (sp == 1)
%         M = Lagrange(wezly, wartosci, uL);
%     else
%         M = 0.8;
%     end
% end
%
% function [y] = Lagrange(wezly, wartosci, x)%klasyczna funkcja
 licząca współczynniki met.Lagrange'a
%     wynik = 0;
%     for i=1:length(wezly)
%         iloczyn = 1;
%         for j=1:length(wezly)
%             if (j == i)
%                 continue;
%             end
%             iloczyn = iloczyn * ((x-wezly(j)) / (wezly(i)-wezly(j)));
%         end
%         wynik = wynik + (iloczyn * wartosci(i));
%     end
%     y = wynik;
% end

function [uC_2, i1_2, i2_2, E_2, r1, r2] = ObliczNumerycznie(n)
    global h;
    global tmax;
    R1 = 0.1;
    R2 = 10;
    C = 0.5;
    L1 = 3;
```

```
    L2 = 5;
    M = 0.8;
    di1dt = @(i1,i2,uC,E) ((1/((L1/M)-(M/L2)))*(((-R1/M)*i1)
    +((R2/L2)*i2)-
    ((1/M)*uC)+((1/M)*E)));
    di2dt = @(i1,i2,uC,E) ((1/((M/L1) - (L2/M)))*(((-R1/L1)*i1)+
    ((R2/M)*i2)-
    ((1/L1)*uC)+((1/L1)*E)));
    %di1dt = @(i1,i2,uC,E) ((-1/M) * (R2*i2 + L2 *
     di2dt(i1,i2,uC,E)));
    duCdt = @(i1) ((1/C)*i1);

    t = 0:h:tmax;
    E = zeros(1, length(t));
    i1 = zeros(1, length(t));
    i2 = zeros(1, length(t));
    uC = zeros(1, length(t));
    i = 1;
    E(1) = fE(n, 0); %funkcja fE liczy E(t) dla roznych (n) podpunktow
    tego zadania (na dole)
    while (t(i) < tmax)
        t(i+1) = t(i) + h;
        E(i+1) = fE(n, t(i+1));%E(t) rysuje sie dobrze
        i1(i+1) = i1(i) + (h * di1dt(i1(i),i2(i),uC(i),E(i)));
        i2(i+1) = i2(i) + (h * di2dt(i1(i),i2(i),uC(i),E(i)));
        uC(i+1) = uC(i) + (h * duCdt(i1(i)));
        i = i + 1;
    end
    uC_2 = uC;
    i1_2 = i1;
    i2_2 = i2;
    E_2 = E;
    r1 = R1;
    r2 = R2;
end

function [t, y] = LiczMacierze(n)
    global tmax;
    global h;
    if (n(1) == 5)
        [t,y] = ode45(@odefun, [0 tmax], [0 0 0]);
```

```
    else
        t = [1 1];
        y = [1 1];
    end
end

function dydt = odefun(t, y)
    R1 = 0.1;
    R2 = 10;
    C = 0.5;
    L1 = 3;
    L2 = 5;
    M = 0.8;
    dydt = [(1/(L1/M - M/L2))*((-R1/M)*y(1) + (R2/L2)*y(2) -
     (1/M)*y(3) + (1/M)*sin(t));
        (1/(M/L1 - L2/M))*((-R1/L1)*y(1) + (R2/M)*y(2) -
        (1/L1)*y(3) + (1/L1)*sin(t));

        (1/C)*y(1)];
end

function [u, i1, i2, energia] = ObliczAnal(n)
    global h;
    global tmax;
    t = 0:h:tmax;
    energia = 0;
    R1 = 0.1;
    R2 = 10;
    i1Anal = zeros(1,length(t));
    i2Anal = zeros(1,length(t));
    uAnal = zeros(1,length(t));
    for i=1:length(t)
        if (n == 4)
        %dla e=120sin(2pi50t)
        uAnal(i) = -0.00605859*exp(-2.0779*t(i)) + 0.449743*exp(-0.023*t(i))*sin
         - 3.63769*10^-19 * exp(-2.0779*t(i))) - 0.00120892*sin(314.759*t(i))
          + 0.00605907*exp(-0.023*t(i)) * cos(0.8184*t(i)) -
          4.75862*10^-7 * cos(314.759*t(i));
        i1Anal(i) = 0.00645155*exp(-2.0799*t(i)) - 0.00770643*exp(-0.023*t(i))*s
        + 0.183808*exp(-0.023*t(i))*cos(0.8184*t(i)) - 0.19026*cos(314.759*t(i))
         + 0;
```

```
    i2Anal(i) = 0.0268642*exp(-2.0779*t(i)) - 0.0106901*exp(-0.023*t(i))*sin
    + 0.00357606*exp(-0.023*t(i))*cos(0.8184*t(i)) - 0.0304402*cos(314.759*t
    elseif (n == 3)
    %dla e=120sin(2pi5t)
    uAnal(i) = -0.105971*exp(-2.0779*t(i)) + 7.90587*exp(-0.023*t(i))*sin(0.
     + 0.106809*exp(-0.023*t(i))*cos(0.8184*t(i)) - 0.000837577*cos(31.416*t
    i1Anal(i) = 0.110099*exp(-2.0779*t(i)) - 0.134624*exp(-0.023*t(i))*sin(0
    + 3.23385*exp(-0.023*t(i))*cos(0.8184*t(i)) - 3.34395*cos(31.416*t(i)) +
     0;
    i2Anal(i) = -0.469884*exp(-2.0779*t(i)) + 0.18792*exp(-0.023*t(i))*sin(0
    - 0.0628553*exp(-0.023*t(i))*cos(0.8184*t(i)) + 0.532739*cos(31.416*t(i)
     + i*(-5.55112*10^-17 * sin(31.416*t(i)) - 1.04083*10^-17
     * cos(31.416*t(i)));
    elseif (n == 5)
    %dla e=sin(t)
    uAnal(i) = -0.0029943*exp(-2.0779*t(i)) + 3.52233*exp(-0.023*t(i))*sin(0
     + 0.449056*exp(-0.023*t(i))*cos(0.8184*t(i)) - 0.446062*cos(t(i)) +
      0;
    i1Anal(i) = 0.00311093*exp(-2.0779*t(i)) - 0.224261*exp(-0.023*t(i))*sin
     + 1.43617*exp(-0.023*t(i))*cos(0.8184*t(i)) - 1.43928*cos(t(i))
      + 0;
    i2Anal(i) = 0.0132769*exp(-2.0779*t(i)) - 0.0870447*exp(-0.023*t(i))*sin
     + 0.0185062*exp(-0.023*t(i))*cos(0.8184*t(i)) - 0.0317831*cos(t(i))
      + 0;
    i1fun = @(x) (0.00311093.*exp(-2.0779.*x) - 0.224261.*exp(-0.023.*x).*si
    + 1.43617.*exp(-0.023.*x).*cos(0.8184.*x) - 1.43928.*cos(x) + 0)
    .^2;
    i2fun = @(x) (0.0132769.*exp(-2.0779.*x) - 0.0870447.*exp(-0.023.*x).*si
    + 0.0185062.*exp(-0.023.*x).*cos(0.8184.*x) - 0.0317831.*cos(x) + 0).^
    2;
    else
    uAnal(i) = 1000;
    i1Anal(i) = 1000;
    i2Anal(i) = 1000;
    end
end

