

PODSTAWY I ALGORYTMY PRZETWARZANIA SYGNAŁÓW  
TEORIA SYGNAŁÓW  
PRZETWARZANIE SYGNAŁÓW  
LABORATORIUM

ćwiczenie 1  
„Wprowadzenie do pakietu Octave”

**Cel ćwiczenia:** Zapoznanie się z podstawami użytkowania pakietu Octave. Typy zmiennych, operatory, instrukcje, funkcje, skrypty. Pisanie prostych skryptów, tworzenie wykresów.

## 1. Wstęp

Octave to pakiet przeznaczony do obliczeń numerycznych. Rozprowadzany jest na licencji GNU General Public License i może być dowolnie kopiowany i modyfikowany. Umożliwia tworzenie skryptów (programów) w języku bardzo zbliżonym do używanego w środowisku MATLAB, możliwe jest zatem bezpośrednie przenoszenie prostych skryptów między Octave i MATLABem.

## 2. Podstawy użytkowania

W systemie Windows program uruchamia się korzystając ze skrótu uruchamiającego, zawierającego dość złożone polecenie wywołujące konsolę. W przypadku skasowania skrótu należy go skopiować z innego źródła. W systemie Linux po zalogowaniu (`login:c`, `password:c`), aby uruchomić Octave należy wywołać program Xterm (Menu Debian→X-shells→Xterm) i wpisać `octave`. Po uruchomieniu programu otwiera się okno, w którym wprowadza się komendy. Obsługiwana jest historia wprowadzanych poleceń (klawisze ↑ i ↓). Przerwanie wykonywanego polecenia uzyskuje się wciskając Ctrl-C. Wyjście z programu następuje po wpisaniu komend `quit` lub `exit`.

Skrypty umożliwiają automatyczne wykonywanie przez program wielu poleceń tak, jakby były one po kolei wpisywane w oknie programu. Octave umożliwia również tworzenie rozbudowanych programów dzięki wbudowanym instrukcjom obsługi pętli, funkcji, instrukcjom warunkowym. Skrypty należy pisać w zewnętrznym edytorze, np. Notatniku w systemie Windows lub jednym z dostępnych edytorów tekstowych w systemie Linux, np. Nedit, który umożliwia oznaczanie składni. Przygotowany plik należy umieścić w katalogu roboczym. W systemie Windows domyślnym jest `octave_files`, znajdujący się w katalogu, w którym zainstalowany został Octave. W Linuxie natomiast jest nim katalog domowy użytkownika, np. `/home/c`. Plik musi mieć rozszerzenie `".m"`. Po wpisaniu polecenia `nazwa` w oknie programu, wykonywany jest skrypt `nazwa.m`, pod warunkiem, że znajduje się w katalogu roboczym. Użytkownik powinien mieć prawa zapisu do katalogu roboczego i katalogu tymczasowego. Jeśli praw takich nie ma, należy zmienić ww katalogi zgodnie z instrukcją przedstawioną poniżej.

### Poruszanie się po strukturze katalogów w Octave:

- Zmiana katalogu poleceniem `cd`. **Przy podawaniu ścieżki należy stosować znak `'/'` a nie windowsowe `'\'`!**
- Wyświetlenie zawartości katalogu poleceniem `ls`
- Wyświetlenie ścieżki aktualnego katalogu roboczego poleceniem `pwd`
- Tworzenie katalogu poleceniem `mkdir 'nazwa katalogu'`

### Zmiana katalogu roboczego Octave:

- Należy przejść do nowego katalogu roboczego używając komendy `cd`
- UWAGA: Wejście do katalogu głównego dysku C w Windowsie wykonuje polecenie `cd C:.` Proszę zwrócić uwagę, że nie oznacza to, że Octave „widzi” w tym momencie ścieżkę `C:/`. Po wpisaniu polecenia `pwd` ukaże się ścieżka `/cygdrive/c`. Wynika to ze sposobu dostępu Octave do systemu plików Windows.

