

Instrukcja: Generowanie fali trójkątnej

Krok 1: Zdefiniowanie funkcji

Utwórz funkcję `triangular_wave``, która przyjmuje trzy parametry:

- `k_max`` – maksymalny rząd harmonicznej (int), określający liczbę składników sumy.
- `omega`` – częstotliwość sygnału (float).
- `x`` – argument funkcji (float), odpowiadający chwilowemu czasowi lub pozycji.

Krok 2: Inicjalizacja zmiennej sumującej

Zainicjalizuj zmienną `result`` jako `0``, ponieważ będzie ona przechowywać sumę kolejnych składników serii Fouriera.

Krok 3: Iteracja po rzędach harmoniczych

Użyj pętli `for``, aby iterować `k`` od `0`` do `k_max``. W każdej iteracji:

1. Oblicz składnik szeregu Fouriera dla aktualnej wartości `k``:

- `(-1)**k`` – wprowadza zmianę znaku, co odpowiada naprzemiennym wartościom funkcji.
 - `np.sin((2*k + 1) * omega * x)`` – oblicza wartość funkcji sinus dla nieparzystych wielokrotności częstotliwości `omega``.
 - Podziel przez `(2*k + 1)**2``, aby uwzględnić malejący wpływ kolejnych harmoniczych.
2. Dodaj obliczoną wartość do `result``.

Krok 4: Skalowanie wyniku

Pomnóż wynik przez `(8 / np.pi**2)``, aby uzyskać prawidłową amplitudę fali trójkątnej.

Krok 5: Zwrócenie wyniku

Zakończ funkcję zwracając obliczoną wartość `result``.

Podsumowanie

Implementacja opiera się na rozwinięciu fali trójkątnej w szereg Fouriera, który wykorzystuje tylko harmoniczne o nieparzystych indeksach. Wzór użyty w kodzie odpowiada klasycznemu rozwinięciu fali trójkątnej:

$$f(x) = (8 / \pi^2) \sum (-1)^k * \sin((2k+1) \omega x) / (2k+1)^2$$

To rozwinięcie pozwala aproksymować falę trójkątną z dowolną dokładnością w zależności od wartości `k_max``.