u = uAnal;
i1 = i1Anal;
i2 = i2Anal;
if (n == 5)
```

```
        energia = R1*integral(i1fun,0,tmax) + R2*integral(i2fun,0,tmax);
    end
end
%funkcja do liczenia pradow
function [E] = fE(n, t)
    global h;
    switch (n(1))
        case 1
            t2 = (1 / h) * t(1);
            T = (1 / h) * 3;
            div = rem(t2, T);
            if (div < (T/2))    %dobieram takie h żeby T/2 było całkowite
                E = 120;
            else
                E = 0;
            end
        case 2
            E = 240*sin(t(1));
        case 3
            E = 210*sin(2 * pi * 5 * t(1));
        case 4
            E = 120*sin(2 * pi * 50 * t(1));
        case 5
            E = sin(t(1));
        case 6
            E = 1;
        otherwise
            E = 1000;
    end
end
%funkcja do liczenia pradow
```

3.4 Wyniki i ich weryfikacja

Obliczone energie wydzielone w trakcie 30s oraz moc czynna w tym czasie zostały przestawione w poniższej tabeli:

wymuszenie	metoda złożona prostokątów		metoda złożona parabol	
-	$\Delta t_1 = 0.01ms$	$\Delta t_2 = 10ms$	$\Delta t_1 = 0.01ms$	$\Delta t_2 = 10ms$
$e(t) = 1V$	0.187J	0.203J	0.187J	0.203J
(moc)	(0.006W)	(0.007W)	(0.006W)	(0.007W)
0 lub $120V$	1787.751J	1873.543J	1787.751J	1873.383J
(moc)	(59.592W)	(62.451W)	(59.592W)	(62.446W)
$e(t) = 240\sin(t)$	212417.542J	222358.543J	212417.534J	222358.448J
(moc)	(7080.585W)	(7411.940W)	(7080.584W)	(7411.682W)
$e(t) = 210\sin(2\pi \cdot 5)$	34.737J	35.803J	34.737J	35.797J
(moc)	(1.158W)	(1.193W)	(1.158W)	(1.193W)
$e(t) = 120\sin(2\pi \cdot 50)$	0.114J	0.114J**	0.114J	0.114J**
(moc)	(0.004W)	(0.004W)**	(0.004W)	(0.004W)**
$e(t) = \sin(t)$	3.688J	3.688J	3.688J	3.688J
(moc)	(0.123W)	(0.123W)	(0.123W)	(0.123W)

** - wartości wyliczone dla $\Delta t = 100ms$ z powodu niestabilności

Celem weryfikacji wyniku przeliczyłem ciepło oraz moc czynną korzystając z symulacji metodą Rungego-Kutty i szacując całkę metodą prostokątów. Weryfikację przeprowadziłem dla wymuszenia $e(t) = \sin(t)$, uzyskując całkowite ciepło **3.680J** oraz moc czynną **0.123W**.

Weryfikację metody prostokątów przeprowadziłem dla wymuszenia $e(t) = \sin(t)$ całkując funkcje $i_1(t)$ oraz $i_2(t)$ wyprowadzone metodą półanalityczną metodą prostokątów oraz - za drugim razem - stosując algorytm adaptacyjnej kwadratury, zaimplementowany w MATLABIE w metodzie `integral()`. Za pierwszym razem uzyskałem wynik **7.609J**, a za drugim razem **7.609J** (tyle samo).

Analizując uzyskane wyniki można zauważyć, że zasadniczo metoda prostokątów i parabol dają podobne wyniki, co więcej, porównywalne z innymi metodami, jak np. adaptując kwadraturę do przebiegu funkcji. Akurat może w tym przypadku wynika to z w miarę "porządnego" zachowywania się przebiegów bez nieprzewidywalnej dynamiki.

Porównując wyniki wobec kroku czasowego, można zauważyć, że istnieje spora różnica między nimi, w szczególności przy wymuszeniach, które pociągały za sobą duże zmienności natężeń prądów (wymuszenie pulsacyjne 0/120V, wymuszenie stałe, wymuszenia przy niskich częstotliwościach) - różnica tu wynosi ok. 5-9%. Przeciwnie, przy wymuszeniach o wysokiej częstotliwości, dających stabilny i przewidywalny przebieg natężeń, różnica wyniosła jedynie ok. 3%.

Analogiczna tabelka dla analogicznych wymuszeń, lecz zmiennego sprzężenia (przeliczyłem interpolacją wielomianową).

wymuszenie	metoda złożona prostokątów		metoda złożona parabol	
-	$\Delta t_1 = 0.01ms$	$\Delta t_2 = 10ms$	$\Delta t_1 = 0.01ms$	$\Delta t_2 = 10ms$
$e(t) = 1V$	0.2J	0.2J	0.2J	0.2J
(moc)	(0W)	(0W)	(0W)	(0W)
0 lub $120V$	1737J	1824J	1737J	1824J
(moc)	(58W)	(61W)	(58W)	(61W)
$e(t) = 240\sin(t)$	73279J	6953180J	73279J	6934593J
(moc)	(2443W)	(231772W)	(2443W)	(231153W)
$e(t) = 210\sin(2\pi \cdot 5)$	21J	22J	21J	22J
(moc)	(0.7W)	(0.7W)	(0.7W)	(0.7W)
$e(t) = 120\sin(2\pi \cdot 50)$	0.1J	***	0.1J	***
(moc)	(0W)	***	(0W)	***
$e(t) = \sin(t)$	3J	3J	3J	3J
(moc)	(0.1W)	(0.1W)	(0.1W)	(0.13W)

*** - układ niestabilny

Uwagę zwraca tutaj duża różnica zależna od kroku czasowego dla wymuszenia $e(t) = 240\sin(t)$. Po odpowiednich symulacjach ustaliłem, że przy długim kroku czasowym układ przechodzi przez fazę niestabilności, zanim dojdzie do stabilizacji - co prawdopodobnie tłumaczy wynik.

Chapter 4

Część 4

4.1 Opis modelu matematycznego

Jak wynika z poprzednich części, istnieje zależność między energią wydzielaną w obwodzie, a częstotliwością wymuszenia. Szukając częstotliwości, dla której wydzielona energia będzie równa 406J, jest szukaniem miejsca zerowego funkcji $F(f)$.

$$F(f) = P(f) - 406 \tag{4.1}$$

Z matematycznego punktu widzenia, ponieważ $P(f)$ jest zdefiniowana jako całka oznaczona dwóch funkcji: $i_1(t)$ i $i_2(t)$, będących rozwiązaniem układu równań tego obwodu, znalezienie miejsca zerowego w sposób analityczny jest zbyt skomplikowane.

4.2 Opis modelu numerycznego

Aby znaleźć miejsca zerowe danej funkcji w sposób numeryczny, należy najpierw znaleźć przedziały izolacji. Rysując wykres $F(f)$ dla różnych f osiągnąłem wykres jak na rysunku 4.1.

Dla wartości powyżej 2Hz funkcja jest malejąca, a moc $P(f)$ stopniowo dąży do zera. Widać, że $F(f)$ posiada dwa miejsca zerowe. Jedno "mniej więcej" w przedziale (0.03-0.05), a drugie w przedziale (0.6-0.7). Ponieważ istnieje metoda szukania miejsc zerowych, która nie wymaga niczego innego prócz zmiany znaku funkcji na przedziałach - metoda bisekcji - zastosuję ją by je oszacować. Okazuje się, że miejsca zerowe to (w przybliżeniu): 0.035 i 0.676 Hz.

To nie kończy rozwiązania problemu. Aby dokładnie określić miejsca zerowe, należy dokładniej określić przedziały izolacji, najlepiej tak, by nie zawierały punktów przegięcia. Ponadto, jeśli nalezy zastosować inne metody

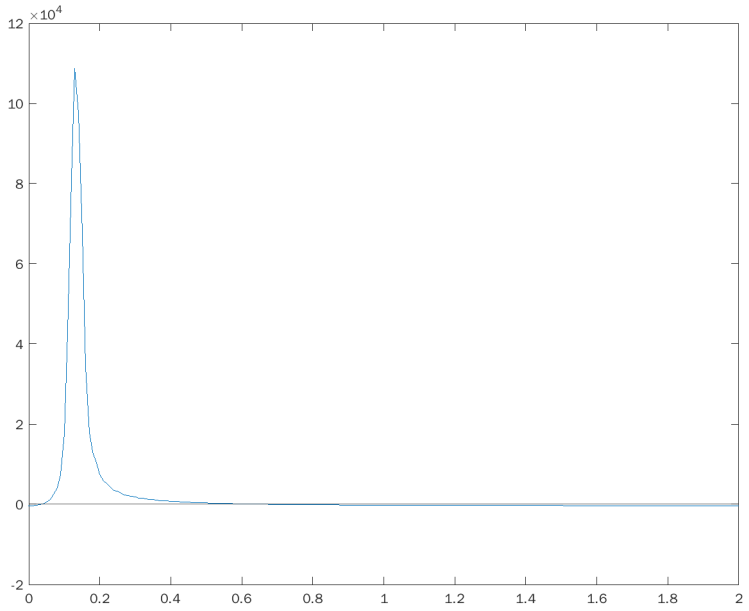


Figure 4.1: Wykres funkcji $F(f)$ obliczonej numerycznie

niż metodę bisekcji, trzeba znać znak pierwszej i drugiej pochodnej szukanej funkcji, gdyż niektóre metody wymagają rozpoczęcia poszukiwań od miejsca o tym samym znaku co druga pochodna. Natomiast, aby znaleźć wartość pierwszej i drugiej pochodnej należy odkryć odpowiednią zmianę argumentu Δ f, która pozwoli te pochodne wyliczyć z akceptowalną dokładnością. Warunki akceptowalnej dokładności są podane w treści zadania, co pozwala na wyprowadzenie algorytmu który wyliczy Δ f.

Następnie, mając wyliczone Δ f, można przybliżyć wartość pierwszej i drugiej pochodnej w obu przedziałach izolacji.

Ostatecznie, powtarzając metodę bisekcji dla lepiej wyznaczonych przedziałów oraz startując z metodą Newtona i siecznych, można będzie wyznaczyć miejsca zerowe $F(f)$.

4.3 Opis implementacji

Poniżej prezentuję kod służący do rozwiązania zadania według powyższych zasad. Ponieważ kolejność wykonywanych czynności jest istotna, skrypt jest uruchamiany kilkukrotnie, za każdym razem wykomentowując niepotrzebne fragmenty. Czyni to zadość wymaganiu w treści zadania, by zaimplementować osobne warianty programu dla każdej części przy jednym pliku źródłowym (wymaganie formalne do projektu jako całości). Objaśnienia działania skryptu

są w jego komentarzach.

