Dziembor Dawid

metody numeryczne

Projekt Zima 2020 - Raport

Politechnika Warszawska

Ośrodek Kształcenia na Odległość

# Wstęp

W raporcie przedstawione będzie działanie i różne obliczenia z zakresu metod numerycznych dla symulatora obwodu elektrycznego pełniącego funkcję filtra środkowo-przepustowego. W celu przygotowania obliczeń posłużono się środowiskiem MATLAB, jednak większość użytych funkcji jest utworzona przez autora raportu bez zastosowania wbudowanych funkcji języka programowania pakietu MATLAB.

# Część 1

function bandpass(freq)

R1 = 30;

R2 = 10;

C1 = 0.1e-6;

C2 = 0.2e-6;

RL = 1e8;

h = 1e-7;

t = [ 0 : h : 0.1e-3];

if freq>0

e = @(t) sin(2\*pi\*t\*freq);

end

if freq=='nosin'

e = @(t) 1;

end

if freq=='cycle'

e = @(t) rectpulse(t,0.05e-3);

end

dy = @(t,y) ...

[ 1/C1 \* ( (e(t) - y(1) - y(2))/R2 + (e(t) - y(1))/R1 )

1/C2 \* ( (e(t) - y(1) - y(2))/R2 - y(2)/RL ) ];

%u = euler(t, h, dy); %metoda Eulera

u = beuler(t, h, dy); %ulepszona metoda Eulera

plot(t, u(1,:)); %rysuj wykres u1

%plot(t, u(2,:)); %rysuj wykres u2

xlim([0 1e-4])

grid on

end

%metoda Eulera

function y = euler(t,h,f)

y = [0 0]';

for i = 1 : length(t)-1

y(:, i+1) = y(:, i) + h \* f(t(i), y(:, i));

end

end

%ulepszona metoda Eulera

function y = beuler(t,h,f)

y = [0 0]';

for i = 1 : length(t)-1

prediction = y(:, i) + h/2 \* f(t(i), y(:, i));

y(:, i+1) = y(:, i) + h \* f(t(i) +h/2, prediction);

end

end

%funkcja okresowa kwadratowokształtna

function y = rectpulse(x,T)

modulo = mod(x,T);

if modulo<(T/2)

y = 1;

else

y = 0;

end

end

W powyżej przedstawionym skrypcie zaimplementowano funkcje o nazwach: euler, która realizuje metodę Eulera, oraz beuler która realizuje ulepszoną metodę Eulera. Jest zaimplementowana również funkcja rectpulse, która jest funkcją pomocniczą aby stworzyć funkcję okresowo zmieniającą się między wartościami 0 i 1.

W celu wywołania metody Eulera należy odkomentować linię:

u = euler(t, h, dy);

natomiast w celu wywołania metody ulepszonej Eulera odkomentować linię

u = beuler(t, h, dy);

Analogicznie, w celu rysowania wykresu przebiegu napięcia U1 na kondensatorze C1 należy odkomentować linię:

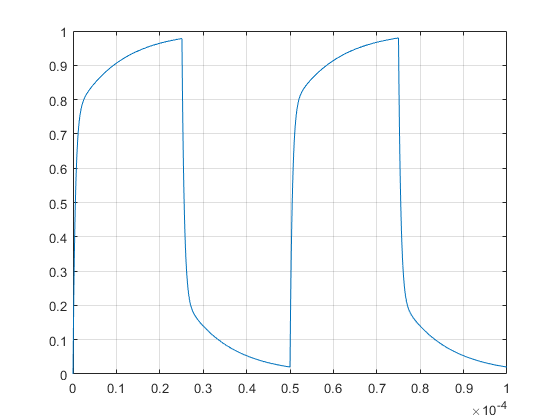
plot(t, u(1,:));

Natomiast, w celu rysowania wykresu przebiegu napięcia U2 na kondensatorze C2 należy odkomentować linię:

plot(t, u(2,:));

