ĆWICZENIE 1-2

SCILAB Wprowadzenie/Przypomnienie

Scilab jest środowiskiem programowym, które w sposób interaktywny umożliwia prowadzenie obliczeń numerycznych o charakterze inżynierskim a także naukowym. Działanie i możliwości programu są zbliżone do oferowanych przez takie programy jak Matlab czy GNU Octave. SCILAB został stworzony przez francuski instytut badawczy INRIA, był rozwijany przez Konsorcjum SCILAB (DIGITEO) a aktualnie (od 2022) zespół SCILAB jest częścią Dassault Systèmes. Na liście referencyjnej SCILAB można znaleźć takie przedsiębiorstwa jak: Airbus, Total, AcelorMittal, Peugeot, Hutchinson i in. Środowisko umożliwia prowadzenie działań o różnym stopniu skomplikowania, od podstawowych obliczeń matematycznych, poprzez tworzenie prostych algorytmów, modelowanie i symulację, graficzna prezentację danych po skomplikowane obliczenia i symulacje z możliwością ich zaprogramowania i prezentacji wyników w postaci wykresów różnej postaci w tym również 3-wymiarowych. Niewątpliwą zaletą SCILAB-a jest to, że jest to oprogramowanie, które każdy może bezpłatnie pobrać, zainstalować i wykorzystywać w dowolnym celu. Na moment pisania skryptu najnowszą wersją jest *Scilab 2023.1.0* , ale dostępne są również wersje poprzednie: *Scilab 6.* oraz Scilab 5.* ,* dostępne na stronie *https://www.scilab.org/* (link do pobierania jest dostępny również ze strony pracowniczej *https://staff.uz.zgora.pl/etertel/linki.html.*)

Scilab jest otwartym środowiskiem programistycznym wyposażonym w język programowania wysokiego poziomu umożliwiający użytkownikowi tworzenie własnych funkcji i bibliotek. Scilab posiada własny Edytor z wbudowanym debugerem. Wśród niezbędnych narzędzi programistycznych, oprócz Edytora należy wymienić: Interpreter poleceń, Biblioteki funkcji, Interfejsy dla innych języków: m.in. Fortran, Tcl/Tk, Java, Modelica, LabVIEW. Dane na których operuje program są przedstawiane w postaci **macierzy/tablic**. Elementy macierzy mogą być rzeczywiste, zespolone lub tekstowe. Pojedyncza liczba lub znak jest traktowany jak macierz o jednym wierszu i jednej kolumnie. Zmienne w Scilab-ie nie są deklarowane. Zmienna w momencie jej wpisania/wyliczenia jest zapisywana (*Przeglądarka zmiennych*) i jest dostępna do dalszych działań pod zdefiniowaną podczas wpisania/wyliczenia nazwą. Typ zmiennej jest automatycznie rozpoznawany przez Scilab. W nazwach zmiennych rozróżniane są duże i małe litery (x oraz X to dwie różne zmienne).

Podstawowy interfejs użytkownika zawiera cztery okna (rys.1):