```
function [result] = Czesc4()
    global h;
    h = 0.001;
    global tmax;
    tmax = 30;
    global dok;
    dok = 0.0000000001;
    global iteracjeBisekcja;
    iteracjeBisekcja = 0;
    global iteracjeBisekcja2;
    iteracjeBisekcja2 = 0;
    global iteracjeSieczne;
    iteracjeSieczne = 0;
    global iteracjeNewton;
    iteracjeNewton = 0;

   % Wstepny wykres - poniższa część generuje wstępny wykresi pozwalający
   % oszacować miejsca zerowe
    ff=0:0.01:100;
    FF = zeros(C1,length(ff));
    for i=1:length(ff)
        FF(i) = ObliczMoc(ff(i));
    end
    figure(1)
    plot(ff,FF);
    yline(0);

    % przedział izolacji i delta f - obliczone w późniejszych częściach
    a = 0.035;
    b = 0.037;
    delf = 0.0001;

    zeroBisekcja = findZeroBisekcja(a, b);%liczy miejsce zerowe met. bisekcji
    zeroBisekcja2 = findZeroBisekcja2(a, b);%liczy jw. tylko korzysta z
     met. Rungego-Kutty
    fprintf('bisekcja: %.9f, iteracji %.1f\nbisekcja2: %.9f, iteracji %.
    1f\n', zeroBisekcja, iteracjeBisekcja, zeroBisekcja2, iteracjeBisekcja2);

    zeroSieczne = findZeroSieczne(a, a+0.001);%liczy met. siecznych
```

```
    fprintf('sieczne: %.9f, iteracji: %.1f\n', zeroSieczne, iteracjeSieczne);

    zeroNewton = findZeroNewton(a, delf);%liczy met. Newtona
    fprintf('newton: %.9f, iteracji: %.1f\n', zeroNewton, iteracjeNewton);

%LICZENIE DF - ten moduł wylicza odpowiedni krok w argumencie zgodnie z
%zasadami z treści zadania
    df = 0.01;%od tego zaczynami
    warunek = 1;
    f = a;
    F1_1 = ObliczMoc(f);
    while (f < b)
        while (warunek)
            F1_2 = ObliczMoc(f + df);
            F1p = (F1_2 - F1_1)/df;
            F1_2pr = ObliczMoc(f + (df/2));
            F1p_pr = (F1_2pr - F1_1) / (df/2);
            if (abs((F1p_pr - F1p)/F1p) < 0.01)
                warunek = 0;
            else
                warunek = 1;
                df = df/2;
            end
        end
        warunek = 1;
        f = f + df;
        F1_1 = F1_2;
    end
    fprintf('df = %.9f',df);