Funkcja **bandpass** przyjmuje jeden wybrany parametr:

* częstotliwość *f* źródła napięcia, gdzie e(t) = sin(2πft)
* *nosin –* wówczas e(t)=1V
* *cycle* – wówczas realizowana jest okresowe wymuszenie

Efektem wywołania funkcji bandpass z zadanymi parametrami jest utworzenie wykresów przedstawionych poniżej. 

|  |  |
| --- | --- |
|  |  |
| Przebieg napięcia u1 wymuszenia stałego  e(t) = 1 V, Metoda Eulera | Przebieg napięcia u1 wymuszenia  e(t) = sin(2πft), przy założeniu f = 50 Hz, Metoda Eulera |
|  |  |
| Przebieg napięcia u1 wymuszenia  e(t) = sin(2πft), przy założeniu f = 500 Hz, Metoda Eulera | Przebieg napięcia u1 wymuszenia  e(t) = sin(2πft), przy założeniu f = 5 kHz, Metoda Eulera |
|  |  |
| Przebieg napięcia u1 wymuszenia  e(t) = sin(2πft), przy założeniu f = 60 kHz, Metoda Eulera | Przebieg napięcia u1 wymuszenia  e(t) = sin(2πft), przy założeniu f = 80 kHz, Metoda Eulera |

|  |  |
| --- | --- |
|  |  |
| Przebieg napięcia u1 wymuszenia  e(t) = sin(2πft), przy założeniu f = 200 kHz, Metoda Eulera | Przebieg napięcia u1 wymuszenia  e(t) = sin(2πft), przy założeniu f = 600 kHz, Metoda Eulera |
|  |  |
| Przebieg napięcia u1 wymuszenia  E=1 dla t<T/2 i E=0 dla t>=T/2 dla T=0.05ms | Przebieg napięcia u2 wymuszenia stałego  e(t) = 1 V, Metoda Eulera |
|  |  |
| Przebieg napięcia u2 wymuszenia  e(t) = sin(2πft), przy założeniu f = 50 Hz, Metoda Eulera | Przebieg napięcia u2 wymuszenia  e(t) = sin(2πft), przy założeniu f = 500 Hz, Metoda Eulera |
|  |  |
| Przebieg napięcia u2 wymuszenia  e(t) = sin(2πft), przy założeniu f = 5 kHz, Metoda Eulera | Przebieg napięcia u2 wymuszenia  e(t) = sin(2πft), przy założeniu f = 60 kHz, Metoda Eulera |
|  |  |
| Przebieg napięcia u2 wymuszenia  e(t) = sin(2πft), przy założeniu f = 80 kHz, Metoda Eulera | Przebieg napięcia u2 wymuszenia  e(t) = sin(2πft), przy założeniu f = 200 kHz, Metoda Eulera |
|  |  |
| Przebieg napięcia u2 wymuszenia  e(t) = sin(2πft), przy założeniu f = 600 kHz, Metoda Eulera | Przebieg napięcia u2 wymuszenia  E=1 dla t<T/2 i E=0 dla t>=T/2 dla T=0.05ms,  Metoda Eulera |
|  |  |
| Przebieg napięcia u1 wymuszenia stałego  e(t) = 1 V, Metoda Ulepszona Eulera | Przebieg napięcia u1 wymuszenia  e(t) = sin(2πft), przy założeniu f = 50 Hz,  Metoda Ulepszona Eulera |
|  |  |
| Przebieg napięcia u1 wymuszenia  e(t) = sin(2πft), przy założeniu f = 500 Hz,  Metoda Ulepszona Eulera | Przebieg napięcia u1 wymuszenia  e(t) = sin(2πft), przy założeniu f = 5 kHz,  Metoda Ulepszona Eulera |
|  |  |
| Przebieg napięcia u1 wymuszenia  e(t) = sin(2πft), przy założeniu f = 60 kHz,  Metoda Ulepszona Eulera | Przebieg napięcia u1 wymuszenia  e(t) = sin(2πft), przy założeniu f = 80 kHz,  Metoda Ulepszona Eulera |
|  |  |
| Przebieg napięcia u1 wymuszenia  e(t) = sin(2πft), przy założeniu f = 200 kHz,  Metoda Ulepszona Eulera | Przebieg napięcia u1 wymuszenia  e(t) = sin(2πft), przy założeniu f = 600 kHz,  Metoda Ulepszona Eulera |
|  |  |
| Przebieg napięcia u1 wymuszenia  E=1 dla t<T/2 i E=0 dla t>=T/2 dla T=0.05ms,  Metoda Ulepszona Eulera | Przebieg napięcia u2 wymuszenia stałego  e(t) = 1 V, Metoda Ulepszona Eulera |
|  |  |
| Przebieg napięcia u2 wymuszenia  e(t) = sin(2πft), przy założeniu f = 50 Hz,  Metoda Ulepszona Eulera | Przebieg napięcia u2 wymuszenia  e(t) = sin(2πft), przy założeniu f = 500 Hz,  Metoda Ulepszona Eulera |
|  |  |
| Przebieg napięcia u2 wymuszenia  e(t) = sin(2πft), przy założeniu f = 5 kHz,  Metoda Ulepszona Eulera | Przebieg napięcia u2 wymuszenia  e(t) = sin(2πft), przy założeniu f = 60 kHz,  Metoda Ulepszona Eulera |
|  |  |
| Przebieg napięcia u2 wymuszenia  e(t) = sin(2πft), przy założeniu f = 80 kHz,  Metoda Ulepszona Eulera | Przebieg napięcia u2 wymuszenia  e(t) = sin(2πft), przy założeniu f = 200 kHz,  Metoda Ulepszona Eulera |
|  |  |
| Przebieg napięcia u2 wymuszenia  e(t) = sin(2πft), przy założeniu f = 600 kHz,  Metoda Ulepszona Eulera | Przebieg napięcia u2 wymuszenia  E=1 dla t<T/2 i E=0 dla t>=T/2 dla T=0.05ms,  Metoda Ulepszona Eulera |