### Zmiana katalogu tymczasowego Octave:

- Po uruchomieniu Octave proszę sprawdzić, czy działają wykresy. W tym celu należy wpisać np. `plot(0:10)`. Powinno otworzyć się okno wykresu, zawierające linię prostą. Jeśli wyświetlony zostanie komunikat o błędzie, oznacza to, że prawdopodobnie program nie miał praw zapisu do katalogu tymczasowego.
- Należy zmienić katalog tymczasowy poleceniem `putenv('TMPDIR','ścieżka');`. Za `'ścieżka'` należy podać ścieżkę do katalogu, w którym użytkownik ma prawa zapisu. Przykładowa ścieżka: `'C:/tmp'`.
- Ścieżkę katalogu tymczasowego wyświetla polecenie `tempdir`

Jeśli wpisywane polecenie nie jest zakończone znakiem ";", następuje wyprowadzenie wyniku jego działania na ekran. Znak komentarza "#" lub "%" powoduje, że tekst od tego znaku do końca linii jest ignorowany. Znak "..." umożliwia natomiast przeniesienie części danego polecenia do kolejnej linii.

Polecenie `help` umożliwia dostęp do pomocy, np. wpisanie `help funkcja1` powoduje wyświetlenie informacji o funkcji "funkcja1". Pod adresem <http://octave.sourceforge.net/index/> dostępna jest obszerna pomoc on-line.

### 3. Typy zmiennych

Podstawowymi typami zmiennych są

- macierze – dwuwymiarowe tablice o rozmiarach podawanych jako liczba wierszy (N) x liczba kolumn (M). Macierze wprowadza się wierszami, oddzielając elementy przecinkami. Kolejne wiersze oddzielane są średnikami, np.:

```
A = [1, 2, 3; 4, 5, 6]
```

Można również wykorzystywać w definicjach wcześniej zdefiniowane zmienne, np.:

```
B = [A; A; 4, 5, 6]
```

We wszystkich operacjach na macierzach ważne jest przestrzeganie spójności ich wymiarów.

Wartość skalarna rozumiana jest jako macierz o wymiarze 1 x 1.

- łańcuchy – ciągi znaków, wprowadzane w apostrofach lub cudzysłowie, np.

```
str1 = "kot"; str2 = 'kat';
```

Na łańcuchach można przeprowadzać operacje takie, jak na macierzach. Tam, gdzie to konieczne, program dokona konwersji znaków na liczby (kody ASCII). Np.:

```
A = [str1; str2; "ala"]
```

```
B = inv(A)
```

Funkcja `inv` odwraca macierz.

- zakresy – definiowane w postaci `a:b:c`, gdzie `a` to początek, `b` to skok i `c` to koniec. Równoważne są wektorowi wierszowemu z tą różnicą, że ich wartości obliczane są tylko wtedy, kiedy jest to wymagane. Poniższe polecenie zapisuje w wektorze wierszowym `sin_tab` wartości jednego okresu funkcji sinus pobierane co 0.2 radiana.

```
sin_tab = sin(0:0.2:2*pi)
```

Octave pracuje na liczbach zespolonych podwójnej precyzji. Liczby zespolone wprowadza się w postaci `a+j*b`, gdzie `a` to część rzeczywista, a `b` część urojona. Oznaczenia jednostki urojonej to również "I", "j" oraz "J". Zastosowany format liczb zawiera również pseudowartości `Inf` – nieskończoność i `NaN` – nie-liczba. Pojawiają się one najczęściej wskutek błędów w obliczeniach i przepełnień zakresów dopuszczalnych wartości. Zdefiniowane są stałe `pi` i `e` (podstawa logarytmu naturalnego).

Można pobierać wartości elementów macierzy i łańcuchów poprzez wskazanie ich indeksów. **Indeksy numerowane są od wartości 1!**. Indeksy podawane mogą być jako liczby, wektory, zakresy, zmienne itd. Operator ":" powoduje pobranie całego wiersza (kolumny). Przykłady:

```
A = [1, 2, 3; 4, 5, 6; 7, 8, 9]
```

```
B = A(1, 2)
```

 pobiera wartość o indeksie wiersza 1 i indeksie kolumny 2, czyli liczbę 2

```
B = A(:, 3)
```

 pobiera całą trzecią kolumnę

```
B = A([1, 3], [1, 3])
```

 pobiera elementy leżące w „rogach” macierzy A

```
B = A(1:2, 2:3)
```

 pobiera elementy należące jednocześnie do wierszy 1 i 2 oraz kolumn 2 i 3

## 4. Operatory

### Arytmetyczne:

+	dodawanie
-	odejmowanie
*	mnożenie macierzowe
.*	mnożenie każdego elementu macierzy z każdym
/	dzielenie macierzowe prawostronne $A/B = A \cdot \text{inv}(B)$
./	dzielenie każdego elementu macierzy przez każdy
\	dzielenie macierzowe lewostronne $A \setminus B = \text{inv}(A) \cdot B$
^	macierzowe podnoszenie do potęgi, np. $A^3 = A \cdot A \cdot A$
.^	podnoszenie do potęgi każdego elementu macierzy

