ĆWICZENIE 2 Wektory i macierze w Scilab – tworzenie, przekształcanie, indeksowanie i obliczenia.

Cel: Celem tego rozdziału jest zapoznanie studenta ze stosowaniem notacji macierzowej w pracy ze SCILAB. Omówione zostaną zasady tworzenia zapisu danych w formie wektorów i macierzy, korzystania z danych zapisanych w macierzach oraz prowadzenia obliczeń z wykorzystaniem zapisu macierzowego.

UWAGA !

Na początku każdych kolejnych zajęć, po uruchomieniu programu należy ustawić własny katalog bieżący w oknie *Przeglądarka plików***. Powinien to być katalog, który został utworzony na dysku komputera w trakcie pierwszych zajęć. Na pierwszych zajęciach należy takie katalogi utworzyć. Kończąc zajęcia, należy „wyczyścić" okna Scilaba: *Console* (polecenie '*clc*') oraz *Historia poleceń* (po kliknięciu w oknie *Historia poleceń* wybrać z menu-*Edycja-Wyczyść historię* lub po 'kliknięciu' w dowolny element historii prawym klawiszem myszy wybrać z menu kontekstowego polecenie '*Wyczyść historię'*). W razie potrzeby uprzednio zapisać do pliku dane z** *Workspace*.

Pierwszym poleceniem jakie należy wpisać w *Konsoli* rozpoczynając zajęcia z Scilabem (zaraz po ustawieniu katalogu bieżącego) jest polecenie *diary('nazwa_pliku')*. Jako *nazwa_pliku* należy wpisać kolejny numer zajęć np*. *lab1.txt*, *lab2.txt*, itd. Polecenie to spowoduje zapisanie do pliku tekstowego całej zawartości pojawiającej się w *Konsoli*, tzn. wszystkich poleceń wpisywanych w trakcie zajęć, a także uzyskiwanych wyników. Należy również otworzyć okno edytora wpisując w Konsoli polecenie *edit* .

Następnie zarówno w edytorze jak też w '*Console*' należy wpisać linię komentarza z imieniem, nazwiskiem i datą:

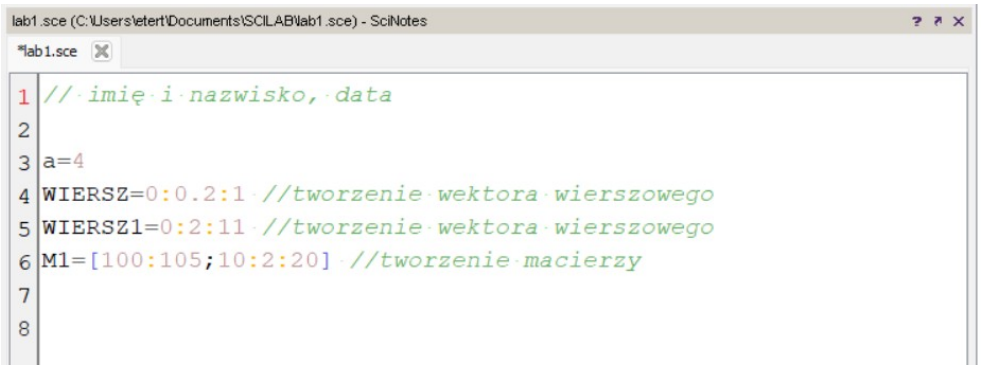
```
--> //  Imię, Nazwisko, Data
```

Plik utworzony w edytorze (SciNotes) zapisać pod nazwą zawierającą numer zajęć np: *lab1.sce.* Oba utworzone pliki (plik tekstowy .txt oraz plik z edytora .sce) będą formą sprawozdania z zajęć.

