SPRAWOZDANIE 4

Zajęcia nr 5 - Analiza harmoniczna (cz. II) - Michał Midor, gr. 5 - 29.10.2025, godz: 16:45

Celem tych zajęć było nauczenie się rekonstrukcji sygnału za pomocą współczynników zespolonych, obliczanie wartości skutecznych oryginału sygnału i jego rekonstrukcji oraz obliczanie współczynnika zniekształceń harmonicznych THD.

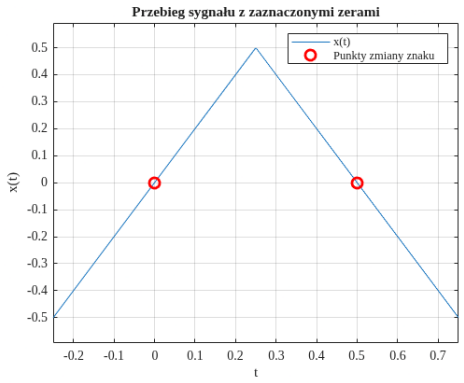
Parametr THD jest bardzo ważny ze względu na to, że przekazuje on informację o tym, jak "brudny" jest sygnał wyjściowy w porównaniu do idealnej sinusoidy. Celem w przekazywaniu sygnału jest jego jak najmniejsze zniekształcenie przez harmoniczne częstotliwości podstawowej i dostarczanie stałej częstotliwości bazowej.

Rozwiązanie zadań:

Zad 1-4 tym razem podaję bez komentarza, służą one jedynie jako podstawa do zadań 5-7.

1)

```
clear all; close all;
syms t t1 t2 offset x
T0 = 1.0; % Okres
t1 = -0.5;
t2 = t1+T0; % t2 = 0.5
offset = T0/4; % offset = 0.25
f0 = 1/T0; % Czestotliwosc
w0 = 2*pi*f0; % Pulsacja
BND = [t1,t2] + offset;
x = triangularPulse(t1,0,t2,t-offset)-0.5;

t_zero_in_boundaries = [0.0, 0.5];
x_zero_value = [0, 0];
figure;
ezplot(x, BND);
grid on;
ylabel('x(t)')
xlabel('t')
title('Przebieg sygnału z zaznaczonymi zerami');
hold on;
plot(t_zero_in_boundaries, x_zero_value, 'ro', 'MarkerSize', 8, 'LineWidth',
2);
legend('x(t)', 'Punkty zmiany znaku', 'Location', 'NorthEast');
hold off;
```

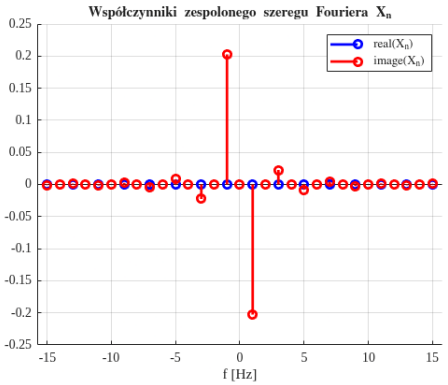


2)

```
syms n
NT = 15;
X=[];
ind = -NT : NT;
for n = ind
 Xn = (1/T0)*int(x*exp(-1i*w0*n*t),t,BND); % bezpieczniej użyć -1i zamiast -i
 X(n + NT + 1) = Xn;
end
X_numeric = double(X)
```

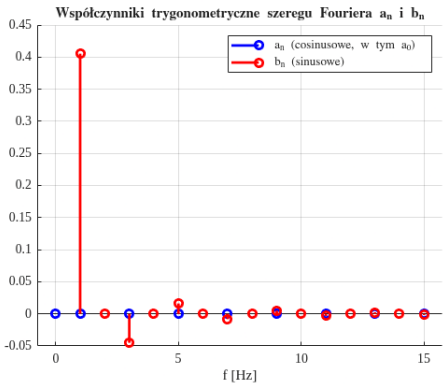
```
X_numeric = 1×31 complex
   0.0000 - 0.0009i   0.0000 + 0.0000i   0.0000 + 0.0012i   0.0000 + 0.0000i ···
```

```
figure; hold on;
stem(ind*f0,real(X_numeric),'b','LineWidth',2);
xlabel('f [Hz]')
stem(ind*f0,imag(X_numeric),'r','LineWidth',2);
grid on
legend('real(X_n)','image(X_n)','Location','NorthEast')
title('Współczynniki zespolonego szeregu Fouriera X_n')
hold off
```