### Logiczne i relacyjne:

==	równości
<> lub ~= lub !=	różności
>	nierówności
<	
>=	
<=	
	alternatywa (OR)
&	koniunkcja (AND)
~ lub !	negacja (NOT)

### Inne:

=	przypisania
++	inkrementacji (zarówno prawo jak i lewostronny)
--	dekrementacji (zarówno prawo jak i lewostronny)
'	sprzężenia hermitowskiego (transpozycja i zamiana na wartości sprzężone)
.'	transpozycji

## 5. Funkcje

Funkcje w Octave można podzielić na skompilowane (wbudowane), które nie mogą być zmieniane, ani nawet oglądane w postaci źródłowej przez użytkownika oraz nieskompilowane, które znaleźć można w postaci źródłowej. Funkcje nieskompilowane można modyfikować. Większość dostępnych funkcji jest właśnie w tej postaci. Użytkownik może również pisać własne funkcje. W tym celu należy stworzyć odpowiedni plik o nazwie tej samej co nazwa funkcji i zapisać go z rozszerzeniem ".m" w katalogu roboczym. W pliku należy umieścić funkcję w postaci:

```
function [wyn1, wyn2, ...] = nazwa_funkcji(arg1, arg2, ...)
% instrukcje
endfunction
```

Jak widać, możliwe jest podanie wielu argumentów (arg1, arg2 itd.) jak również uzyskanie wielu zmiennych wyjściowych (wyn1, wyn2 itd.). Wywołanie funkcji wygląda np. tak:

```
[a, b] = funkcja1(c, d);
a = funkcja2(e);
funkcja3(f);
funkcja4();
```

Instrukcja `return` powoduje natychmiastowe opuszczenie wykonywanej funkcji.

Zmienne globalne umożliwiają dostęp do tej samej zmiennej w różnych funkcjach. Żeby użyć zmiennych globalnych, należy zadeklarować je instrukcją `global zmienna1 zmienna2 ...` w **każdej** funkcji, w której zamierzamy z tych zmiennych korzystać.

## 6. Instrukcje

### Instrukcje warunkowe:

```
if (wyrażenie)
% instrukcje
endif
```

```
if (wyrażenie)
% instrukcje
else
% instrukcje
endif
```

```
if (wyrażenie)
% instrukcje
elseif (wyrażenie)
% instrukcje
else
% instrukcje
endif
```

Wyrażenie uznawane jest jako prawda, gdy jest różne od zera, a jako fałsz, gdy jest równe zeru.

### Petle:

```
while (wyrażenie)
% instrukcje
endwhile
```

```
do
% instrukcje
until (wyrażenie)
```

```
for var1 = var2
% instrukcje
endfor
```

W pętli for w kolejnych iteracjach zmienna var1 przyjmuje wartości kolejnych kolumn zmiennej var2, np.:

```
for k=(1:10)
k
endfor
```

Wyświetli w oknie liczby od 1 do 10, bo wyrażenie (1:10) interpretowane jest jako wektor wierszowy.

Instrukcja break przerywa wykonywanie bieżącej pętli, a continue powoduje skok do wyrażenia warunkowego pętli.

## 7. Przykłady podstawowych funkcji wbudowanych

### Tworzenie macierzy i zakresów:

zeros(N,M)	macierz NxM zer
ones(N,M)	macierz NxM jedynek
eye(N,M)	macierz NxM zer z jedynkami na głównej przekątnej, gdy N=M – macierz jednostkowa
rand(N,M)	macierz NxM wartości losowych o rozkładzie jednostajnym w przedziale [0;1]
randn(N,M)	macierz NxM wartości losowych o rozkładzie normalnym N(0,1)
linspace(P,K,N)	wektor wierszowy zawierający N liniowo rozłożonych wartości z przedziału od P do K