1. WSTĘP

W Scilab można pracować na dwa komplementarne sposoby. Najprostszy to wpisywanie poleceń bezpośrednio w *Konsoli*: nadaje się do szybkich prób, pojedynczych obliczeń, sprawdzania składni i podglądu wyników. Przy takiej pracy otrzymujemy natychmiastowy podgląd wyników, a gdy chcemy powtórzyć polecenie bądź poprawić jego zapis, wygodne jest korzystanie z historii poleceń. Praca w *Konsoli* środowiska Scilab polega na wpisywaniu poleceń, które po zatwierdzeniu (*ENTER*) są wykonywane przez interpreter programu. Wynik obliczeń (wartość) zostaje przypisany zmiennej, dla której prowadzono obliczenia. Jeżeli nazwy zmiennej nie podano, to wynik jest przypisany standardowej zmiennej ***ans***.

Gdy tworzymy dłuższe obliczenia, warto użyć edytora SciNotes. W edytorze zapisujemy skrypt (najczęściej jako plik .sce), który może zawierać wiele poleceń (wiele linii kodu), a także komentarz objaśniający działanie polecenia (rys.2.1).



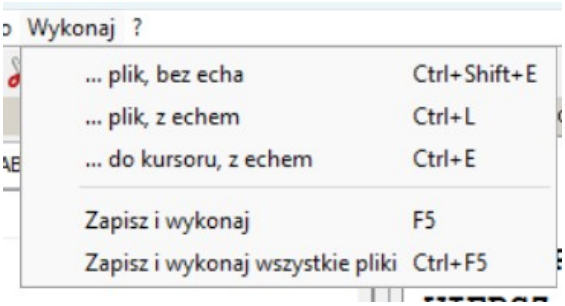
Rys.2.1: Okno edytora (SciNotes) z listą poleceń oraz komentarzami.

Taki plik może zostać zapisany i wykorzystywany w późniejszym czasie. Polecenia zapisane w pliku można uruchomić z poziomu przycisku/menu „Wykonaj” (wyniki i ewentualne komunikaty pojawią się w konsoli), a w razie potrzeby można także zaznaczyć wybrane polecenia i uruchomić tylko wybrane linie. Jeżeli dokonano zmian w poleceniach zapisanych w edytorze należy zapisać plik przed uruchomieniem poleceń.

Na rysunku obok tekstu widać tryby uruchamiania poleceń zapisanych w edytorze.

…plik, bez echa - wykonuje wszystkie zapisane polecenia bez wyświetlania wyników w *Konsoli*, zmienne są obliczane i dostępne w *Przeglądarce zmiennych*,

…plik, z echem - wykonuje wszystkie zapisane polecenia, wyświetla wyniki w *Konsoli*, zmienne są obliczane i dostępne w *Przeglądarce zmiennych*,



…do kursoru, z echem - wykonuje polecenia od początku do miejsca, w którym ustawimy kursor, wyświetla wyniki w *Konsoli*, zmienne są obliczane i dostępne w *Przeglądarce zmiennych*,

Jeśli zaznaczymy w edytorze tylko wybrane linie to polecenie *‘…do kursoru, z echem’* zostanie zastąpione poleceniem:

…zaznaczenie z echem - wykonuje zaznaczone polecenia, wyświetla wyniki w *Konsoli*, zmienne są obliczane i dostępne w *Przeglądarce zmiennych*,

Polecenia: ***…do kursoru, z echem***, ***…zaznaczenie z echem*** są dostępne również w menu kontekstowym po kliknięciu prawym klawiszem w oknie edytora.

Obie metody wprowadzania poleceń współdziałają: zmienne utworzone przez skrypt zapisany w edytorze są widoczne w bieżącej sesji, jakby powstały w Konsoli i mogą być wykorzystywanie w poleceniach “Konsolowych”. Błędy zgłaszane podczas wykonania poleceń zapisanych w edytorze są wyświetlane w *Konsoli* ale odwołują się do numerów linii w edytorze, co ułatwia identyfikację błędów i ich poprawianie.