# Część 2

W tej części projektu będzie porównywana skuteczność różnych metod wyznaczania częstotliwości granicznej f analizowanego w tym projekcie filtra. W tym celu skrypt będzie szukał rozwiązania równania liniowego zadanego przez warunki zadania, przy przyjęciu określonych parametrów.

function fc = find\_fc(R1, R2, C1, C2)

F = @(x) (0.7071 \* 0.2727)...

- abs(((2i \* pi \* x) \* (1/(C2 \* R2))) ...

/ (((2i \* pi \* x) ^ 2) + (2i \* pi \* x)...

\* (1 / (C1 \* R1) + 1 / (C1 \* R2) + 1 / (C2 \* R2))...

+ 1 / (C1 \* C2 \* R1 \* R2)));

precision = 1e-7;

fprintf("Metoda bisekcji | dolna ");

freq\_bisf(F, 0, 35000, precision);

fprintf("Metoda bisekcji | górna ");

freq\_bisf(F, 300000, 500000, precision);

fprintf("Metoda siecznych | dolna ");

freq\_secf(F, 0, 35000, precision);

fprintf("Metoda siecznych | górna ");

freq\_secf(F, 300000, 500000, precision);

fprintf("Metoda quasi-Newtona | dolna ");

freq\_qn(F, 0, 35000, 1e-7, precision);

fprintf("Metoda quasi-Newtona | górna ");

freq\_qn(F, 300000, 500000, 1e-7, precision);

end

%metoda bisekcji

function x = freq\_bisf(F, x0, x1, precision)

iterator = 0;

x = (x0 + x1) / 2;

while abs(F(x)-F(x1)) > precision

if (F(x) \* F(x0) < 0)

x1 = x;

else

x0 = x;

end

x = (x0 + x1) / 2;

iterator = iterator + 1;