### Sprawdzanie wielkości macierzy:

size(A)	podaje wymiary macierzy
length(a)	podaje wymiar wektora

### Odczyt i zapis:

save plik zm1 zm2	zapisuje zmienne do pliku
load plik	wczytuje zmienne z pliku
Parametr -ascii	spowoduje odczyt i zapis danych w postaci znakowej.
autoload	wczytuje plik w formacie .wav
autosave	zapisuje plik w formacie .wav

### Funkcje matematyczne:

log(x)	logarytm naturalny
log2(x)	logarytm o podstawie 2
log10(x)	logarytm dziesiętny
exp(x)	e <sup>x</sup>
sin(x)	
cos(x)	
tan(x)	funkcje trygonometryczne
floor(x)	
ceil(x)	zaokrąglanie do liczby całkowitej
round(x)	
mod(x,y)	reszta z dzielenia x przez y
sign(x)	znak x

### Manipulacja zmiennymi:

who	wyświetla zmienne
clear zmienna	kasuje zmienną
clear all	kasuje wszystkie zmienne
exist("zmienna")	sprawdza, czy zmienna istnieje

### Inne:

disp("tekst")	wyświetla tekst
any(x)	przyjmuje wartość 1, gdy jakiegokolwiek element wektora x jest niezerowy

<code>abs(x)</code>	moduł $x$	<code>all(x)</code>	przyjmuje wartość 1, gdy wszystkie elementy wektora $x$ są niezerowe
<code>angle(x)</code>	argument $x$		
<code>real(x)</code>	część rzeczywista $x$		
<code>imag(x)</code>	część urojona $x$		

### Wykresy:

`plot(y)`

Rysuje wykres, w którym wartości umieszczone na osi odciętych zawiera wektor  $y$ , a odpowiadające im wartości na osi rzędnych przyjmowane są jako kolejne liczby naturalne.

`plot(x1,y1,"format1", x2,y2,"format2",...)`

Rysuje nałożone na siebie wykresy, w których wartości umieszczone na osi rzędnych zawierają wektory  $x1$  (pierwszy wykres),  $x2$  (drugi wykres) itd., a wartości umieszczone na osi odciętych wektory  $y1$ ,  $y2$  itd. Łańcuch "format" zawiera informacje o wyglądzie wykresu. Np. '-' oznacza linię ciągłą, '.' – tylko punkty, 'b' – kolor niebieski, 'r' – kolor czerwony, '\*' – markerem będzie gwiazdka. Pełny spis oznaczeń można zobaczyć w pomocy (`help plot`). Oznaczenia można łączyć ze sobą, np. "-\*r". Dołączenie łańcucha ";opis;" spowoduje opisanie wykresu. Przykład:

```
x = (0:0.1:2*pi);
y1 = sin(x);
y2 = cos(x);
plot(x, y1, "-b;sinus;", x, y2, "-r;cosinus;");
```

<code>bar(x,y)</code>	wykres słupkowy
<code>hist(x,n)</code>	histogram zmiennej $x$ rysowany dla $n$ przedziałów

<code>xlabel("opis")</code>	opis osi $x$
<code>ylabel("opis")</code>	opis osi $y$
<code>title("opis")</code>	tytuł wykresu
<code>axis([x1, x2, y1, y2])</code>	ustawianie zakresów osi
<code>replot</code>	odświeżanie wykresu
<code>clg</code>	kasowanie aktualnego wykresu
<code>hold on</code>	„blokowanie” wykresu tak, że kolejne będą nakładane na niego
<code>hold off</code>	„odblokowanie” wykresu

`subplot(x, y, n)`

Umożliwia rysowanie wielu wykresów w jednym oknie. Dzieli okno na  $x$  rzędów i  $y$  kolumn, co daje  $n=x*y$  pól. Instrukcja ta przełącza na pole o numerze  $n$ . Przykład użycia:

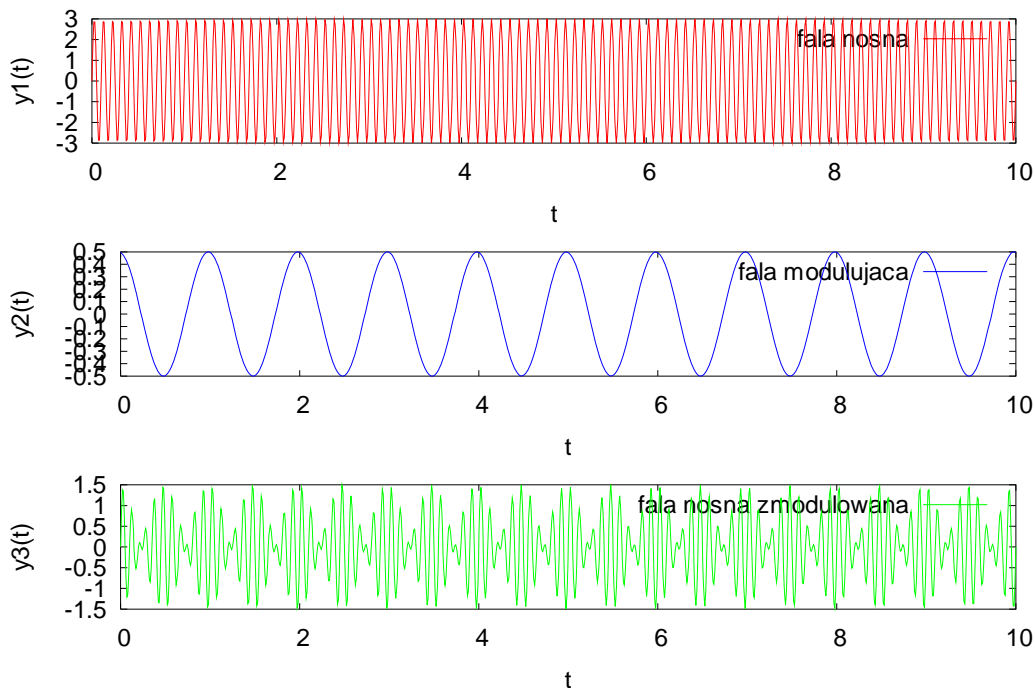
```
subplot(2, 2, 1); plot(a);
subplot(2, 2, 2); plot(b);
subplot(2, 2, 3); plot(c);
subplot(2, 2, 4); plot(d);
oneplot()                powrót do trybu jednego wykresu w oknie
```

## 8. Jak przyspieszyć wykonywanie skryptów?

- unikać pętli, stosować gdzie się tylko da obliczenia macierzowe
- używać iteracji zamiast rekurencji
- unikać sytuacji wymagających zmieniania wielkości macierzy, alokować macierze na początku obliczeń

## 9. Ćwiczenia do wykonania

- napisać funkcję obliczającą silnię
  - a) iteracyjnie
  - b) rekurencyjnie
- napisać funkcję rysującą wykres podobny do zamieszczonego poniżej



Wykres górny przedstawia sinusoidalną falę nośną  $y_1(t)=A_1\sin(2\pi f_1 t+\phi_1)$ , wykres środkowy sinusoidalną falę modulującą  $y_2(t)=A_2\sin(2\pi f_2 t+\phi_2)$ , wykres dolny natomiast falę nośną zmodulowaną amplitudowo  $y_3(t)=y_1(t)*y_2(t)$ . Funkcja ma umożliwiać podanie parametrów  $A_1$ ,  $A_2$ ,  $f_1$ ,  $f_2$ ,  $\phi_1$ ,  $\phi_2$  i zakresu zmiennej  $t$ .

**Wskazówka:** N próbek sinusoidy o częstotliwości  $f$ , próbkowanej z częstotliwością  $f_p$ , o fazie początkowej  $\phi$  generuje instrukcja

```
sin((0:2*pi*f/fp:2*pi*f/fp*(N-1))+phi)
```

lub

```
sin(linspace(0,2*pi*f/fp*(N-1),N)+phi)
```

Należy też przećwiczyć kopiowanie wykresów do innych programów, np. do Worda. W systemie Windows w menu okna otwieranego przez program rysujący wykresy Gnuplot znajduje się funkcja Options→Copy to Clipboard. W systemie Linux natomiast można skorzystać z polecenia `print`, które umożliwia zapisanie **pojedynczego** wykresu w pliku w kilku formatach, np. komenda `print("wykres.eps", "-color")` zapisze bieżący wykres w pliku o podanej nazwie w formacie EPS z wykorzystaniem kolorów. W przypadku wykresów w trybie multiplot (wiele wykresów w jednym oknie) można użyć programu do robienia zrzutów ekranowych (Actions→Take Screenshot...). Przygotowane wykresy można wysłać pocztą elektroniczną, korzystając np. z programu Mozilla Mail.