UWAGA

W dalszej pracy będą wykorzystywane obie metody prowadzenia obliczeń. Pojedyncze polecenia umieszczone w tekście skryptu należy wpisywać bezpośrednio do *Konsoli.* **Przykłady oraz zadania** (w skrypcie wyraźnie oznaczone słowami **Przykład** oraz **Zadanie**) należy wpisywać do *edytora*. **UWAGA:** do *edytora* nie przepisujemy znaków zachęty. Wpisywanie w edytorze przykładu lub zadania należy rozpocząć linią komentarza z numerem przykładu/zadania, np. `//Przykład 2.1`

Podstawową strukturą danych Scilaba jest tablica. Elementami tablicy mogą być liczby rzeczywiste, zespolone, wartości logiczne znaki, kombinacje tych zmiennych lub inne tablice. Szczególnym przypadkiem tablicy (tablica dwuwymiarowa) jest macierz, szczególnym i najczęściej wykorzystywanym w obliczeniach przypadkiem macierzy jest macierz liczbowa. Działanie Scilaba oparto na operacjach wektorowo-macierzowych. W programie nie występują zmienne skalarne. Pojedynczy element (liczba/znak) jest traktowany jak macierz o jednym wierszu i jednej kolumnie. Wektor będący szczególnym przypadkiem macierzy, jest traktowany jako macierz zawierająca jedną kolumnę (kolumnowa) lub jeden wiersz (wierszowa). Scilab stosuje „kolumnową" konwencję obliczeń (uporządkowanie pamięci kolumnami), co bywa istotne np. przy zmianie 'kształtu' macierzy.

W praktyce macierze służą do:

- opisu serii pomiarowych (np. każda kolumna to inna seria pomiarów),
- rozwiązywania układów równań liniowych i zagadnień algebry liniowej,
- przetwarzania sygnałów i obrazów (macierze pikseli),
- generowania siatek punktów do wykresów i symulacji.

2. Tworzenie wektorów i macierzy

2.1. Wpisywanie z klawiatury (notacja z nawiasami kwadratowymi)

Zapisując macierz w SCILAB, wykorzystuje się nawiasy kwadratowe [ ]. Poszczególne elementy wiersza macierzy oddziela się spacjami lub przecinkami, a wiersze średnikami lub umieszcza się je w oddzielnych liniach.

```
-->A = [1 2 3;4 5 6] // macierz 2x3
-->A = [1,2,3;4,5,6] // macierz j.w.
-->v = [10, 20, 30] // wektor wierszowy 1x3 `
-->w = [10;20;30] // wektor kolumnowy 3x1
```

Do momentu zamknięcia nawiasu można 'łamać' linię wciskając klawisz 'enter' w miejscu średnika. Powyższa metoda wymaga podania wartości wszystkich elementów macierzy, co może być kłopotliwe przy macierzach zawierających znaczną ilość elementów. Np. macierz 20x20 zawiera 400 elementów.

2.2. Tworzenie wektorów równomiernych: operator dwukropka ' : ' , polecenie linspace.

Macierz/wektor można uzyskać bez wypisywania jej wszystkich elementów. W tym celu należy zastosować dwukropek jako operator generowania wektorów i tablic.
Definiowanie wektora przez generowanie elementów:
A=min:krok:max lub A=[min:krok:max]– zostanie utworzony wektor wierszowy
Polecenie generuje wektor wierszowy rozpoczynając od elementu o wartości *min*, następnie każdy kolejny element jest zwiększany o wartość *krok* aż do osiągnięcia elementu o wartości *max* (Uwaga: ostatni element wektora nie musi mieć wartości *max*).

*min:krok:max → [min, min+krok, min+2*krok, min+3*krok ... max]*

```
-->WIERSZ=0:0.2:1
-->WIERSZ1=0:2:11
-->WIERSZ2=10:-2:0
```

Jeżeli parametr *krok* zostanie pominięty, przyjmuje się, iż *krok*=1.

min:max → [min, min+1, min+2, min+3 ... max]

```
-->M1=100:105
```

Aby utworzyć wektor kolumnowy należy wektor wierszowy poddać transpozycji:

```
-->KOLUMNA=[0:0.2:1].'
```

Metoda wykorzystująca dwukropek może być wygodna również przy tworzeniu macierzy:

```
-->A2=[1:3;4:6]
```

Operator dwukropka ':' oprócz tworzenia wektorów równomiernych umożliwia również wybieranie/indeksowanie zakresów elementów macierzy (o tym dalej).