- ***Konsola/Console*** (główne okno do wpisywania poleceń, wyświetlania wyników obliczeń numerycznych oraz komunikatów programu). Wyniki prowadzonych obliczeń/działań wyświetlane w *Console* są jedynie tzw. 'echem' działania programu, mają więc jedynie charakter informacyjny. Formalne wyniki są zapisywane w pamięci jako nowo utworzone zmienne i są widoczne/dostępne w *Przeglądarce zmiennych* pod odpowiednią nazwą. Zawartość *Konsoli* znajdująca się powyżej znaku zachęty (-->) nie może być edytowana. Nie można więc wprowadzać poprawek do polecenia, które zostało już wykonane w programie. Jeśli polecenie zawierało błąd, należy wpisać je ponownie, bez błędu. Wyczyszczenie zawartości *Konsoli* (polecenie *clc*) nie usuwa wyników przeprowadzonych obliczeń. Wpisując na nowo poprawione polecenie, które zawierało błąd można skorzystać z *Historii poleceń* przenosząc polecenie z błędem do *Konsoli* i poprawiając błąd przed zaakceptowaniem polecenia. Szybki dostęp do poleceń z *Historii poleceń* można też uzyskać wciskając klawisz kierunkowy (strzałka w górę).
- ***Przeglądarka plików*** (przeglądarka zorganizowana w postaci drzewa katalogów z ustawioną ścieżką do katalogu bieżącego, w którym są zapisywane pliki utworzone podczas pracy z programem),
- ***Przeglądarka zmiennych*** (okno z wyszczególnionymi zmiennymi z podaniem nazwy i typu zmiennej),
- ***Historia poleceń*** (historia poleceń wpisywanych w *Konsoli* podczas bieżącej sesji programu, a także podczas sesji historycznych – z podziałem wg dat sesji).

Dodatkowo uaktywnione mogą być dodatkowe okna:

- ***edytor*** (narzędzie programistyczne do pisania programów- funkcji, skryptów) wywoływany poleceniem ***edit*** wpisanym w *konsoli,* lub wybierając ikonę  na pasku narzędzi przy aktywnym oknie Console,
- ***okno graficzne*** (służące do wyświetlania grafiki np. wykresów),
- ***okno pomocy*** (wbudowana pomoc do programu) uaktywniana poleceniem *help* .

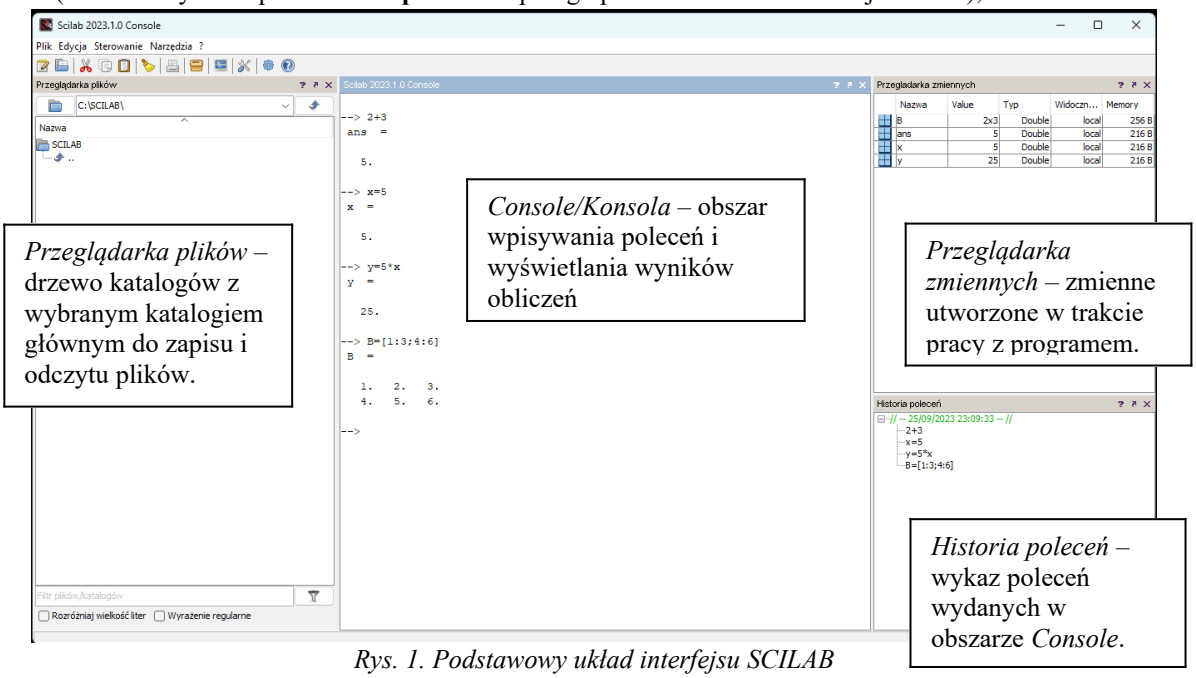
Interfejs użytkownika może być dowolnie konfigurowany. Każde z okien można „oddokować” z głównego okna SCILAB, zmienić jego położenie lub zamknąć jeśli nie będzie nam w danej sesji potrzebne. W każdym czasie możemy przywrócić podstawowy układ okien interfejsu wybierając polecenia z menu górnego *Edycja-Preferences* a następnie w pojawiającym się oknie (rys.2) należy wybrać *General-Desktop layout-Reset layout*. Przywrócenie podstawowego układu okien nastąpi po ponownym uruchomieniu programu.