%LICZENIE F(f) - ten moduł wylicza wartości funkcji F(f) oraz jej pierwsza
%i druga pochodną
    df = 0.0001;
    f = a:df:b;
    F1 = zeros(1, length(f));
    for i=1:length(f)
        F1(i) = ObliczMoc(f(i));
    end
% %WYKRESY
    F1p = zeros(1, length(f));
    for i=1:(length(F1)-1)
```

```
        F1p(i) = (F1(i+1)-F1(i))/df;
    end
    F1b = zeros(1, length(f));
    for i=1:(length(F1p)-1)
        F1b(i) = (F1p(i+1) - F1p(i))/df;
        if (i > 1 && (F1b(i)*F1b(i-1)) < 0)
            disp(f(i));
        end
    end

    figure(1);
    hold on;
    plot(f, F1);
    plot(zeroBisekcja,0,'o');
    Cyline(0);
    title('F(f)');
    figure(2);
    plot(f, F1p);
    yline(0);
    title('Fprim(f)');
    figure(3);
    plot(f, F1b);
    yline(0);
    title('Fbis(f)');
    hold off;

    result = 'wykonano';
end

function [zero] = findZeroNewton(x0, df)
    global dok;
    global iteracjeNewton;
    iteracjeNewton = iteracjeNewton + 1;
    funk = ObliczMoc(x0);
    poch = (ObliczMoc(x0 + df) - funk)/df;
    x1 = x0 - (funk/poch);
    if (abs(ObliczMoc(x1)) < dok)
        zero = x1;
        return;
    end
    zero = findZeroNewton(x1, df);
```

```
end

function [zero] = findZeroSieczne(x0, x1)
    global iteracjeSieczne;
    iteracjeSieczne = iteracjeSieczne + 1;
    global dok;
    x2 = x1 - (ObliczMoc(x1)*((x1 - x0)/(ObliczMoc(x1)-ObliczMoc(x0))));
    if (abs(ObliczMoc(x2)) < dok)
        zero = x2;
        return;
    end
    zero = findZeroSieczne(x1, x2);
end

function [zero] = findZeroBisekcja(a, b)
    global dok;
    global iteracjeBisekcja;
    iteracjeBisekcja = iteracjeBisekcja + 1;
    f = (a + b) / 2;
    mocf = ObliczMoc(f);
    if (abs(mocf) < dok)
        zero = f;
        return;
    end
    if ((ObliczMoc(a) * mocf) < 0)
        zero = findZeroBisekcja(a, f);
    else
        zero = findZeroBisekcja(f, b);
    end
end

function [zero] = findZeroBisekcja2(a, b)
    global dok;
    global iteracjeBisekcja2;
    iteracjeBisekcja2 = iteracjeBisekcja2 + 1;
    f = (a + b) / 2;
    mocf = ObliczMoc2(f);
    if (abs(mocf) < dok)
        zero = f;
        return;
    end
```

```
    if ((ObliczMoc2(a) * mocf) < 0)
        zero = findZeroBisekcja2(a, f);
    else
        zero = findZeroBisekcja2(f, b);
    end
end

function [E] = fE(f, t)
    E = 100 * sin(2 * pi * f * t);
end

function [F] = ObliczMoc2(f)
    R1 = 0.1;
    R2 = 10;
    [t, y] = LiczMacierze(f);
    i1M = y(:,1);
    i2M = y(:,2);
    calkaProstMac1 = 0;
    calkaProstMac2 = 0;
    calkaProstMac = 0;
    for i=1:(length(t)-1)
        calkaProstMac1 = calkaProstMac1 + ((t(i+1)-t(i))*i1M(i)^2);
        calkaProstMac2 = calkaProstMac2 + ((t(i+1)-t(i))*i2M(i)^2);
    end
    calkaProstMac1 = calkaProstMac1 * R1;
    calkaProstMac2 = calkaProstMac2 * R2;
    calkaProstMac = calkaProstMac1 + calkaProstMac2;
    F = calkaProstMac - 406;
end

function [t, y] = LiczMacierze(f)
    global tmax;
    [t,y] = ode45(@(t,y) odefun(t, y, f), [0 tmax], [0 0 0]);
end

function dydt = odefun(t, y, f)
    R1 = 0.1;
    R2 = 10;
    C = 0.5;
    L1 = 3;
    L2 = 5;
```

```
    M = 0.8;
    dydt = [(1/(L1/M - M/L2))*((-R1/M)*y(1) + (R2/L2)*y(2) - (1/M)*y(3) +
    (1/M)*100*sin(2*pi*f*t));
        (1/(M/L1 - L2/M))*((-R1/L1)*y(1) + (R2/M)*y(2) - (1/L1)*y(3) +
        (1/L1)*100*sin(2*pi*f*t));
        (1/C)*y(1)];
end

function [F] = ObliczMoc(f)
    global h;
    global tmax;
    R1 = 0.1;
    R2 = 10;
    C = 0.5;
    L1 = 3;
    L2 = 5;
    M = 0.8;
    di1dt = @(i1,i2,uC,E) ((1/((L1/M)-(M/L2)))*(((-R1/M)*i1)+((R2/L2)*i2)-
    ((1/M)*uC)+((1/M)*E)));
    di2dt = @(i1,i2,uC,E) ((1/((M/L1) - (L2/M)))*(((-R1/L1)*i1)+((R2/M)*i2)-
    ((1/L1)*uC)+((1/L1)*E)));
    %di1dt = @(i1,i2,uC,E) ((-1/M) * (R2*i2 + L2 * di2dt(i1,i2,uC,E)));
    duCdt = @(i1) ((1/C)*i1);

    t = 0:h:tmax;
    E = zeros(1, length(t));
    i1 = zeros(1, length(t));
    i2 = zeros(1, length(t));
    uC = zeros(1, length(t));
    i = 1;
    E(1) = fE(f, 0); %funkcja fE liczy E(t) dla roznych (n) podpunktow
    tego zadania (na dole)
    while (t(i) < tmax)
        E(i+1) = fE(f, t(i+1));%E(t) rysuje sie dobrze
        i1(i+1) = i1(i) + (h * di1dt(i1(i),i2(i),uC(i),E(i)));
        i2(i+1) = i2(i) + (h * di2dt(i1(i),i2(i),uC(i),E(i)));
        uC(i+1) = uC(i) + (h * duCdt(i1(i)));
        i = i + 1;
    end
    %liczenie calki
    calkaProstokat1 = 0;
```