end

fprintf("granica to %f. %d iteracji użyto \n",x,iterator);

end

%metoda siecznych

function x = freq\_secf(F, x0, x1, precision)

iterator = 0;

x = x1 - F(x1) \* (x1 - x0) / (F(x1) - F(x0));

while abs(F(x)) > precision

x = x1 - F(x1) \* (x1 - x0) / (F(x1) - F(x0));

x0 = x1;

x1 = x;

iterator = iterator + 1;

end

fprintf("granica to %f. %d iteracji użyto \n",x,iterator);

end

%metoda quasi-Newtona

function x = freq\_qn(F, x0,x1, delta, precision)

dif = differ(F,x0,delta);

if(F(x0)\*differ(F, dif,delta)>0)

x = x0;

else

x = x1;

end

iterator = 0;

while abs(F(x)) > precision

x = x - F(x) / differ(F, x, delta);

iterator = iterator + 1;

end

fprintf("granica to %f. %d iteracji użyto \n",x,iterator);

end

%obliczanie pochodnej

function dif = differ (F, x, delta)

dif = (F(x + delta) - F(x))/delta;

end

W skrypcie zaimplementowano 3 funkcje, które znajdują rozwiązanie zadanego problemu. Freq\_bisf implementuje metodę bisekcji, freq\_secf implementuje metodę siecznych, natomiast freq\_qn implementuje metodę quasi-Newtona, differ oblicza pochodną dla zadanych parametrów i jest używana przy metodzie quasi-Newtona.

Funkcja **find\_fc** przyjmuje cztery parametry R1, R2, C1, C2, odpowiadające wartościom, odpowiednio rezystancji i pojemności poszczególnych elementów zadanego obwodu.

Po wywołaniu funkcji z argumentami z części pierwszej:

find\_fc(30, 10, 0.1e-6, 0.2e-6)

Otrzymujemy odpowiedź w postaci zbiorczej informacji o zastosowanych metodach, częstotliwościach granicznych oraz liczbach iteracji.

|  |  |  |  |
| --- | --- | --- | --- |
| Metoda | Granica | Częstotliwość graniczna | Liczba iteracji |
| Metoda bisekcji | Dolna granica | 13811.806440 | 21 |
| Górna granica | 305659.675598 | 19 |
| Metoda siecznych | Dolna granica | 13811.813304 | 8 |
| Górna granica | 305659.830325 | 4 |
| Metoda quasi-Newtona | Dolna granica | 13811.801114 | 4 |
| Górna granica | 305659.838834 | 2 |
| Rozwiązanie analityczne | Dolna granica | 13811.8 | nd |
| Górna granica | 305660 | nd |

Najbardziej efektywną metodą dla poszukiwania górnej i dolnej granicy jest metoda quasi-Newtona, która w przypadku rozwiązywania równania zadanego w tej części wymagała 6 iteracji (łącznie) w porównaniu do 40 iteracji, których potrzebowała metoda bisekcji i 12 iteracji, których wymagała metoda siecznych.

W celu weryfikacji wyników znajdowane będzie rozwiązanie analityczne dla 5 różnie wybranych zestawów parametrów R1,R2,C1,C2, z różnymi przyjętymi przedziałami znajdowania rozwiązania oraz porównywane z wynikami otrzymanymi przez użycie skryptu.

Parametry: R1=20 R2=15 C1=0.2e-6 C2=0.2e-6

|  |  |  |  |
| --- | --- | --- | --- |
| Metoda | Granica | Częstotliwość graniczna | Liczba iteracji |
| Metoda bisekcji | Dolna granica | 8723.158836 | 20 |
| Górna granica | 241983.327866 | 20 |
| Metoda siecznych | Dolna granica | 8723.157422 | 4 |
| Górna granica | 241983.254680 | 4 |
| Metoda quasi-Newtona | Dolna granica | 8723.157293 | 4 |
| Górna granica | 241983.254733 | 4 |
| Rozwiązanie analityczne | Dolna granica | 8723.16 | nd |
| Górna granica | 241983 | nd |