UWAGA: w danym momencie aktywne jest jedno z podstawowych okien (okno aktywne ma pasek tytułowy w kolorze jasno-niebieskim, okna nieaktywne mają paski tytułowe w kolorze szarym). Na rys.1. aktywnym oknem jest *Console.* Dane okno jest aktywowane po kliknięciu w jego obszarze. Należy zwrócić uwagę, że wraz ze zmianą aktywnego okna zmienia się zawartość górnego menu tekstowego oraz paska narzędzi. „Chwytając” wskaźnikiem „myszy” za pasek tytułowy możemy zmieniać położenie poszczególnych okien. Po prawej stronie paska tytułowego znajdziemy 3 przyciski służące do wywołania okna pomocy, „oddokowania” okna oraz zamknięcia okna (rys.3).

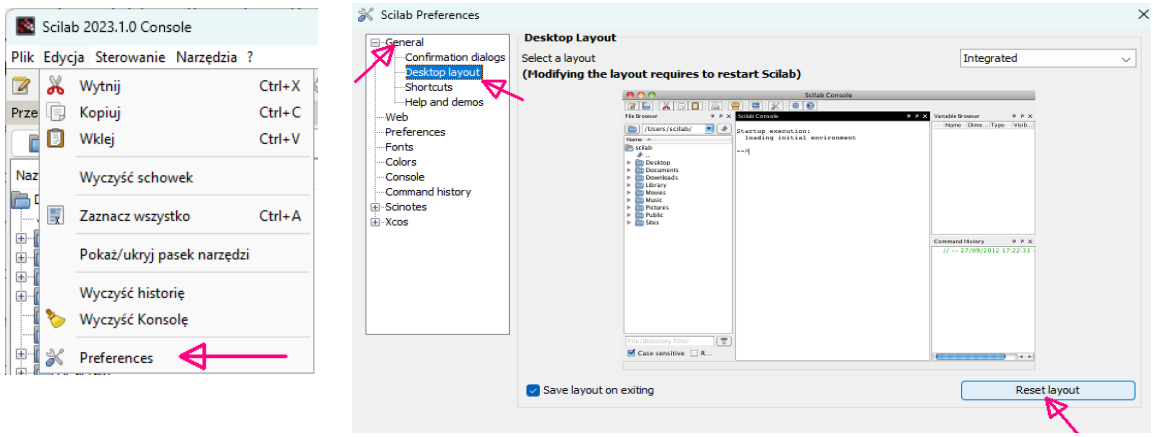
SCILAB jest wyposażony we budowane elementy informacyjne, którymi są: *Przeglądarka pomocy*, *Demonstracje* oraz *Komunikaty błędów*.