Polecenie *x=linspace(x1,x2,n)* utworzy macierz jednowierszową (wektor wierszowy) mającą n wyrazów rozłożonych liniowo (równomiernie), gdzie pierwszym wyrazem jest liczba *x1* a ostatnim *x2* a odległości (różnice) między sąsiednimi elementami są takie same.

```
-->M2=linspace(6,1,6)
```

Analogicznie można utworzyć wektor kolumnowy o elementach rozłożonych liniowo (równomiernie):

```
-->M2k=linspace(1,5,6)'
```

Polecenie *linspace* jest wygodne, gdy zależy nam na włączeniu końców przedziału przy danej liczbie punktów. Dwukropek ":" preferujemy, gdy "krok" pomiędzy punktami jest istotniejszy niż liczba punktów.

2.3. „Łączenie" (konkatenacja) wektorów i macierzy.

Rozdzielając wektory średnikami, lub zapisując je w oddzielnych liniach można za ich pomocą zdefiniować macierz.

```
-->MA=[WIERSZ;M1;M2]
MA  =
 0.    0.2   0.4   0.6   0.8   1.
 100.  101.  102.  103.  104.  105.
 6.    5.    4.    3.    2.    1.
```

→ WIERSZ

→ M1

→ M2

Przykład 2.1: *(wpisać w oknie edytora)*

Macierz dwuwierszowa o wyrazach od 1 do 10 w pierwszym wierszu i o wyrazach od 2 do 20 (co 2) w wierszu drugim.

```
--> A=[1:10; 2:2:20]
```

Możliwe jest też wygenerowanie macierzy korzystają z innych macierzy.

Przykład 2.2: *(wpisać w oknie edytora)*

Macierz **D** zbudowana ze zdefiniowanych macierzy **A**, **B** i **C**.

```
--> A=[1 4 1; 2 0 1]
-->B=[3 1; 4 1]
-->C=[1 2 2 0 1; 2 4 7 1 0]
--> D=[A B; C]
```

D =

1.	4.	1.	3.	1.
2.	0.	1.	4.	1.
1.	2.	2.	0.	1.
2.	4.	7.	1.	0.

UWAGA:

Przy takim budowaniu macierzy należy pamiętać o zgodności wymiarów (zgodności liczby elementów w wierszach i kolumnach odpowiednich macierzy).

2.4. Wyświetlanie i podstawowe informacje o macierzy.

Wyświetlanie macierzy

Wpisanie nazwy zmiennej zawierającej macierz/wektor i *Enter* wyświetla jej nazwę i zawartość.

```
--> A
```

Polecenie *disp(A)* - pokazuje tylko zawartość macierzy;

```
--> disp(A)
```

Wymiar macierzy

`--> size(A)` - zwraca wektor dwuelementowy, gdzie pierwszy element podaje liczbę wierszy a drugi kolumn

`-->[m,n]=size(A)` – przypisuje liczbę wierszy macierzy **A** do zmiennej *m* a liczbę kolumn do zmiennej *n*;

`--> size(A, 'r')` - tylko liczba wierszy macierzy **A**;

`-->size(A, 'c')` - tylko liczba kolumn macierzy **A**;

`-->n=length(B)` - zwraca wymiar wektora **B** (lub liczbę elementów macierzy **B**);

Zadanie 2.1: *(wpisać w oknie edytora)*

Wygenerować wektory:

w1 – kolejne liczby całkowite od 1 do 6;

w2 – kolejne liczby całkowite od 1 do 6.5

w3 – liczby od 1 do 2 co 0.1;

w4 – liczby od 55 do 10 co 10;

Zadanie 2.2: *(wpisać w oknie edytora)*

Wygenerować macierze:

A o wymiarze 5x5, o wierszach z kolejnych liczb całkowitych od 1 do 25.