```
    calkaProstokat2 = 0;
    calkaProstokat = 0;
    for i=1:length(t)
        calkaProstokat1 = calkaProstokat1 + h*(i1(i)^2);
        calkaProstokat2 = calkaProstokat2 + h*(i2(i)^2);
    end
    calkaProstokat1 = calkaProstokat1 * R1;
    calkaProstokat2 = calkaProstokat2 * R2;
    calkaProstokat = calkaProstokat1 + calkaProstokat2;
    F = (calkaProstokat) - 406;
end
```

4.4 Wyniki i ich weryfikacja

Stosując powyższy algorytm, dla wstępnego pierwszego przedziału izolacji 0.03-0.04 Δ f wynosi **0.00031**. Analogicznie, dla wstępnego drugiego przedziału izolacji Δ f wynosi **0.00007**.

Rysując funkcję $F(f)$ i jej dwie pierwsze pochodne w opisywanych przedziałach można osiągnąc następujące wykresy. Dla pierwszego, wstępnego, przedziału izolacji - na rysunkach 4.2 i 4.3. Dla drugiego, wstępnego, przedziału izolacji - na rysunkach 4.4 i 4.5.

Na podstawie tych wykresów widać, że dobrymi przedziałami izolacji - to jest takimi, w których zarówno pierwsza, jak i druga pochodna nie zmieniają znaku, będą: **(0.035, 0.037)** i **(0.674, 0.678)**.

Teraz można bezpiecznie zastosować wszystkie trzy metody szukania miejsc zerowych i je ze sobą porównać. Porównanie jest obecne w wypełnionej wymaganej tabeli.

metoda	częstotliwość [Hz]	F [J]	liczba iteracji	liczba obliczeń mocy
bisekcji	0.035334269	0	38	38
-	0.675589931	0	34	34
siecznych	0.035334269	0	4	4
-	0.675589931	0	5	5
quasi-Newtona	0.035334269	0	4	4
-	0.675589931	0	5	5

Celem weryfikacji przeliczyłem metodę bisekcji dla układu rozwiązanego metodą Rungego-Kutty uzyskując wyniki 0.035388414 i 0.676495791 Hz -

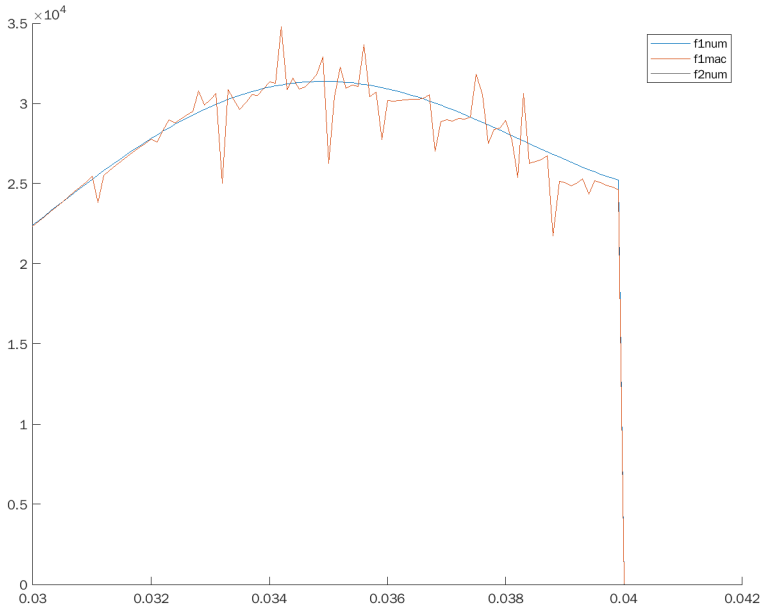


Figure 4.2: Wykres pierwszej pochodnej w pierwszym przedziale $F(f)$ obliczonej numerycznie

wyniki zbliżone do powyższych.