Przeglądarka pomocy:

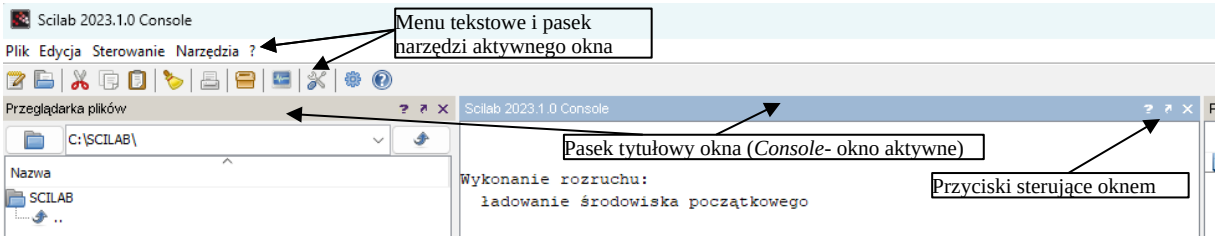
Dostęp do przeglądarki pomocy można uzyskać z wielu okien wybierając jeden z przycisków na pasku tytułowym . Jednak podstawowym sposobem wywoływania będzie polecenie **help** *temat* wpisywane w *Konsoli*, co spowoduje otwarcie okna *Przeglądarki pomocy* zawierającego szczegółowe objaśnienia dotyczące słowa *temat* (warto korzystać z polecenia **help** w celu lepszego poznania działania funkcji Scilaba);



Rys. 1. Podstawowy układ interfejsu SCILAB

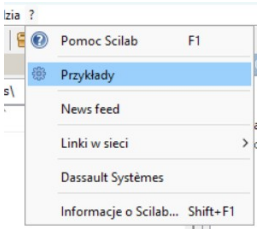


Rys. 2. Przywracanie podstawowego interfejsu SCILAB



Rys. 3. Elementy okien interfejsu SCILAB

Demonstracje:



Dostęp do przykładów demonstracyjnych programu SCILAB jest możliwy z menu tekstowego okna *Console* wybierając symbol ' *?* ' a następnie '*Przykłady*'.
W dodatkowym oknie można będzie wybrać przykłady realizacji różnorodnych działań z wykorzystaniem funkcjonalności programu.

Komunikaty błędów:

Komunikaty wyświetlane w oknie *Console.* Zakres komunikatu jest dość podstawowy, jednak warto zwracać uwagę na ich treść, szczególnie podczas kompilacji i uruchamiania zaprogramowanych funkcji i skryptów. Poniżej pokazano przykład komunikatu o błędzie składni wpisanego polecenia. Jak widać oprócz komunikatu o rodzaju błędu wskazywane jest też sugerowane miejsce w poleceniu gdzie ten błąd wystąpił.

```
x=5=-3
   ^^
Error: syntax error, unexpected =, expecting end of file
```

Wybrane polecenia i elementy środowiska Scilab:

--> 'znak zachęty' wiersza poleceń w oknie *Console* – miejsce gdzie można wpisywać polecenia, które będą realizowane w programie.

W dalszej części instrukcji po 'znaku zachęty' zapisane są przykłady odnoszące się do omawianych zagadnień. Przykłady te należy wpisać do 'Console' i zaobserwować uzyskane wyniki.

Nawiasy [ ] : tworzenie tablic (wektorów i macierzy), elementy wierszy macierzy oddzielone są spacjami lub przecinkami, zaś wiersze kończą się średnikami (wpisując elementy macierzy z klawiatury, wiersze można zakończyć wciśnięciem klawisza ENTER);

--> A=[2 2 2 1; 1 2 3 1] lub: --> A=[2,2,2,1;1,2,3,1] lub: --> A=[2 2 2 1
1 2 3 1]

Nawiasy ( ) :

określenie kolejności wykonywania działań matematycznych: -->x=(2+6)*7
przy wywoływaniu funkcji nawiasy () otaczają listę argumentów funkcji: -->sin(%pi/4)
oznaczanie indeksów wektorów i tablic: -->A(2,3)

Kropka . : oddzielenie części ułamkowej liczby dziesiętnej. Kropka umieszczona przed operatorami: * , ^ , /, \ oznacza działania na kolejnych elementach tablic. Dwie lub więcej kropek nas końcu linii oznacza, że następną linię należy traktować jako kontynuację poprzedniej.