Powyższą macierz utworzyć:

- wpisując z klawiatury zakresy poszczególnych wierszy rozdzielone średnikami;
- tworząc wektory dla każdego wiersza, a następnie macierz z wektorów wierszowych,
- wykorzystując polecenie *matrix()* - wykorzystać okno pomocy (help matrix) aby zapoznać się z poleceniem.

2.5. Funkcje wspomagające konstruowanie macierzy

Funkcja	Opis
eye(n, m)	tworzy macierz jednostkową o rozmiarze n x m z jedynkami na głównej przekątnej
ones(n, m)	tworzy macierz n x m o wszystkich elementach równych 1
zeros(n, m)	tworzy macierz n x m o wszystkich elementach równych 0
rand(n, m)	tworzy macierz o rozmiarze n x m wypełnioną liczbami pseudolosowymi z przedziału <0,1> o rozkładzie jednostajnym
randn(n,m,'normal')	tworzy macierz o rozmiarze n x m wypełnioną liczbami pseudolosowymi o rozkładzie normalnym ze średnią 0 i wariancją równą 1
inv(A)	macierz odwrotna do **A**, **inv(A)=A^(-1)**

Macierz jednostkowa (*eye*): (1 na diagonali, 0 poza)

Utwórz kwadratową macierz jednostkową **AA** o wymiarze 3x3.

```
--> AA=eye(3,3)
```

Macierz wypełniona jedynkami (*ones*): Utwórz macierz **AB** o wymiarze 2x3 wypełnioną jedynkami.

```
--> AB=ones(2,3)
```

Macierz wypełniona zerami (*zeros*): Utwórz macierz **AC** o wymiarze 2x3 wypełnioną zerami.

```
--> AC=zeros(2,3)
```

Macierz o elementach losowych: Utwórz macierz **AL** o wymiarze 4x4 wypełnioną liczbami losowymi.

Liczby o rozkładzie równomiernym w zakresie od 0 do 1:

```
--> AL=rand(4,4)
```

Liczby o rozkładzie normalnym z wartością średnią 0 i wariancją 1:

```
--> AL1=rand(4,4,'normal')
```

Przykład 2.3: *(wpisać w oknie edytora)*

Utworzyć macierz Durera (magiczny kwadrat):

```
-->MM=[16 3 2 13 ; 5 10 11 8 ; 9 6 7 12 ; 4 15 14 1]
```

W takiej macierzy sumy elementów w poszczególnych wierszach, kolumnach oraz w głównych przekątnych są sobie równe i wynoszą 34.

Wyliczenie sum w wierszach macierzy:

```
--> sum(MM,'c')
```

Wyliczenie sum w kolumnach macierzy:

```
--> sum(MM,'r')
```

Wyliczenie sumy elementów głównej przekątnej:

```
--> sum(diag(MM))
```

Powyższe polecenie zawiera w sobie dwa działania wykonywane w następującej kolejności: polecenie *diag(MM)* tworzy wektor kolumnowy z elementów przekątnej, polecenie *sum()* sumuje składniki tego wektora.

Wyliczenie sumy elementów drugiej przekątnej:

```
--> sum(diag(flipdim(MM,2)))
```

Powyższe polecenie zawiera w sobie trzy działania wykonywane w następującej kolejności: polecenie *flipdim(MM,2)* przekształca macierz odwracając kolejność jej kolumn, polecenie *diag()* tworzy wektor kolumnowy z elementów przekątnej przekształconej macierzy, polecenie *sum()* sumuje składniki tego wektora.

