

1. Macierz odpowiedzi impulsowej (2 pkt)

Wykorzystując odpowiedź impulsową \mathbf{h} o długości $L=32$ (załadowaną z pliku `h.mat`), utwórz macierz \mathbf{H} o strukturze Toeplitza według zależności podanej na wykładzie (wykład 2, str. 4). Następnie wykonaj splot sygnału \mathbf{u} z odpowiedzią impulsową \mathbf{h} za pomocą mnożenia macierzowego:

$$\begin{bmatrix} y_1 \\ \vdots \\ y_{L+N} \end{bmatrix} = \begin{bmatrix} h_0 & 0 & \cdots & 0 \\ h_1 & h_0 & \ddots & \vdots \\ \vdots & h_1 & \ddots & 0 \\ h_{L-1} & \ddots & \ddots & h_0 \\ 0 & h_{L-1} & \ddots & h_1 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & h_{L-1} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix}, \quad \mathbf{y} = \mathbf{H} \mathbf{x}$$

dla trzech różnych wektorów \mathbf{u} zdefiniowanych następująco:

$$\mathbf{x}^{(1)} = [1, 0, \dots, 0_L]^T$$

$$\mathbf{x}^{(2)} = [0, 1, 1, 0.5, 0.5, -1, -1, 0, 0, \dots, 0_L]^T$$

$$\mathbf{x}_n^{(3)} = \sin(2\pi 2n/L) + \sin(2\pi 6n/L), \quad \text{gdzie } n=0, 1, \dots, L-1$$

Interpretacja wyników: \mathbf{h} to odpowiedź impulsowa filtru (układu), $\mathbf{x}^{(1)}$ to „impuls” (formalnie delta Kroneckera), tak więc, z definicji, filtrując sygnał impulsowy otrzymujemy „odpowiedź impulsową” filtru (układu). Sprawdź czy $\mathbf{y}^{(1)}$ jest takie samo jak \mathbf{h} .

Sygnał $\mathbf{x}^{(2)}$ można potraktować jako próbę przesłania danych, kolejno wartości: 0, 1, 1, 0.5, 0.5, -1, -1, 0, 0, ..., 0, przez kanał komunikacyjny o charakterystyce (odpowiedzi impulsowej) \mathbf{h} . Wtedy $\mathbf{y}^{(2)}$ można potraktować jako sygnał zarejestrowany przez odbiornik. Wyświetl na jednym wykresie sygnał nadany i odebrany. Czy możliwe jest zdekodowanie wysłanych danych? Wykonaj korekcję kanału:

$$\tilde{\mathbf{x}} = \mathbf{P} \mathbf{y}, \quad \text{gdzie } \mathbf{P} \text{ jest macierzą „pseudo odwrotną” macierzy } \mathbf{H} \text{ (użyj funkcji } \texttt{pinv}(\dots))$$

a następnie porównaj sygnał wysłany i odebrany po korekcji.

Sygnał $\mathbf{x}^{(3)}$ to suma dwóch harmoniczných o częstotliwościach 2 i 6 herców (dt to okres próbkowania równy odwrotności częstotliwości próbkowania: $1/f_{pr}$, w powyższym przypadku założono czas trwania sygnału równy 1 s). Charakterystyka częstotliwościowa filtru z odpowiedzią impulsową \mathbf{h} została dobrana tak, aby przepuścić pierwszą składową oscylacyjną o częstotliwości 2 Hz i stłumić drugą składową o częstotliwości 6 Hz. Sprawdź czy filtracja się udała.

Opcjonalnie (+1 pkt) porównaj wyniki splotu macierzowego ze splotem realizowanym funkcją `y=conv(h,x)`, wyświetl charakterystyki częstotliwościowe filtru \mathbf{h} oraz widmo sygnału $\mathbf{x}^{(3)}$ przed filtracją i po filtracji - użyj funkcji `freqz(...)`.

2. Dyskretna Transformata Fouriera DFT (2 pkt)

Skonstruuj macierz \mathbf{F} transformaty DFT według zależności:

$$\mathbf{F}(n, m) = \frac{1}{\sqrt{N}} \cos\left(\frac{2\pi}{N} nm\right) - j \frac{1}{\sqrt{N}} \sin\left(\frac{2\pi}{N} nm\right) = \frac{1}{\sqrt{N}} \exp\left(-j \frac{2\pi}{N} nm\right) \quad n, m = 0, \dots, N-1$$

dla $N=32$. Wykonaj transformację sygnału $\mathbf{x}^{(3)}$ z ćwiczenia 1 według zależności:

$$\mathbf{X} = \mathbf{F} \mathbf{x}$$

Wynik transformaty jest traktowany jako sygnał w dziedzinie częstotliwości i przedstawiany zazwyczaj jako widmo gęstości mocy w skali decybelowej tj.: $20\log_{10}(|\mathbf{X}|)$.

Następnie „wróć” do dziedziny czasu poprzez wykonanie odwrotnej transformacji Fouriera na sygnale \mathbf{X} na dwa sposoby:

$$\tilde{\mathbf{x}} = \mathbf{F}^{-1} \mathbf{X} ,$$
$$\tilde{\mathbf{x}} = \mathbf{F}^H \mathbf{X} = (\mathbf{F}^*)^T \mathbf{X}$$

Interpretacja wyników: Dla macierzy ortogonalnej prawdziwa jest zależność $\mathbf{F}^{-1} = \mathbf{F}^H = (\mathbf{F}^*)^T$, czyli oba powyższe sposoby powinny wygenerować takie same wyniki (sprawdź).