Przecinek , Oddziela indeksy tablic, elementy jednego wiersza oraz argumenty funkcji.

Średnik ; :oddziela kolejne wiersze macierzy, umieszczony na kończy wyrażenia powoduje, że wynik wyrażenia nie jest wyświetlany na ekranie, następuje zapis wyniku do *Przeglądarki zmiennych*.

Znak // : tekst po znaku // aż do końca linii jest traktowany jako komentarz.

Dwukropek : : Służy do konstrukcji wektorów o elementach liniowo narastających lub malejących

-->X=1:10
-->X1=1:2:10
-->X2=20:-3:0

clc : czyści ekran tekstowy;
-->clc

clear *x* usuwa zmienną *x* z pamięci, polecenie **clear** bez wyszczególnienia zmiennej usuwa z pamięci wszystkie zmienne.
-→clear X

who : wyświetla listę zmiennych obecnych w pamięci;

whos : wyświetla listę zmiennych obecnych w pamięci oraz ich wymiar.

plot2d : tworzenie wykresów (podstawowe polecenie tworzenia wykresów w SCILAB), stosuje się rożną składnię polecenia np.: *plot2d(wektor1,wektor2,styl)* wyświetli wykres o współrzędnych (*x,y*) określonych przez wektory: (*wektor1,wektor2*), parametr *styl* jest liczbą całkowitą. Różne wartości dodatnie tego parametru umożliwiają wybór koloru linii wykresu zaś różne wartości ujemne umożliwiają wybór typu znacznika punktów wykresu.

Aby zobaczyć inne typy składni polecenia warto skorzystać z *Okna pomocy*:
--> help plot2d

W *Oknie pomocy* mamy podany formalny opis polecenia plot2d oraz szereg przykładów wykresów uzyskanych z zastosowaniem różnych wariantów polecenia.

plot : tworzenie wykresów (polecenie wg składni Matlaba) , w zależności jakie wykresy mają zostać wyświetlone stosuje się rożną składnię polecenia np.: *plot(wektor1,wektor2,'kolor rodzaj_linii_lub_markera')* wyświetli wykres o współrzędnych (*x,y*) określonych przez wektory: (*wektor1,wektor2*) ,w kolorze określonym przez *kolor* oraz z rodzajem linii określonym przez *rodzaj_linii_lub_markera.*

Rodzaje linii:

- \- ciągła;
- -- przerywana;
- : kropkowa;
- -. typu „kropka-kreska"

Rodzaje markerów: . + * x o

Kolory: r-czerwony, g-zielony, b-niebieski, w-biały, i-kolor tła (niewidoczny).

Podane powyżej rodzaje linii, markerów i kolorów nie wyczerpują pełnego ich katalogu. Aby dowiedzieć się więcej zastosuj polecenie *Help* i skorzystaj z *Przeglądarki pomocy.*

diary('nazwaPliku') powoduje, że wszystkie polecenia i odpowiedzi zapisywane są do wskazanego pliku. Rejestrowanie kończymy poleceniem diary(0). Plik zostanie utworzony w katalogu bieżącym, polecenie pwd (print work directory) wyświetli nazwę tego katalogu. Istniejący plik o takiej samej nazwie zostanie, bez uprzedzenia nadpisany.

edit : otwiera okno edytora.

format *postać_liczby*: określenie sposobu, w jaki liczby rzeczywiste są przedstawione na ekranie, gdzie *postać_liczby* to: *e-zapis inżynierski, v-zapis liczbowy.*

Przykład:

Przedstaw liczbę 2,5 w różnej postaci używając funkcji format.
--> *format v*
--> 2.5
--> *format e*
--> 2.5