Przekształcenia macierzy

```
--> A=[1 4 1; 2 0 1]
```

Transpozycja (zamiana miejscami wierszy i kolumn):

`-->A'` //(dla liczb rzeczywistych równoważne zwykłemu transponowaniu). Dla liczb zespolonych `A'` obejmuje sprzężenie; transponowanie bez sprzężenia zapewnia `A.'`

Odwracanie kolejności wierszy/kolumn:

`-->flipdim(A, 1)` // odwraca kolejność wierszy (góra–dół)

`-->flipdim(A, 2)` // odwraca kolejność kolumn (lewo–prawo)

Zmiana kształtu (tzw. „reshape"):

`--> B = matrix(A, 3, 2)` // przebudowuje A na rozmiar 3x2 (liczba elementów musi się zgadzać, czyli macierz A musi zawierać 6 elementów)

Trójkątowanie:

`-->L = tril(A)` // część trójkątna dolna (pozostałe zera)

`-->U = triu(A)` // część trójkątna górna (pozostałe zera)

Podstawowe działania agregujące (sumowanie, średnia, min/max)

`-->sum(A)` // suma elementów macierzy A

`-->sum(A, 'r')` // suma elementów w kolumnach macierzy A (wynik jest wektorem wierszowym)

`-->sum(A, 'c')` // suma elementów w wierszach macierzy A (wynik jest wektorem kolumnowym)

`-->mean(A), mean(A,'r'), mean(A,'c')` // średnia wg reguł j.w.

`-->prod(A), prod(A,'r'), prod(A,'c')` // iloczyn wg reguł j.w.

`-->min(A), min(A,'r'), min(A,'c')` // wartość minimalna wg reguł j.w.

`-->max(A), max(A,'r'), max(A,'c')` // wartość maksymalna wg reguł j.w.

2.6. Indeksowanie: wybór elementów i fragmentów macierzy

Podstawowe metody wyboru elementów macierzy:

Element macierzy znajdujący się w wierszu o indeksie **i** oraz w kolumnie o indeksie **j** jest określony jako **A(i,j)**. Elementem takim można posługiwać się jak każdą zmienną. Do elementów macierzy można się też odwoływać przy użyciu tylko jednego indeksu, np. **A(k)**. W przypadku wektora odwołanie takie odnosi się do kolejnego elementu wektora, natomiast w przypadku macierzy odwołanie takie zostanie potraktowane jako odwołanie do wektora kolumnowego utworzonego z kolejnych kolumn macierzy.

Podstawowy sposób indeksowania pojedynczego elementu macierzy ma postać A(i, j),
gdzie: i – numer wiersza, j – numer kolumny.
A(i,j) – wypisanie elementu z i-tego wiersza i j-tej kolumny macierzy A;
Możliwe jest również indeksowanie zakresów zawierających wiele elementów macierzy:
A(:,j) – wypisanie całej j-tej kolumny macierzy A;
A(i,:) – wypisanie całego i-tego wiersza macierzy A;
A(k) – wypisanie k-tego elementu macierzy A (elementy zliczane kolumnami);
A(:) – wypisanie wszystkich elementów macierzy A w jednej kolumnie.

Symbol $ odnosi się do elementu ostatniego, np: A(2,$) indeksuje element z 2-go wiersza i ostatniej kolumny.

Przykład 2.4: *(wpisać w oknie edytora)*
Wygenerować macierz **A** o wymiarze 3x3 o wierszach z kolejnych liczb całkowitych od 1 do 9 a następnie wykonać polecenia:

`-->A`	`-->mean(A)`	`-->A(1:$)`
`-->A(1,1)`	`-->mean(A,'r')`	`-->A(1,3)`
`-->A(1,2)`	`-->A(1)`	`-->A(3,1)`
`-->A(:,1)`	`-->A(6)`	`-->min(A)`
`-->A(:,3)`	`-->A(7)`	`-->max(A)`
`-->A($,:)`	`-->A(1:10)`	`-->A(:)`

Przykład 2.5: *(wpisać w oknie edytora)*
Złożone metody wyboru elementów macierzy:
Wykonać polecenia i przeanalizować ich wyniki: (polecenia wpisywać kolumnowo, w edytorze opisać działanie polecenia w postaci komentarza do polecenia)

```
-->A=[1 2 3 4 5 6; 0 9 8 7 6 5; 1 1 0 0 2 2]
```