Podsumowując, najbardziej efektywne są metody siecznych i Newtona. Jednak to metoda bisekcji (jako metoda o najmniejszej liczbie warunków) umożliwiła wstępne oszacowanie miejsc zerowych celem dokładniejszej ich analizy i znalezienia przedziałów, gdzie można (skutecznie) zastosować pozostałe dwie metody.

Oba miejsca zerowe wraz z przebiegiem funkcji są zaznaczone na wykresach 4.6 i 4.7.

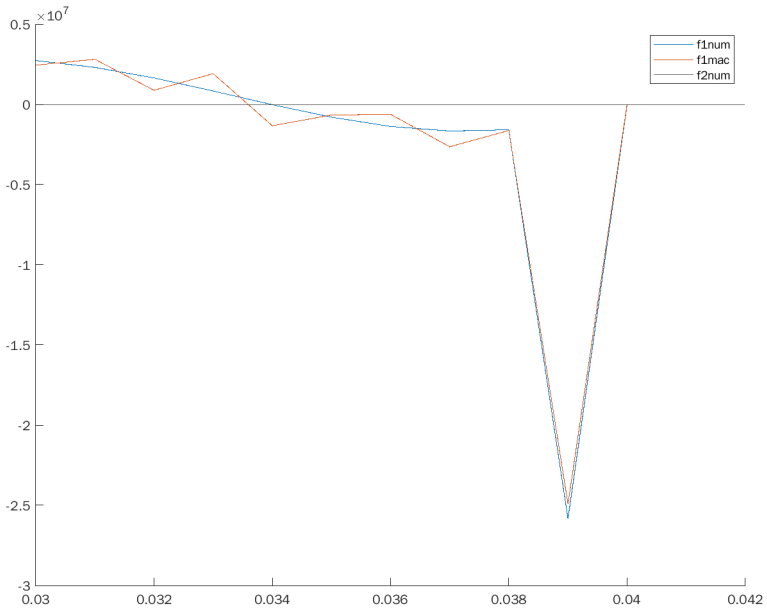


Figure 4.3: Wykres drugiej pochodnej w pierwszym przedziale $F(f)$ obliczonej numerycznie

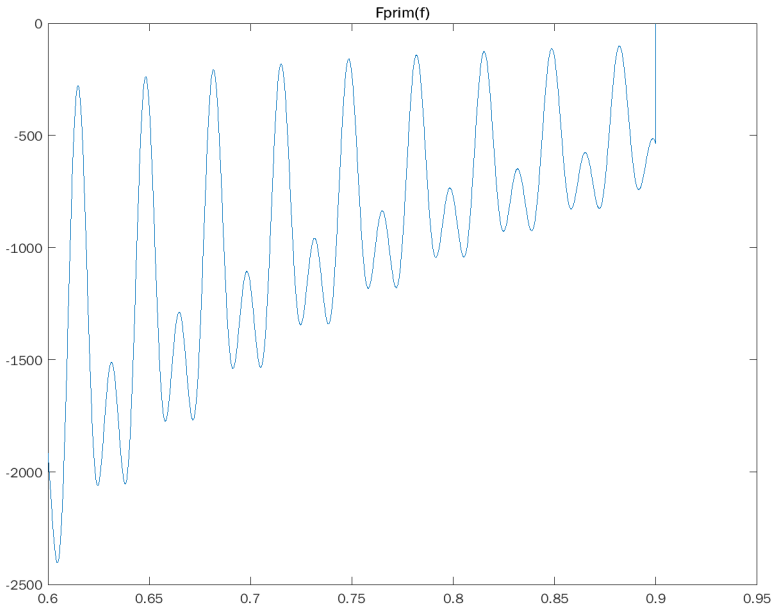


Figure 4.4: Wykres pierwszej pochodnej w drugim przedziale $F(f)$ obliczonej numerycznie

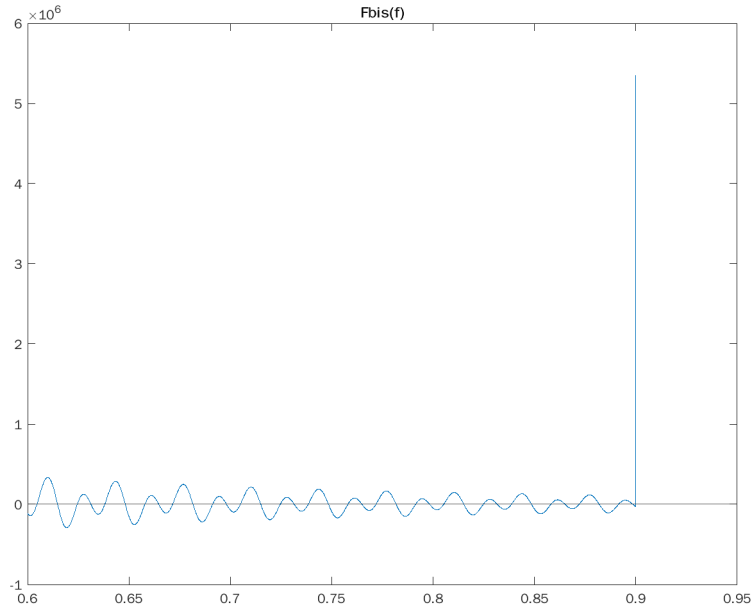


Figure 4.5: Wykres drugiej pochodnej w drugim przedziale $F(f)$ obliczonej numerycznie

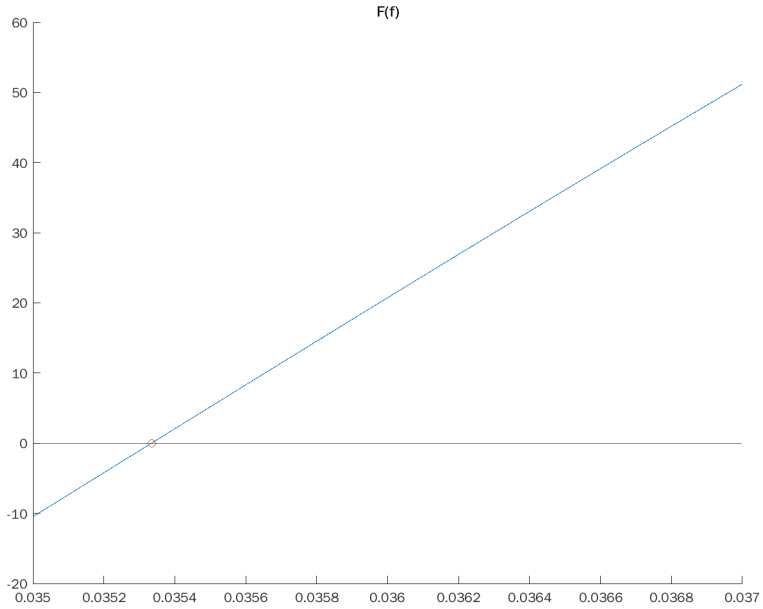


Figure 4.6: Pierwsze miejsce zerowe $F(f)$ obliczone numerycznie

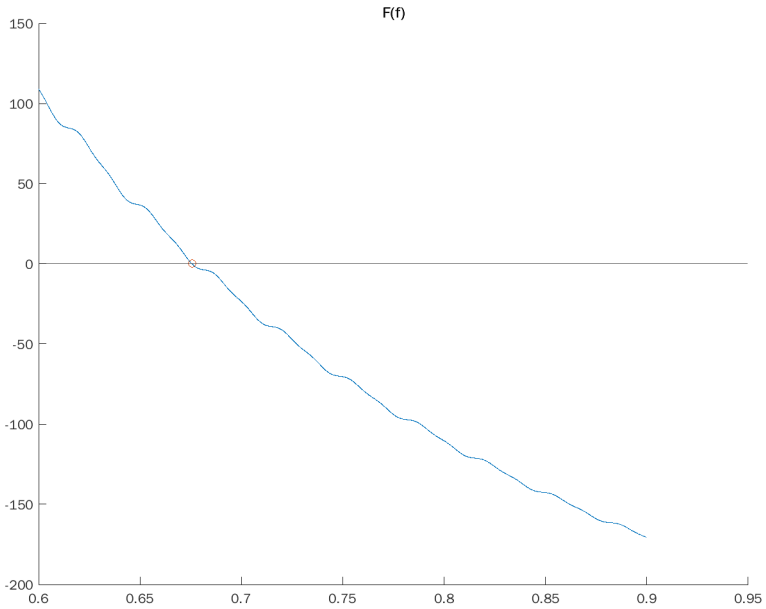
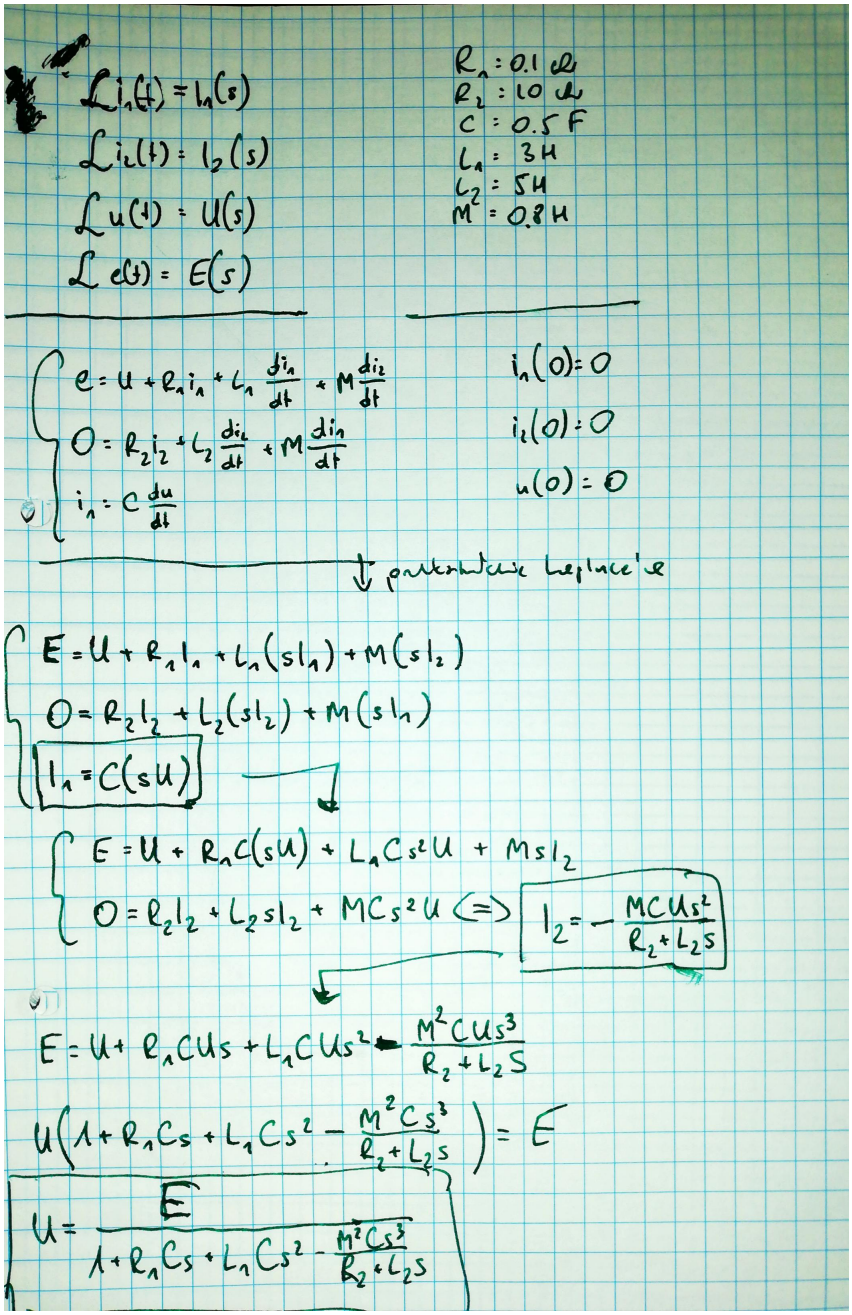
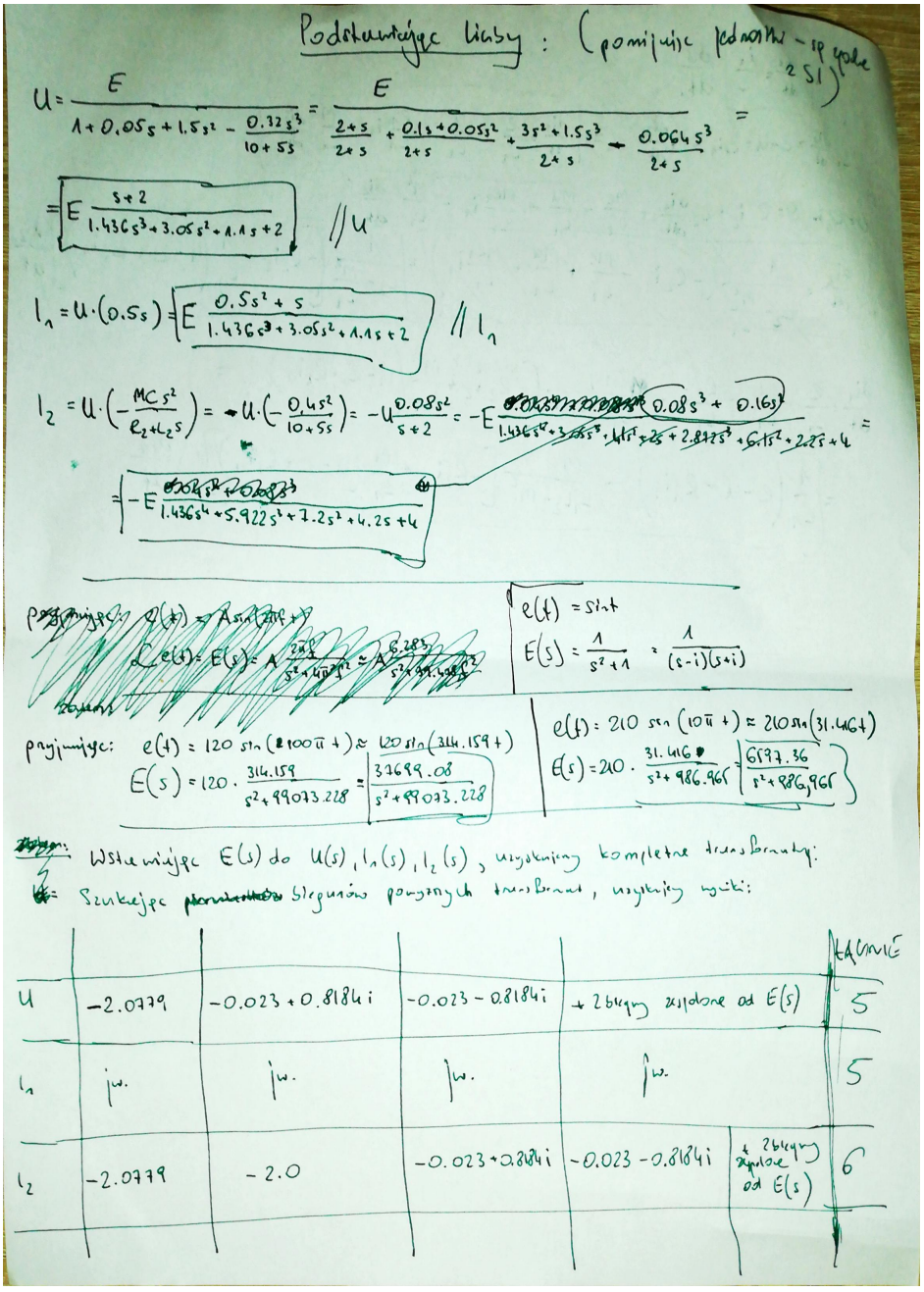


Figure 4.7: Drugie miejsce zerowe $F(f)$ obliczone numerycznie

Chapter 5

Wyliczenia (pół)analityczne


$\mathcal{L}\, i_1(t) = I_1(s)$
$\mathcal{L}\, i_2(t) = I_2(s)$
$\mathcal{L}\, u(t) = U(s)$
$\mathcal{L}\, e(t) = E(s)$
$R_1 = 0.1\ \Omega$
$R_2 = 10\ \Omega$
$C = 0.5\ F$
$L_1 = 3\ H$
$L_2 = 5\ H$
$M = 0.8\ H$
$e = u + R_1 i_1 + L_1 \frac{di_1}{dt} + M \frac{di_2}{dt}$
$0 = R_2 i_2 + L_2 \frac{di_2}{dt} + M \frac{di_1}{dt}$
$i_1 = C \frac{du}{dt}$
$i_1(0) = 0$
$i_2(0) = 0$
$u(0) = 0$
↓ partendo dalla Laplace'se
$E = U + R_1 I_1 + L_1 (sI_1) + M(sI_2)$
$0 = R_2 I_2 + L_2 (sI_2) + M(sI_1)$
$I_1 = C(sU)$
$E = U + R_1 C(sU) + L_1 C s^2 U + M s I_2$
$0 = R_2 I_2 + L_2 s I_2 + M C s^2 U \Leftrightarrow I_2 = -\frac{MCUs^2}{R_2 + L_2 s}$