Najważniejsze funkcje matematyczne:

sin( ), cos( ), tan( ), cotg( )	Funkcje trygonometryczne
asin( ), acos( ), atan( )	Funkcje cyklometryczne - arkfunkcje
sinh( ), cosh( ), tanh( ), coth( )	Funkcje hiperboliczne

exp()	Funkcja exponent *exp(x)=e^x*
log( ) , log10( )	Logarytm naturalny, logarytm o podstawie 10
abs( )	Wartość bezwzględna
sqrt( )	Pierwiastek kwadratowy
round( )	Przybliżenie całkowite (zaokrąglenie)
sum( )	Sumowanie
min( ), max( )	Minimum, maksimum
real( ), imag( )	Rzeczywista, urojona część liczby zespolonej
floor( )	Przybliżenie całkowite z 'dołu' *floor(2.3)=2*
ceil( )	Przybliżenie całkowite z 'góry' *ceil(2.3)=3*
int( )	Obcięcie części ułamkowej *int(-3.4)=-3*
rand( )	Liczba (macierz) losowa, losowane są liczby z przedziału [0,1]. rand() = liczba (losowa) rand(5,4) macierz 5x4 (o losowych elementach)

Najważniejsze predefiniowane zmienne i stałe systemowe :

%i	Jednostka urojona *i=√-1*
%pi	Liczba Pi
%e	Podstawa logarytmu naturalnego
%eps	Dokładność obliczeń eps=2.22*10^{-16} może zależeć od możliwości procesora
%inf	Nieskończoność (jako wynik dzielenia przez zero)
%nan	Nit a Number - wynik nie jest liczbą
%s, %z	Zmienne wielomianowe
%t, %T	true/prawda /logiczne '1'– zmienna logiczna
%f, %F	false/fałsz /logiczne '0'– zmienna logiczna

Wykonać polecenia: (kolejność wg kolumn tabeli) i przeanalizować uzyskane wyniki.

-->1+2	-->%pi	-->d
-->1e2	-->sin(%pi/2)	-->z1=1+2%i
-->1e3+2e5	-->cos(%pi)	-->z2=10%i
-->log(2)	-->sin(2*%pi/4)	-->real(z1)
-->log(2.7183)	-->2*sin(%pi/4)*cos(%pi/4)	-->imag(z1)
-->exp(1)	-->sin(0.2)^2+cos(0.2)^2	-->sqrt(9)
-->log(exp(1))	-->a=8.3	-->sqrt(-1)
-->log10(2.7183)	-->a=8.8;	-->%i^2
--> log10(10)	-->a	-->sqrt(-4)
-->2*(3+4)	-->b=2+4	-->1/0
-->(5-3)/(4-3*1)	-->a	-->0/0
-->3^3	-->b	-->f=a+b+c-d
-->8^(1/3)	-->c=a+b	-->g=sqrt(f)
-->2^3	-->d=c-a;	--> sqrt(f)

Napisać polecenia wyliczające wyrażenia:

$\sin\left(\frac{\pi}{5}\right)$	$\frac{\sqrt{\left(3+e^3\right)}}{3+\cos\left(\sqrt{2+\pi}\right)}$	$\sqrt{2+e^5}$	$\sqrt[3]{\left\lvert e-\frac{3}{2}\pi\right\rvert}$	$\frac{\sqrt[3]{2\pi+\pi^4}}{\left(\sin\left(1-\frac{1}{\pi}\right)\right)^2}$
0.5877853	2.0373741	12.264304	1.2586824	11.832808

UWAGA !
Na początku każdych zajęć ustawić własny katalog bieżący w oknie *Przeglądarka plików***. Powinien to być podkatalog katalogu** *Nazwa_Grupy* **i we własnej nazwie powinien zawierać nazwisko studenta. Na pierwszych zajęciach należy takie katalogi utworzyć. Kończąc zajęcia należy „wyczyścić" okna Scilaba** ***Console*** **oraz** ***Historia poleceń*** **(w razie potrzeby uprzednio zapisując dane z** *Workspace* **).**
Pierwszym poleceniem jakie należy wydać 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 wszystkich poleceń wpisywanych w trakcie zajęć, a także uzyskiwanych wyników.**
Następnie w '*Console*' należy wpisać linię komentarza z imieniem, nazwiskiem i datą:
--> //Imię, Nazwisko, Data