`-->B=A(:,[1:3 5])`	`-->B=A([1 2], 1:3: $)`	`-->B=A([1 3],[2 4 5])`	`-->A(2,:)=[ ]`
`-->B=A([1 3],1:2:5)`	`-->B=A([1 3],3:$)`	`-->B=A([3 $-1], 2: $-2)`	`-->A(:,1:2)=[ ]`

2.7 Podstawowe działania na macierzach i tablicach.

Scilab umożliwia wykonywanie dwóch rodzajów operacji: na macierzach oraz na ich elementach. Operacje **macierzowe** są wykonywane zgodnie z regułami zasad algebry liniowej (np. A*B -mnożenie macierzy wg zasad algebry liniowej). Drugi rodzaj działań to tzw. arytmetyczne operacje **tablicowe**, wykonywane na elementach macierzy (np. A.*B –wykonuje mnożenie elementów macierzy o tych samych indeksach). Aby możliwe było wykonanie mnożenia macierzowo lub tablicowo, macierze muszą mieć odpowiednie rozmiary. Dla działań tablicowych oraz mecierzowego dodawania i odejmowania macierze muszą mieć takie same rozmiary. Wyjątkiem jest tu skalar który jest „rozszerzany" automatycznie, np. A + 2 doda 2 do każdego elementu A.

Dla mnożenia macierzowego, pierwsza macierz powinna mieć tyle kolumn ile druga ma wierszy, wynik ma tyle wierszy, ile ma macierz pierwsza i tyle kolumn ile ma druga. Szczególnym przypadkiem mnożenia macierzy jest potęgowanie, które należy rozpatrywać jako wielokrotne mnożenie macierzowe i jest możliwe tylko dla macierzy kwadratowej.

A(m x **n**) * **B**(**n** x k) = **W**(m x k)

Szczególne przypadki mnożenia wektorów (A – wektor kolumnowy, B - wektor wierszowy):

A(m x **1**) * **B**(**1** x k) = **W**(m x k) – wynik jest macierzą

B(1 x **n**) * **A**(**n** x 1) = **W**(1 x 1) – wynik jest liczbą

Mnożenie wektora i macierzy:

B(1 x **n**) * **C**(**n** x k) = **W**(1 x k) – wynik jest wektorem wierszowym

C(k x **m**) * **A**(**m** x 1) = **W**(k x 1) – wynik jest wektorem kolumnowym

Dzielenie macierzy w SCILAB jest możliwe tylko dla nieosobliwych macierzy kwadratowych. W SCILAB występuje dzielenie lewostronne i prawostronne. Pamiętając o nieprzemienności operacji macierzowych, z zastosowaniem dwóch operatorów dzielenia otrzymamy cztery wyniki dzielenia. Operator lewostronny A\B – macierz stojąca po prawej stronie jest dzielona przez macierz stojącą po lewej stronie - ogólnie służy do rozwiązywania równań liniowych. Operator prawostronny A/B– macierz stojąca po lewej stronie operatora jest dzielona przez macierz stojącą po jego prawej stronie:

Transpozycja macierzy jest to zamiana wierszy macierzy z kolumnami. Jeżeli mamy macierz o składnikach rzeczywistych to transpozycja macierzowa i tablicowa daje taki sam wynik, natomiast różni się gdy składnikami macierzy są liczby zespolone. W tym przypadku transpozycja tablicowa zamienia tylko wiersze z kolumnami, natomiast transpozycja macierzowa zwraca macierz o elementach sprzężonych: : A' - macierzowa, A.’ – tablicowa.

Przykład 2.6: Rozwiązywanie układu równań liniowych. *(wpisać w oknie edytora)*

Rozwiązaniem równania liniowego $a*x = b$ jest wyrażenie: $x = b/a$

W przypadku układu równań

$$\begin{matrix} a_{11}x_1 & a_{12}x_2 & \dots & a_{1n}x_n \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1}x_1 & a_{n2}x_2 & \dots & a_{nn}x_n \end{matrix} = \begin{matrix} b_1 \\ \vdots \\ b_n \end{matrix}$$

można zastosować zapis w postaci macierzowej

$A*x=b$ gdzie:

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \quad \boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \quad b = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$$

Rozwiązanie powyższego układu równań uzyskuje się przez dzielenie x = A\b. (Nietypowy jest zapis: najpierw dzielnik, potem dzielna oraz odwrotna kreska ułamkowa (backslash).)