$E = U + R_1 CUs + L_1 CUs^2 - \frac{M^2 CUs^3}{R_2 + L_2 s}$
$U\left(1 + R_1 Cs + L_1 Cs^2 - \frac{M^2 Cs^3}{R_2 + L_2 s}\right) = E$
$U = \frac{E}{1 + R_1 Cs + L_1 Cs^2 - \frac{M^2 Cs^3}{R_2 + L_2 s}}$


Podstawiając liczby : (pomijamy jednostki - są spójne z SI)
$U = \frac{E}{1 + 0.05s + 1.5s^2 - \frac{0.32s^3}{10+5s}} = \frac{E}{\frac{2+s}{2+s} + \frac{0.1s+0.05s^2}{2+s} + \frac{3s^2+1.5s^3}{2+s} - \frac{0.064s^3}{2+s}} =$
$= E\frac{s+2}{1.436s^3+3.05s^2+1.1s+2}$ // U
$I_1 = U\cdot(0.5s) = E\frac{0.5s^2+s}{1.436s^3+3.05s^2+1.1s+2}$ // I_1
$I_2 = U\cdot\left(-\frac{MCs^2}{R_2+L_2s}\right) = U\cdot\left(-\frac{0.4s^2}{10+5s}\right) = -U\frac{0.08s^2}{s+2} = -E\frac{0.08s^3+0.16s^2}{1.436s^4+5.922s^3+7.2s^2+4.2s+4}$
$e(t) = \sin t$
$E(s) = \frac{1}{s^2+1} = \frac{1}{(s-i)(s+i)}$
pomijamy: $e(t) = 120\sin(2\cdot100\pi t) \approx 120\sin(314.159t)$
$E(s) = 120\cdot\frac{314.159}{s^2+99073.228} = \frac{37699.08}{s^2+99073.228}$
$e(t) = 210\sin(10\pi t) = 210\sin(31.416t)$
$E(s) = 210\cdot\frac{31.416}{s^2+986.965} = \frac{6597.36}{s^2+986.965}$
Wstawiając E(s) do U(s), $I_1(s)$, $I_2(s)$, uzyskujemy kompletne transmitancje.
Szukając biegunów powyższych transmitancji, uzyskujemy wyniki:
					OCENA
U	−2.0779	−0.023 + 0.8184i	−0.023 − 0.8184i	+ 2 bieguny zyskane od E(s)	5
I_1	j.w.	j.w.	j.w.	j.w.	5
I_2	−2.0779	−2.0	−0.023 + 0.8184i	−0.023 − 0.8184i + 2 bieguny zyskane od E(s)	6

$$U(s) = \frac{s+2}{(s+2.0779)(s+0.023-0.8184i)(s+0.023+0.8184i)(s-i)(s+i)}$$

$$\operatorname*{res}_{-i} U(s)e^{st} + \operatorname*{res}_{+i} U(s)e^{st} = 2\,\mathrm{Re}\left\{\operatorname*{res}_{+i} U(s)e^{st}\right\} = 2\,\mathrm{Re}\lim_{s\to i}\frac{s+2}{(\dots)(s+i)}e^{st} =$$

$$= 2\,\mathrm{Re}\left\{\frac{i+2}{(i+2.0779)(i+0.023-0.8184i)(i+0.023+0.8184i)(i+i)}e^{it}\right\} =$$