Parametry: R1=1 R2=2 C1=0.2e-6 C2=0.2e-6

|  |  |  |  |
| --- | --- | --- | --- |
| Metoda | Granica | Częstotliwość graniczna | Liczba iteracji |
| Metoda bisekcji | Dolna granica | 208112.597466 | 20 |
| Górna granica | 1521429.967880 | 20 |
| Metoda siecznych | Dolna granica | 208112.660262 | 5 |
| Górna granica | 1521429.279769 | 5 |
| Metoda quasi-Newtona | Dolna granica | 208112.598728 | 4 |
| Górna granica | 1521430.481525 | 3 |
| Rozwiązanie analityczne | Dolna granica | 208113 | nd |
| Górna granica | 1521430 | nd |

Parametry: R1=10 R2=10 C1=0.3e-6 C2=0.2e-6

|  |  |  |  |
| --- | --- | --- | --- |
| Metoda | Granica | Częstotliwość graniczna | Liczba iteracji |
| Metoda bisekcji | Dolna granica | 11119.190454 | 21 |
| Górna granica | 379678.235054 | 20 |
| Metoda siecznych | Dolna granica | 11119.191782 | 4 |
| Górna granica | 379678.206946 | 5 |
| Metoda quasi-Newtona | Dolna granica | 11119.185294 | 3 |
| Górna granica | 379678.393720 | 7 |
| Rozwiązanie analityczne | Dolna granica | 11119.2 | nd |
| Górna granica | 379678 | nd |

Parametry: R1=7 R2=7 C1=0.7e-7 C2=0.7e-7

|  |  |  |  |
| --- | --- | --- | --- |
| Metoda | Granica | Częstotliwość graniczna | Liczba iteracji |
| Metoda bisekcji | Dolna granica | 72913.007736 | 19 |
| Górna granica | 1446915.626526 | 19 |
| Metoda siecznych | Dolna granica | 72913.032592 | 4 |
| Górna granica | 1446914.739245 | 5 |
| Metoda quasi-Newtona | Dolna granica | 72913.034568 | 2 |
| Górna granica | 1446914.745624 | 4 |
| Rozwiązanie analityczne | Dolna granica | 72913 | nd |
| Górna granica | 1446914 | nd |

Parametry: R1=4 R2=2 C1=0.2e-6 C2=0.2e-6

|  |  |  |  |
| --- | --- | --- | --- |
| Metoda | Granica | Częstotliwość graniczna | Liczba iteracji |
| Metoda bisekcji | Dolna granica | 42772.974968 | 21 |
| Górna granica | 1850634.574890 | 19 |
| Metoda siecznych | Dolna granica | 42772.978234 | 5 |
| Górna granica | 1850635.138607 | 4 |
| Metoda quasi-Newtona | Dolna granica | 42772.963857 | 2 |
| Górna granica | 1850635.432940 | 4 |
| Rozwiązanie analityczne | Dolna granica | 42773 | nd |
| Górna granica | 1850635 | nd |

# Część 3

function plot\_g(R1, R2, C1, C2, fmin, fmax)

F = @(x) 20 \* log10 (abs((x / (C2 \* R2)) / ...

(x ^ 2 + x \* (1 / (C1 \* R1) + 1 / (C1 \* R2) ...

