| Mariusz Więcławek | Projekt:<br>Zastosowanie DCT do aproksymacji funkcji dla zadanej liczby harmonicznych |
|---|---|

Zadaniem projektowym jest aproksymacja danej funkcji przy pomocy dyskretnej transformacji kosinusowej DCT dla zadanej liczby harmonicznych.

## 1. Wstęp teoretyczny.

Transformacja kosinusowa jest metodą transformacji danych. Dla określenia transformacji stosowany jest często anglojęzyczny akronim DCT (ang. Discrete Cosine Transform). DCT to liniowa odwracalna funkcja $f : R^N \to R^N$ przekształcająca ciąg liczb x(0), x(1), ..., x(N −1) w inny ciąg liczb s(0), s(1), ..., s(N −1). Dyskretna transformata kosinusowa wykorzystuje rozwinięcie sygnału w bazie ortogonalnych wielomianów Czebyszewa:

$$\left\{ \frac{1}{\sqrt{N}}, \sqrt{\frac{2}{N}} \cos \frac{\pi k(2n+1)}{2N} \right\}, \qquad k, n = 1, 2, ..., N-1,$$

a macierz dyskretnego przekształcenia kosinusowego zbudowana jest ze zdyskretyzowanych wielomianów Czebyszewa.

***Definicja 1***: Dyskretną transformacją kosinusową (ang. Discrete Cosine Transform) rzeczywistych wartości sygnału **x(n)**, określonych dla **n = 0, 1, ..., N-1**, nazywamy ciąg rzeczywistych współczynników rozwinięcia sygnału **x(n)**:

$$s(k) = c(k) \cdot \sum_{n=0}^{N-1} x(n) \cos\left(\frac{\pi k(2n+1)}{2N}\right), \qquad k = 0, 1, ..., N-1,$$

$$gdzie\!: \; c(0) = \sqrt{1/N}, \; c(k) = \sqrt{2/N} \; dla \; k > 0.$$

***Definicja 2:*** Dyskretną odwrotną transformacją kosinusową (ang. Inverse Discrete Cosine Transform) jest odwzorowanie, które na podstawie widma sygnału s(k), k = 0, ..., N − 1, odtwarza próbki sygnału pierwotnego x(n), n = 0, ..., N − 1:

$$x(n) = \sqrt{\frac{1}{N}} s(0) + \sqrt{\frac{2}{N}} \cdot \sum_{k=1}^{N-1} s(k) \cos\left(\frac{\pi k(2n+1)}{2N}\right), \quad n = 0, 1, ..., N-1.$$

Z definicji transformacji kosinusowej wynika, że jądro transformacji jest takie samo, jak jądro transformacji odwrotnej, co ułatwia organizację obliczeń numerycznych. Macierz jądrowa transformacji ma postać:

$$\boldsymbol{W}_c = [w_{k,n}]_{N\times N} = c(k) \cos\left(\frac{\pi k(2n+1)}{2N}\right),$$

gdzie:

$$c(k) = \begin{cases} \sqrt{1/N} & \text{dla} \quad k = 0 \\ \sqrt{2/N} & \text{dla} \quad k > 0 \end{cases}.$$

Tym samym można ją zapisać w formie macierzowej:

$$\boldsymbol{s} = \boldsymbol{x} \cdot \boldsymbol{W}_c.$$

Z ortogonalności macierzy **Wc** wynika, że:

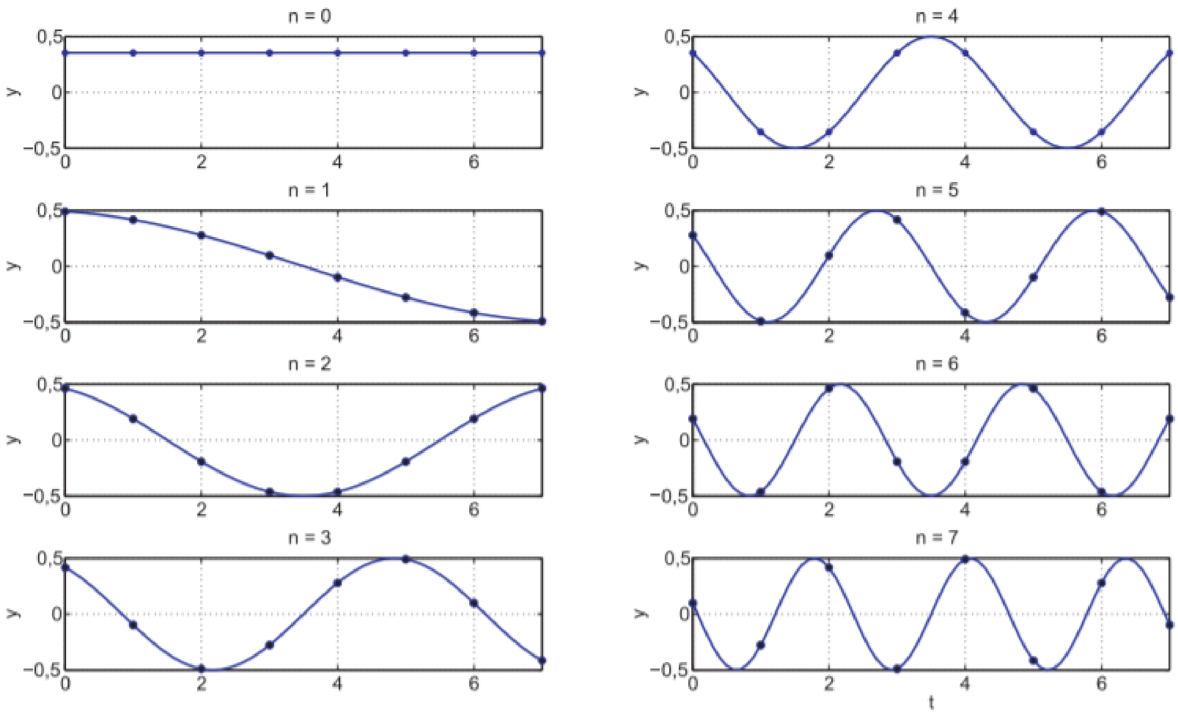
$$\boldsymbol{W}_c^{-1} = \boldsymbol{W}_c^T \ ,$$