$$= 2\,\mathrm{Re}\left\{(-0.2230 + 1.4393\,i)e^{it}\right\} =$$

$$= \boxed{-2.8786\sin t - 0.446\cos t}$$

$$L(s) = s+2$$

$$M(s) = 1.436s^5 + 3.05s^4 + 1.1s^3 + 2s^2 + 1.436s^3 + 3.05s^2 + 1.1s + 2 =$$

$$= 1.436\,s^5 + 3.05\,s^4 + 2.536\,s^3 + 5.05\,s^2 + 1.1s + 2$$

$$M'(s) = 7.18s^4 + 12.2s^3 + 7.608s^2 + 10.1s + 1.1$$

$$\operatorname*{res}_{s=-2.0779} U(s)e^{st} = \boxed{\frac{-0.0779}{37.3589}e^{-2.0779t}}$$

$$2\,\mathrm{Re}\left\{\operatorname*{Res}_{-0.023+0.8184i} U(s)e^{st}\right\} = 2\,\mathrm{Re}\left\{\frac{1.937 + 0.8184i}{(2.0549+0.8184i)(1.6368i)(-0.023-0.8816i)(-0.023+1.8841i)}e^{(-0.023+0.8184i)t}\right\}$$

$$= 2\,\mathrm{Re}\left\{(0.2245 - 1.7612\,i)e^{(-0.023+0.8184i)t}\right\} =$$

$$= \boxed{3.5224\,e^{-0.023t}\sin(0.8184t) + 0.449\,e^{-0.023t}\cos(0.8184t)}$$

$$\frac{du}{dt} = \frac{i_1}{C} \Big| \frac{d}{dt}.$$

$$L_1 \frac{di_1}{dt} = e - U - R_1 i_1 - M \frac{di_2}{dt} \quad \Leftrightarrow \quad \frac{di_1}{dt} = \frac{e}{L_1} - \frac{U}{L_1} - \frac{R_1}{L_1} i_1 - \frac{M}{L_1} \frac{di_2}{dt}$$

$$0 = R_2 i_2 + L_2 \frac{di_2}{dt} + \frac{Me}{L_1} - \frac{Mu}{L_1} - \frac{MR_1}{L_1} i_1 - \frac{M^2}{L_1} \frac{di_2}{dt}$$

$$\frac{di_2}{dt} = \frac{1}{L_2 - \frac{M^2}{L_1}} \left(-R_2 i_2 - \frac{Me}{L_1} + \frac{Mu}{L_1} + \frac{MR_1}{L_1} i_1 \right) = \frac{1}{L_2 - \frac{M^2}{L_1}} \left[-R_2 i_2 + \frac{M}{L_1} \left(U - e + R_1 i_1 \right) \right] \frac{di_2}{dt}$$

$$\frac{di_1}{dt} = \frac{e}{L_1} - \frac{U}{L_1} - \frac{R_1}{L_1} i_1 - \frac{M}{L_1 L_2 - M^2} \left[-R_2 i_2 + \frac{M}{L_1} \left(U - e + R_1 i_1 \right) \right] =$$

$$= \left(\frac{1}{L_1} \left(e - U - R_1 i_1 \right) - \frac{M}{L_1 L_2 - M^2} \left[-R_2 i_2 + \frac{M}{L_1} \left(U - e + R_1 i_1 \right) \right] \right) \frac{di_1}{dt}$$