+ 1 / (C2 \* R2)) + 1 / (C1 \* C2 \* R1 \* R2))));

x = logspace (log10(fmin), log10(fmax));

z = 2i \* pi \* x;

y = zeros(1, length(x));

for i = 1 : length(x)

y(i) = F(z(i));

end

fc1=13811.801114;

fc2=305659.838834;

thr = max(y) - 3;

semilogx(x, y);

ylabel('Amplituda[dB]');

xlabel('f[Hz]');

hold on

yline(thr,'-.','-3db');

xline(fc1,'--','fc1');

xline(fc2,'--','fc2');

semilogx(fc1,thr,'or');

semilogx(fc2,thr,'or');

grid on

end

Żeby narysować wykres z wartościami z części pierwszej instrukcji, wywołuję funkcje z parametrami:

plot\_g (30,10,0.1e-6,0.2e-6,10,10e6)

Wywołanie programu z podanymi parametrami generuje wykres:

|  |
| --- |
|  |
| Charakterystyka transmitancji obwodu (linia niebieska). Poziom -3dB zaznaczony linią poziomą.  fc1 i fc2 - wartości częstotliwości granicznych obliczonych metodą quasi-Newtona |

W celu weryfikacji wyników przedstawiam wykres w skupieniu na punktach, w których linia pozioma tłumienia -3dB przecina się z punktami częstotliwości granicznych obliczonych metodą quasi-Newtona.

|  |  |
| --- | --- |
|  |  |
| Częstotliwość graniczna dolna (powiększenie) | Częstotliwość graniczna górna (powiększenie) |

Jak widać na powyższych wykresach, wyniki obarczone są błędami. Powodem rozbieżności jest błąd jakim obarczona jest metoda quasi-Newtona jak również przyjęta metoda graficznej weryfikacji wyników.

# Część 4

W tej części będzie wyznaczana całkowita moc chwilowa wydzielającą się na elementach pasywnych układu. W tym celu będą obliczane całki metodą złożoną prostokątów lewych i metodą Simpsona (złożoną parabol) na podstawie danych obliczonych przy pomocy metody Eulera z zadania 1. W celu prezentacji skryptu przedstawiona jest jedynie dodatkowa część realizująca niezbędne metody, oraz obliczająca wartości dP dla wszystkich przekazanych z metody Eulera wartości napięć u1 i u2. Funkcja obliczająca całkę, podobnie jak funkcja z części 1 wywoływana jest z parametrem oznaczającym częstotliwość wymuszenia.

...

%obliczanie wartości dP ze wzoru

for i=1 : length(t)

dP(i) = (e(t(i))-u(1,i))^2/R1 + ...

(e(t(i))-u(1,i)-u(2,i))^2/R2;

end

parabole = int\_simps (t,h,dP);

prostokaty = int\_rect (t,h,dP);

fprintf('Metoda prostokatów: %e \n\n',prostokaty);

fprintf('Metoda Simpsona: %e \n\n',parabole);

...

%złożona metoda parabol (Simpsona)

function calka = int\_simps (t,h,df)

for i = 1 : 2 : length(t)-2

simpson((i + 1) / 2) = h/3\*(df(i)+4\*df(i+1)+df(i+2));

end

calka = sum(simpson);

end

%złożona metoda prostokątów lewych

function calka = int\_rect (t,h,df)

for i = 1 : length(t)-1

dfdx(i) = df(i) \* h;

end

calka = sum(dfdx);

end

...

Dla obliczeń, jako bardzo krótki przyjęto krok Δt1=1e-8, ze względu na czas potrzebny na obliczanie przy trwającym sekundę przedziale czasowym, oraz krok bardzo długi Δt2=1e-6, przy którym otrzymujemy matematycznie poprawne wyniki. W skrypcie zmienną odpowiadająca za ustawianie Δt jest h.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Wymuszenie | Metoda złożona prostokątów | | Metoda złożona parabol | |
| Δt1=1e-8 | Δt2=1e-6 | Δt1=1e-8 | Δt2=1e-6 |
| e(t) = 1 V | 5.033620e-08 | 3.000000e-07 | 4.966571e-08 | 2.268427e-07 |
| e(t) = sin(2πft), dla f = 50 Hz | 1.480386e-08 | 1.480389e-08 | 1.480386e-08 | 1.480391e-08 |
| e(t) = sin(2πft), dla f = 5000 Hz | 1.357785e-04 | 1.368422e-04 | 1.357785e-04 | 1.368424e-04 |
| e(t) = sin(2πft), dla f = 50 kHz | 3.155441e-03 | 3.358593e-03 | 3.155441e-03 | 3.358600e-03 |
| e(t) = sin(2πft), dla f = 65 kHz | 4.552155e-03 | 4.947800e-03 | 4.552155e-03 | 4.947810e-03 |
| e(t) = sin(2πft), dla f = 200 kHz | 2.380168e-02 | 4.655737e-02 | 2.380168e-02 | 4.655743e-02 |
| E=1 dla t<T/2 i E=0 dla t>=T/2 dla T=0.05ms, | 1.931467e-03 | 1.191418e-02 | 1.929345e-03 | 1.093318e-02 |