Zważywszy na ortonormalność macierzy \mathbf{F} , wektory $\tilde{\mathbf{x}}$ i \mathbf{x} powinny być identyczne z dokładnością do błędów numerycznych (sprawdź). Transformata DFT rozkłada sygnał na sumę składowych sinusoidalnych, tak więc w wyniku \mathbf{X} powinny być widoczne dwie wartości odpowiadające dwóm harmonicznym z sygnału \mathbf{x} . Wartości dla pozostałych harmonicznych powinny być na poziomie błędów numerycznych. Sprawdź. Dlaczego widzisz cztery składowe? Zamień funkcję $\sin(\dots)$ na $\exp(-j \dots)$ w sygnale $\mathbf{x}^{(3)}$ i powtórz cały eksperyment.

Powtórz analizę dla sygnału $\mathbf{y}^{(3)}$, czyli sygnału $\mathbf{x}^{(3)}$ przefiltrowanego przez kanał o charakterystyce \mathbf{h} . W tym celu wykonaj spłot odpowiedzi impulsowej \mathbf{h} z sygnałem $\mathbf{x}^{(3)}$. Do wykonania spłotu wykorzystaj funkcję `conv(...)`.

Wykonaj analizę i syntezę sygnału losowego (wykorzystaj funkcję `randn(...)` generującą sygnał pseudolosowy o rozkładzie normalnym). Sprawdź zawartość harmonicznych i błędy numeryczne po wykonaniu transformacji odwrotnej.

3. Autokorelacja sekwencji pseudolosowych (1 pkt)

Wygeneruj sekwencję pseudolosową C-REVERB1 o długości $n=1024$ bitów. Jest ona używana w modemach ADSL do estymacji odpowiedzi impulsowej kanału. Sekwencja ta definiowana jest następująco:

$$d_n = \begin{cases} 1 & n=1 \dots 9 \\ d_{n-4} \oplus d_{n-9} & n \geq 10 \end{cases}$$

Wykorzystując funkcję autokorelacji, wyznacz po ilu bitach sekwencja zaczyna się powtarzać.

4. Obliczenia macierzowe w C (opcjonalnie 2 pkt)

Standard języka C/C++ nie obejmuje mnożenia macierzowego. Taką operację należy zaimplementować samemu lub można skorzystać z gotowych bibliotek.

Najprostsza implementacja¹ zależności $\mathbf{C} = \mathbf{A}\mathbf{B}$ została zaprezentowana w tabeli poniżej.

```
for (i = 0; i < n; i++)
    for (j = 0; j < n; j++) {
        C[i + j*n] = 0;
        for (k = 0; k < n; k++)
            C[i + j*n] += A[i + k*n]*B[k + n*j];
    }
```

Zauważ, że wykorzystano tablice jednowymiarowe do reprezentacji macierzy dwuwymiarowych. Ma to sens ponieważ pamięć w komputerze jest jednowymiarowa więc i tak macierz dwuwymiarowa zostanie rozwinięta do jednowymiarowej. Reprezentacja jednowymiarowa potencjalnie umożliwi lepszą optymalizację kodu przez kompilator. Jeszcze lepsza byłaby bezpośrednia arytmetyka na wskaźnikach.

Wyznacz czas wykonania powyższego kodu dla macierzy o rozmiarach 32x32 i typu zmiennych `float` dla 3 opcji kompilacji:

- O0 (optymalizacja wyłączona)
- O2 (optymalizacja włączona)
- O2 -funroll-loops (jw. + rozwijanie pętli)

Wykonaj mnożenie macierzowe wielokrotnie aby sumaryczny czas wykonania wyniósł co najmniej 100 ms. Do dalszych obliczeń wykorzystaj dane z ostatnich 10 ms obliczeń (dlaczego?). Jeżeli tablica `C` nie zostanie w kodzie nigdzie użyta to kompilator w ramach optymalizacji może pominąć cały powyższy kod. Do dokładnego obliczania czasu wykorzystaj funkcje `gettimeofday(...)` lub `clock_gettime(...)`. Pierwsza funkcja pozwala na dokładność mikrosekundową i jest zgodna z

1 <http://www.stanford.edu/~jacobm/matrixmultiply.html>

POSIX.1-2001 lecz oznaczona jako przestarzała według nowego standardu. Druga instrukcja udostępnia teoretycznie dokładność nanosekundową, jest zgodna ze standardem POSIX.1-2008.

Dla najszybszej metody oblicz parametr FLOPS i porównaj z danymi deklarowanymi przez producenta CPU użytego w systemie. Traktuj operacje $+$ i $*$ równoważnie, nie zapomnij o indeksowaniu tablic ($C[n]$ jest traktowane przez kompilator jako $C+n$) oraz o operacjach arytmetycznych wynikających z konstrukcji pętli. Możesz zaniedbać operacje porównania wewnątrz pętli.s

Czas wykonania mnożenia porównaj z czasem wykonania równoważnego kodu w Matlabie. Użyj funkcji `tic` `toc` do mierzenia czasu. Wykonaj algorytm wielokrotnie i uśrednij wynik pomiaru czasu.