3)

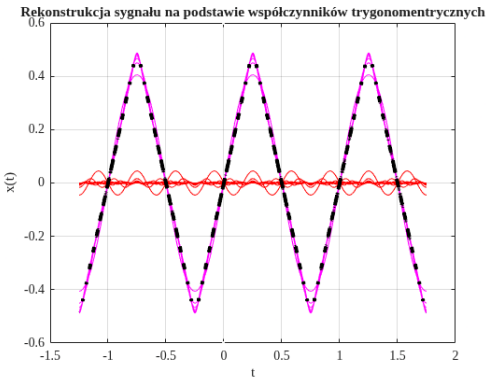
```
n_ind = 1:NT;
a0 = (1/T0)*int(x, t, BND);
a = zeros(1, NT + 1);
b = zeros(1, NT + 1);
a(1) = a0;
for k = n_ind
 n = k;

 an = (2/T0) * int(x * cos(n * w0 * t), t, BND);
 a(k + 1) = an;

 bn = (2/T0) * int(x * sin(n * w0 * t), t, BND);
 b(k + 1) = bn;
end
a_numeric = double(a);
b_numeric = double(b);
figure;
hold on;
f_ind = (0:NT)*f0;
stem(f_ind, a_numeric, 'b', 'LineWidth', 2);
stem(f_ind(2:end), b_numeric(2:end), 'r', 'LineWidth', 2);
xlabel('f [Hz]');
title('Współczynniki trygonometryczne szeregu Fouriera a_n i b_n');
legend('a_n (cosinusowe, w tym a_0)', 'b_n (sinusowe)', 'Location',
'NorthEast');
grid on;
hold off;
```



4)

```
step = (BND(2) - BND(1))/1000;
tt = [BND(1)-T0 : step: BND(2) + T0];
```

```
xx = zeros(1,length(tt));
xx = xx + a(1); % skladowa stala

figure
plot(tt,xx,'m'); grid on, hold on;
plot([0,0],[-0.6,0.6],'w.')
xlabel('t'); ylabel('x(t)');
pause(0.5)

for n = 1 : NT
 xx_n = (a_numeric(n+1)*cos(w0*n*tt) + b_numeric(n+1)*sin(w0*n*tt));
 xx = xx + xx_n;
 plot(tt,xx_n,'r'); plot(tt,xx,'m');
 title(sprintf('n = %d',n+1)); pause(0.5)
end
plot(tt,xx,'k','LineWidth',3);
title('Rekonstrukcja sygnału na podstawie współczynników
trygonomentrycznych');
```



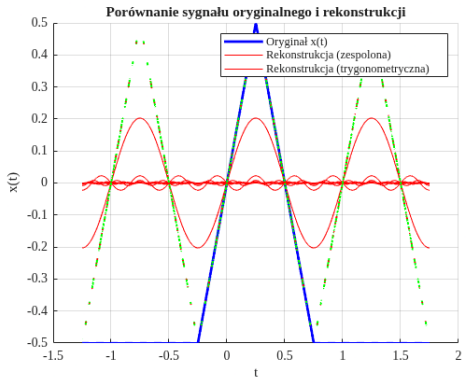

```

```

5)

Celem tego zadania jest zrekonstruowanie oryginalnego sygnału x(t), ale tym razem na podstawie współczynników zespolonych, nie trygonometrycznych jak poprzednio.

```
xx_zespolony = zeros(1, length(tt));
x_original_plot = double(subs(x, t, tt));


figure;
hold on;
plot(tt, x_original_plot, 'b', 'LineWidth', 2); % Oryginał


for n_idx = 1:length(ind)
    n = ind(n_idx);
    Xn = X_numeric(n_idx);
```

```
    harmoniczna_n = Xn * exp(1i*n*w0*tt);
    xx_zespolony = xx_zespolony + harmoniczna_n;

    plot(tt, real(harmoniczna_n), 'r');
end

xx_zespolony_real = real(xx_zespolony);


plot(tt, xx_zespolony_real, 'r--', 'LineWidth', 1.5); % Nowy (zespolony)
plot(tt, xx, 'g:', 'LineWidth', 1.5); % Stary (trygonometryczny)
grid on;
xlabel('t');
ylabel('x(t)');
title('Porównanie sygnału oryginalnego i rekonstrukcji');
legend('Oryginał x(t)', 'Rekonstrukcja (zespolona)', 'Rekonstrukcja
(trygonometryczna)');
hold off;
```




```
```