W tabeli powyżej przedstawione są obliczone wartości całki metodami: prostokątów i parabol dla przyjętych różnych kroków.

# Część 5

W tym ćwiczeniu elementem obciążającym filtr jest element nieliniowy, którego charakterystyka przetwarzania została podana w tabeli. W poniższym skrypcie przedstawione będą implementacje interpolacji wielomianowej z zastosowaniem macierzy Vandermonde’a, aproksymacji n stopnia, oraz interpolacji naturalnym splajnem kubicznym. Jako, że jest to kompletna wersja używanego skryptu, zawierająca zarówno metodę Simpsona do obliczania mocy jak i metodę Eulera, która jest zmieniona ze względu na zmianę w tej części zadania, to przedstawiam całą (wraz z dodatkowymi implementacjami dla tej części) poniżej.

function nonlinear()

R1 = 30;

R2 = 10;

C1 = 0.1e-6;

C2 = 0.2e-6;

h = 1e-7;

t = [ 0 : h : 0.05];

freq = 1000;

e = @(t) sin(2\*pi\*t\*freq);

un = [-1 -0.75 -0.5 -0.25 0 0.25 0.5 0.75 1];

in = [0.01 -0.01 0.02 0.01 0 0.23 0.42 0.6 0.95];

c\_aprox3 = aprox(un, in, 3);

c\_aprox5 = aprox(un, in, 5);

diff2 = difmatrix(un, in, 0.25);

i\_inter = @(u) ctv(interpolate(un, in), u);

i\_aprox3 = @(u) ctv(c\_aprox3, u);

i\_aprox5 = @(u) ctv(c\_aprox5, u);

i\_spline = @(u) splineit(un, in, diff2, u);

dy = @(t,y,In) ...

[ 1/C1 \* ( (e(t) - y(1) - y(2))/R2 + (e(t) - y(1))/R1 )

1/C2 \* ( (e(t) - y(1) - y(2))/R2 - In )];

u = euler(t, h, dy, i\_spline);

%for v=1 : length(t)

% dP(v) = ((e(t(v))-u(1,v)-u(2,v))\*i\_inter(u(2,v)));

%end

%p\_nonlinear = int\_simps(t,h,dP);

%fprintf('Moc wydzielana na rezystorze to %f', p\_nonlinear);

plot(t, u(2,:));

%interpolacja

function A = interpolate(X, Y)

M = vandermonde(X);

A = M \ Y';

end

%aproksymacja stopnia n

function A = aprox(X, Y, n)

N = length(X);

M = zeros(N,n-1);

for w = 1:N

for k = 1:n+1

M(w, k) = X(w) ^ (k - 1);

end

end

MTM = M'\*M;

MTY = M'\*Y';

A = MTM \ MTY;

end

%macierz Vandermonde'a

function M = vandermonde(X)

N = length(X);

for w = 1:N

for k = 1:N

A(w, k) = X(w) ^ (k - 1);

end

end

M = A;

end

%na podstawie macierzy współczynników i x wylicza wartość wielomianu W(x)

function y = ctv(A, x)

s = 0;

for i = 1:length(A)

s = s + A(i) \* x ^ (i - 1);

end

y = s;

end

%znajduje do którego sektora należy x do spline'a

function sect = sector(x,X)

N = length(X);

R = N;

L = 1;

while 1

if(R-L) <= 1

sect = L;

break;

end

sect = fix((L+R)/2);

if x < X(sect)

R = sect;

else

L = sect;

end

end

end

%oblicza macierz drugich pochodnych do spline'a

function diff2 = difmatrix(X,Y,h)

N = length(X);

M = diag(ones(1, N - 3), 1) + diag(4 \* ones(1, N - 2))...