Praca w środowisku języka 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***. Podstawowym typem danych Scilaba jest tablica. Elementami tablicy mogą być liczby rzeczywiste, zespolone, znaki lub inne tablice. Szczególnym przypadkiem tablicy (tablica dwuwymiarowa) jest macierz. Działanie Scilaba oparto na operacjach wektorowo-macierzowych W programie nie występują zmienne skalarne. Pojedyncze liczy/znaki są traktowane jako macierz o jednym wierszu i jednej kolumnie. Wektor jest traktowany jako macierz kolumnowa lub wierszowa. Poszczególne elementy wiersza macierzy oddziela się spacjami lub przecinkami, a wiersze średnikami lub umieszcza się je w oddzielnych liniach.

-->A = [11 12 13; 21 22 23; 31 32 33]

-->A1= [11, 12, 13; 21, 22, 23; 31, 32, 33]

Macierz 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 mysi 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
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]'
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.
-->M2=linspace(6,1,6)
Analogicznie można utworzyć wektor kolumnowy o elementach rozłożonych liniowo (równomiernie):
-->M2k=linspace(5,1,5)'
Rozdzielając wektory średnikami, lub zapisując je w oddzielnych liniach można zdefiniować macierz.
-->MA=[WIERSZ;M1;M2]
MA =

0. 0.2 0.4 0.6 0.8 1.	→ WIERSZ
100. 101. 102. 103. 104. 105.	→ M1
6. 5. 4. 3. 2. 1.	→ M2

Przykład:
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 tez wygenerowanie macierzy korzystają z innych macierzy.
Przykład:
Macierz **D z**budowana 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).
Wymiar i wyświetlanie macierzy
• [m,n]=size(A) - zwraca liczbę kolumn *n* i wierszy *m* macierzy **A**;
• n=length(B) - zwraca wymiar wektora **B** (lub liczbę elementów macierzy **B**);
• A lub disp(A) - pokazuje macierz **A** na ekranie;
Zadania:
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;
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.

Funkcje wspomagające konstruowanie macierzy
Macierz jednostkowa (*eye*):
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:
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: 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: 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.

Podstawowe metody wyboru elementów macierzy:
A(i,j) – wypisanie elementu z i-tego wiersza i j-tej kolumny macierzy A;
A(:,j) – wypisanie j-tej kolumny macierzy A;
A(i,:) – wypisanie 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.

Przykłady:
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:$)
-->A(1,1)	-->A(1)	-->A(1,3)
-->A(1,2)	-->A(2)	-->A(3,1)
-->A(:,1)	-->A(6)	-->min(A)
-->A(:,3)	-->A(7)	--> max(A)
-->A($,:)	-->A(1:10)	-->A(:)

Złożone metody wyboru elementów macierzy:
Wykonać polecenia i przeanalizować ich wyniki: (polecenia wpisywać kolumnowo)
-->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)=[ ]

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 wykona 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, macierze muszą mieć takie same rozmiary. Dla mnożenia macierzowego, pierwsza macierz powinna mieć tyle kolumn ile druga ma wierszy, wynik ma tyle wierszy, ile macierz pierwsza i tyle kolumn ile druga.
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

Operatory arytmetyczne działań macierzowych i tablicowych:

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

Wykonać działania i przeanalizować wyniki:

Utworzyć macierze:

A =

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

B =

1. 1. 1.
2. 2. 2.
3. 3. 3.

Wykonać działania:

-->A+1	-->A.*B	-->A+B	-->B^2
-->A*2	-->A*B	-->B/2	-->B.^2
-->A’	-->B’	-->A-A	-->B./B

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

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

Zadanie 1

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

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 3

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.
```