Obliczenie błedu aproksymacji:

```
blad_sygnal = x_original_plot - xx_zespolony_real;

mse_error = mean(blad_sygnal.^2);

fprintf('Błąd średniokwadratowy (MSE) dla NT=%d wynosi: %e\n', NT,
mse_error);
```

```
Błąd średniokwadratowy (MSE) dla NT=15 wynosi: 2.221471e-01
```

Zwiększanie liczby elementów szeregu zmniejsza błąd przybliżenia.

6)

W tym zadaniu celem jest obliczenie numerycznie wartości skutecznej dla sygnału sinusoidalnego, a następnie numerycznie dla oryginalnego sygnału x(t) oraz jego rekonstrukcji z zadania 4 (czyli trygonometrycznej).

```
syms A w0 t;
```

```
T0_sin = 2*pi/w0;

signal_squared = (A*sin(w0*t)).^2;
mean_signal_squared = (1/T0_sin)*int(signal_squared, t, 0, T0_sin);
rms_signal_symbolic = sqrt(mean_signal_squared)
```

```
rms_signal_symbolic =
```

$$\sqrt{\frac{A^2}{2}}$$

```
skd = double(subs(rms_signal_symbolic, A, 1.0)) % Wynik jest poprawny.
```

```
skd =
0.7071
```

Wartość skuteczna sygnału x(t).

```
x_squared = x.^2;
mean_x_squared = (1/T0) * int(x_squared, t, BND(1), BND(2));
rms_x_symbolic = sqrt(mean_x_squared);

x_rms = double(rms_x_symbolic);
fprintf('Wartość skuteczna sygnału x(t) wynosi: %f\n', x_rms);
```

```
Wartość skuteczna sygnału x(t) wynosi: 0.288675
```

Wartość skuteczna sygnału zrekonstruowanego (z ćw. 4).

Liczenie RMS z wektora numerycznego xx to tylko przybliżenie zależne od kroku. Skorzystam z Twierdzenia Parsevala: "całkowita moc sygnału jest równa sumie mocy wszystkich jego składowych harmonicznych".

```
% Moc składowej stałej (n=0)
P_0 = a_numeric(1)^2;

% Moce harmonicznych (n=1 do NT)
P_n = sum( (a_numeric(2:end).^2 + b_numeric(2:end).^2) / 2 );

P_c = P_n + P_0;

rms_xx = sqrt(P_c);
fprintf('Wartość skuteczna sygnału zrekonstruowanego (z %d harm.) wynosi:
%f\n', NT, rms_xx);
```

```
Wartość skuteczna sygnału zrekonstruowanego (z 15 harm.) wynosi: 0.288669
```

7)

Celem tego zadania jest wyznaczenie wartości numerycznej THD (współczynnik całkowitych zniekształceń harmonicznych).

```
P_podst = a_numeric(2)^2 + b_numeric(2)^2;
W_skut_podst = sqrt(P_podst)
```

```
W_skut_podst =
0.4053
```

```
N_values = [5, 10, 15];
THD_results = [];

for NT = N_values
    P_harmonic = sum(a_numeric(3:NT+1).^2 + b_numeric(3:NT+1).^2);
    W_skut_harmonic = sqrt(P_harmonic);

    THD = W_skut_harmonic/W_skut_podst;
    THD_results(end+1) = THD;

    fprintf('Dla NT = %d, THD = %f (czyli %.2f%%)\n', NT, THD, THD*100);
end
```

```
Dla NT = 5, THD = 0.118092 (czyli 11.81%)
Dla NT = 10, THD = 0.120477 (czyli 12.05%)
Dla NT = 15, THD = 0.120986 (czyli 12.10%)
```

Im więcej harmonicznych, tym współczynnik zniekształceń wywołany przez nie jest większy.