Znaleźć rozwiązanie układu równań:

$$\begin{cases} x + y + z = 3 \\ x - y + z = 1 \\ x - y - z = -3 \end{cases}$$

```
--> A=[1,1,1;1,-1,1;1,-1,-1]
--> b=[3,1,-3]’
--> xyz=A\b
```

Operatory arytmetyczne działań macierzowych i tablicowych:

macierzowe		tablicowe
+	Dodawanie	+
-	Odejmowanie	-
*	Mnożenie	.*
^	Potęgowanie	.^
/	Dzielenie prawostronne	./
\	Dzielenie lewostronne	.\
‘	Sprzężenie macierzy	
.’	Transpozycja macierzy	

Inne przydatne funkcje:

Funkcja	Opis
modulo(n,m)	reszta z dzielenia n/m
getdate()	znacznik czasu wg zegara komputera [Rok, miesiąc_R, tydzień_R, dzień_R, dzień_tyg., dzień_mies. , godzina_dnia, minuta, sekunda, milisekunda]
unique(A)	wyodrębnia i sortuje unikalne elementy wektora lub wiersze/kolumny macierzy
string(x)	konwersja zmiennej x do formatu tekstowego
disp()	wyświetlanie zmiennych i tekstu w konsoli
write(f,a)	zapis liczby lub tekst w konsoli - (f=%io(2)) lub w pliku - (f=nazwa pliku)

Zadanie 2.3: Wykonać działania i przeanalizować wyniki: *(wpisać w oknie edytora)*

Utworzyć macierze:

A =
1. 2. 3.
4. 5. 6.
7. 8. 9.

B =
1. 1. 1.
2. 2. 2.
3. 3. 3.

Wykonać działania:

Obliczyć różnicę macierzy A i B

0. 1. 2.
2. 3. 4.
4. 5. 6.

Utworzyć macierz C będącą iloczynem macierzy B i A oraz C1 będącą iloczynem macierzy A i B

C =
12. 15. 18.
24. 30. 36.
36. 45. 54.

C1 =
14. 14. 14.
32. 32. 32.
50. 50. 50.

Utworzyć macierz D przez pomnożenie elementów macierzy A przez 2 i dodanie elementów macierzy C

D =
14. 19. 24.
32. 40. 48.
50. 61. 72.

Dokonać mnożenia elementów macierzy A i C o tych samych indeksach (mnożenie tablicowe)

12. 30. 54.
96. 150. 216.
252. 360. 486.

Zadanie 2.4: *(wpisać w oknie edytora)*

Utworzyć wektor kolumnowy **W1** złożony z kolejnych liczb całkowitych od 1 do 5 i następnie obliczyć sumę kwadratów elementów tego wektora. (zastosować reguły mnożenia wektorów)

Zadanie 2.5: *(wpisać w oknie edytora)*

Utwórz macierz **M5** o wymiarach 10x10, której elementy pierwszego i ostatniego wiersza, oraz pierwszej i ostatniej kolumny to jedynki a pozostałe elementy to zera. (Zastosować polecenia zeros(), ones())

Zadanie 2.6: *(wpisać w oknie edytora)*

Utworzyć macierz: (zastosować reguły mnożenia wektorów)

1.	2.	3.	4.	5.	6.	7.	8.
2.	4.	6.	8.	10.	12.	14.	16.
3.	6.	9.	12.	15.	18.	21.	24.
4.	8.	12.	16.	20.	24.	28.	32.
5.	10.	15.	20.	25.	30.	35.	40.
6.	12.	18.	24.	30.	36.	42.	48.
7.	14.	21.	28.	35.	42.	49.	56.
8.	16.	24.	32.	40.	48.	56.	64.
9.	18.	27.	36.	45.	54.	63.	72.
10.	20.	30.	40.	50.	60.	70.	80.