+ diag(ones(1, N - 3), - 1);

for i = 1:N - 2

z(i) = (6 / h ^ 2) \* (Y(i + 2) ...

- 2 \* Y(i + 1) + Y(i));

end

diff2 = [0; M \ z'; 0];

end

%oblicza wartość spline'a w punkcie x

function y = splineit(X,Y,diff2,x)

p = sector(x,X);

h = X(p)-X(p+1);

y = ((x - X(p+1))^3/h - (x - X(p+1))\*h)\*diff2(p)/6 ...

- ((x - X(p))^3/h - (x - X(p))\*h)\*diff2(p+1)/6 ...

+ Y(p)\*(x - X(p+1))/h - Y(p+1)\*(x - X(p))/h;

end

%metoda Eulera

function y = euler(t,h,f,In)

y = [0 0]';

for i = 1 : length(t)-1

y(:, i+1) = y(:, i) + h \* ...

f(t(i), y(:, i), In(y(2,i)));

end

end

%złożona metoda parabol (Simpsona)

function calka = int\_simps (t,h,df)

for i = 1 : 2 : length(t)-2

simpson((i + 1) / 2) = h/3\*(df(i)+4\*df(i+1)+df(i+2));

end

calka = sum(simpson);

end

end

Wywołanie powyższego skryptu, z różnie zmienionymi wartościami zmiennej freq, oraz przy zmianie w wywołaniu funkcji euler, na odpowiednio: *i\_inter* dla interpolacji wielomianowej, *i\_aprox3* dla aproksymacji 3 stopnia, *i\_aprox5* dla aproksymacji 5 stopnia, *i\_spline* dla splajna kubicznego generuje poniższe wykresy:

|  |  |
| --- | --- |
|  |  |
| Przebieg napięcia u2 wymuszenia stałego  e(t) = sin(2πft), przy założeniu f = 1000 Hz,  obciążenie elementem nieliniowym, interpolacja wielomianowa | Przebieg napięcia u2 wymuszenia stałego  e(t) = sin(2πft), przy założeniu f = 1000 Hz,  obciążenie elementem nieliniowym, aproksymacja wielomianem 3 stopnia |
|  |  |
| Przebieg napięcia u2 wymuszenia stałego  e(t) = sin(2πft), przy założeniu f = 1000 Hz,  obciążenie elementem nieliniowym, aproksymacja wielomianem 5 stopnia | Przebieg napięcia u2 wymuszenia stałego  e(t) = sin(2πft), przy założeniu f = 1000 Hz,  obciążenie elementem nieliniowym, interpolacja splajnem kubicznym |
|  |  |
| Przebieg napięcia u2 wymuszenia stałego  e(t) = sin(2πft), przy założeniu f = 65000 Hz  obciążenie elementem nieliniowym, interpolacja wielomianowa | Przebieg napięcia u2 wymuszenia stałego  e(t) = sin(2πft), przy założeniu f = 65000 Hz  obciążenie elementem nieliniowym, aproksymacja wielomianem 3 stopnia |
|  |  |
| Przebieg napięcia u2 wymuszenia stałego  e(t) = sin(2πft), przy założeniu f = 65000 Hz | Przebieg napięcia u2 wymuszenia stałego  e(t) = sin(2πft), przy założeniu f = 65000 Hz |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Interp. Wiel. | Interp. func. sklej. | Aproks. Wiel. n=3 | Aproks. Wiel. n=5 |
| f=65 kHz |  |  |  |  |
| f=1 kHz |  |  |  |  |