co ułatwia przeprowadzenie przekształcenia odwrotnego, gdyż macierz odwrotna jest tutaj równa macierzy transponowanej, tę zaś można uzyskać w prosty sposób:

$$\boldsymbol{x} = \boldsymbol{s} \cdot \boldsymbol{W}_c^{-1} = \boldsymbol{s} \cdot \boldsymbol{W}_c^T.$$

Rysunek poniżej przedstawia pierwszych osiem funkcji bazowych stosowanych w transformacji kosinusowej z zaznaczonymi punktami wartości dyskretnych. Wartości elementów macierzy Wc można odnaleźć na wykresach funkcji bazowych. Można także zauważyć, że wraz ze wzrostem n wzrasta częstotliwość przebiegu kolejnych funkcji.

Wykres pierwszych $N = 8$ funkcji bazowych transformacji kosinusowej $y = c(n)\cos(\frac{\pi n(2t+1)}{2N})$ wraz z punktami reprezentacji dyskretnej



## 2. Opis kluczowych fragmentów kodu.

==================================================================

```
v=zeros(N);
v(1,:)=ones(N,1)*sqrt(1/N);
for k=2:N
    for n=1:N
        v(k,n)=sqrt(2/N)*cos(((k-1)*(n+1/2)*pi)/N);
    end
end
```

W tym fragmencie kodu tworzymy jądro transformacji korzystając z zależności przedstawionych we wstępie teoretycznym. Pierwszy wiersz naszej macierzy jądra jest równy $\sqrt{\frac{1}{N}}$, z kolei następne wiersze opisane są następującą zależnością $\sqrt{\frac{2}{N}} * \cos\left[\frac{\pi k(n+\frac{1}{2})}{2}\right]$

==================================================================

```
yt=y*v';
```

W celu aproksymacji przekształcamy funkcje wejściową względem macierzy jądra transformacji.

==================================================================

```
yn=ones(N,1)*sqrt(1/N)*yt(1);
for i=1:N
    for k=2:10
        yn(i,1)=yn(i,1)+sqrt(2/N)*cos(((k-1)*(i+1/2)*pi)/N)*yt(k);
    end
end
```

Następnie korzystamy z definicji transformacji kosinusowej opisanej zależnością:

$$s(k) = c(k) \cdot \sum_{n=0}^{N-1} x(n) \cos\left(\frac{\pi k(2n+1)}{2N}\right), \qquad k = 0, 1, ..., N-1,$$

$$gdzie\colon\ c(0) = \sqrt{1/N},\ c(k) = \sqrt{2/N}\ dla\ k > 0.$$

gdzie x(n) to nasz sygnał wejściowy czyli yt. Funkcja yn jest opisana dla 10 harmonicznych.

==================================================================

```
e1 = abs(yn - y');
```

Następnie obliczamy błąd wyznaczenia aproksymacji, czyli jak bardzo zaproksymowana funkcja yn różni się od naszej funkcji wejściowej y.

==================================================================

==================================================================

```
xn=ones(1,N)*sqrt(1/N)*yt(1);
for n=1:N
    for k=2:N
        xn(1,n)=xn(1,n)+sqrt(2/N)*cos(((k-1)*(n+1/2)*pi)/N)*yt(k);
    end
end
```

Następnie w celu pokazania, że obliczenia zostały wykonane poprawnie obliczamy transformatę odwrotną IDCT. Korzystamy z zależności opisanych we wstępie teoretycznym.

==================================================================

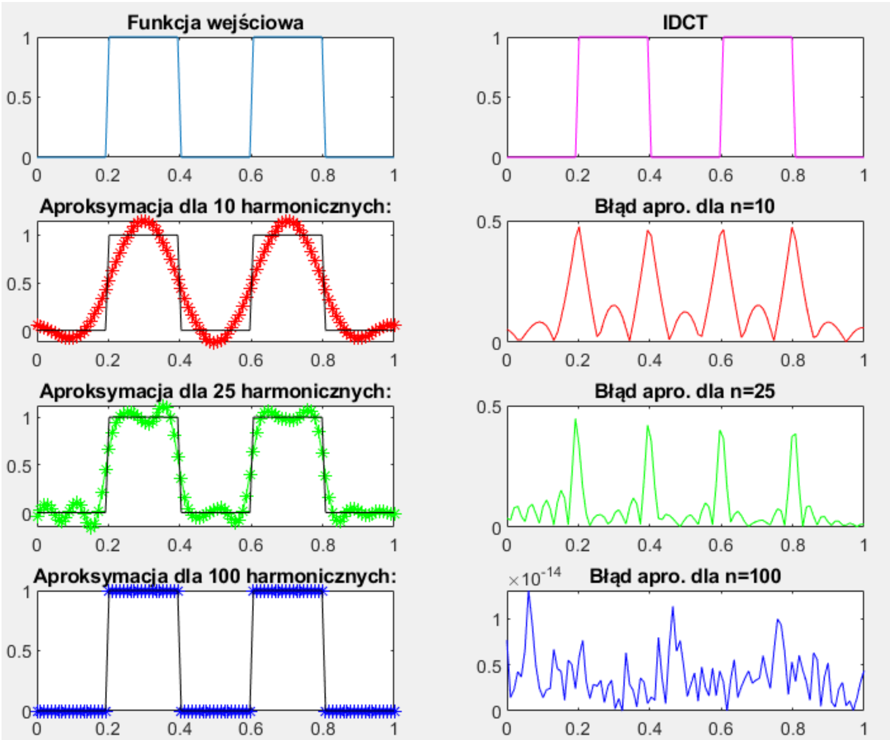
```
for u=3:N
    ys=ones(N,1)*sqrt(1/N)*yt(1);
    for i=1:N
        for k=2:u
            ys(i,1)=ys(i,1)+sqrt(2/N)*cos(((k-1)*(i+1/2)*pi)/N)*yt(k);
        end
    end

    MSE=0;
    for n=1:1:N
        MSE = MSE + (1/n)*(y(n)-ys(n))^2;
    end

    MSE_suma(1,(u-2))=MSE;
end
```

W celu pokazania jak zmienia się wartość błędu średniokwadratowego MSE wraz ze wzrostem harmonicznych obliczamy wartość MSE dla każdej z harmonicznych.

==================================================================

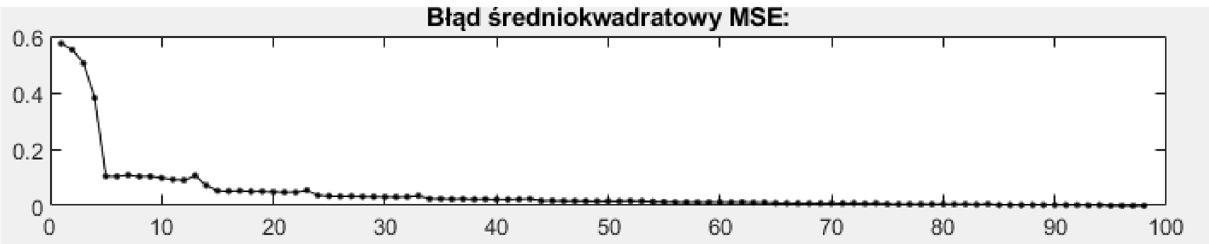
## 3. Obserwacje oraz wnioski.

***Rys.1***




Przeprowadzając symulacje naszego programu widzimy jak aproksymowana jest funkcja wejściowa dla różnych wartości harmonicznych. Widoczne jest wpływ tych harmonicznych tj. wraz ze wzrostem ich ilości funkcja jest aproksymowana dokładniej. To znaczy, że im więcej tych harmonicznych tym przebieg wyjściowy będzie lepiej obrazować przebieg wejściowy określony funkcja prostokątną.
Po prawej stronie powyższych schematów przedstawiliśmy błędy aproksymacji dla zadanych harmonicznych, można z nich odczytać jak duże są błędy dla określonego argumentu funkcji. Widać, że błędy te mają największa wartość w momencie kiedy przebieg prostokątny zmienia wartość z 0 na 1 czy tez na odwrót. Aproksymacja nie nadąża zbyt dobrze dla szybkiej zmiany wartości przebiegu wejściowego. Analizując wartości poszczególnych błędów dla 3 podanych wartości harmonicznych możemy dojść do wniosku, że błąd aproksymacji maleje wraz ze wzrostem harmonicznych. Lepiej przedstawi nam to wykres błędu średniokwadratowego MSE:

***Rys.2***




Jak można odczytać z tego wykresu błąd zmienia się zgodnie z naszymi założeniami powyżej, tj. wraz ze wzrostem harmonicznych maleje błąd aproksymacji. Błąd ten jest duży dla niewielkiej ilości harmonicznych, lecz wraz z ich wzrostem stopniowo maleje. Dla dużych wartości harmonicznych błąd ten jest bardzo niewielki.