Laboratorium Metod Numerycznych
Wprowadzenie do środowiska Scilab
Laboratorium 1

Metody numeryczne są dziedziną matematyki zajmującą się rozwiązywaniem problemów matematycznych na liczbach. Wyniki otrzymywane w efekcie działania określonych procedur obliczeniowych zazwyczaj osiągane są z pewnym przybliżeniem, determinowanym przez z góry założoną dokładnością.

Metody numeryczne znalazły szerokie zastosowanie przy analizie i rozwiązywaniu problemów inżynierskich i naukowych, szczególnie w sytuacjach gdy przedmiot badań nie ma rozwiązania analitycznego, określonego równaniem, lub jego określenie jest czasochłonne lub skomplikowane.

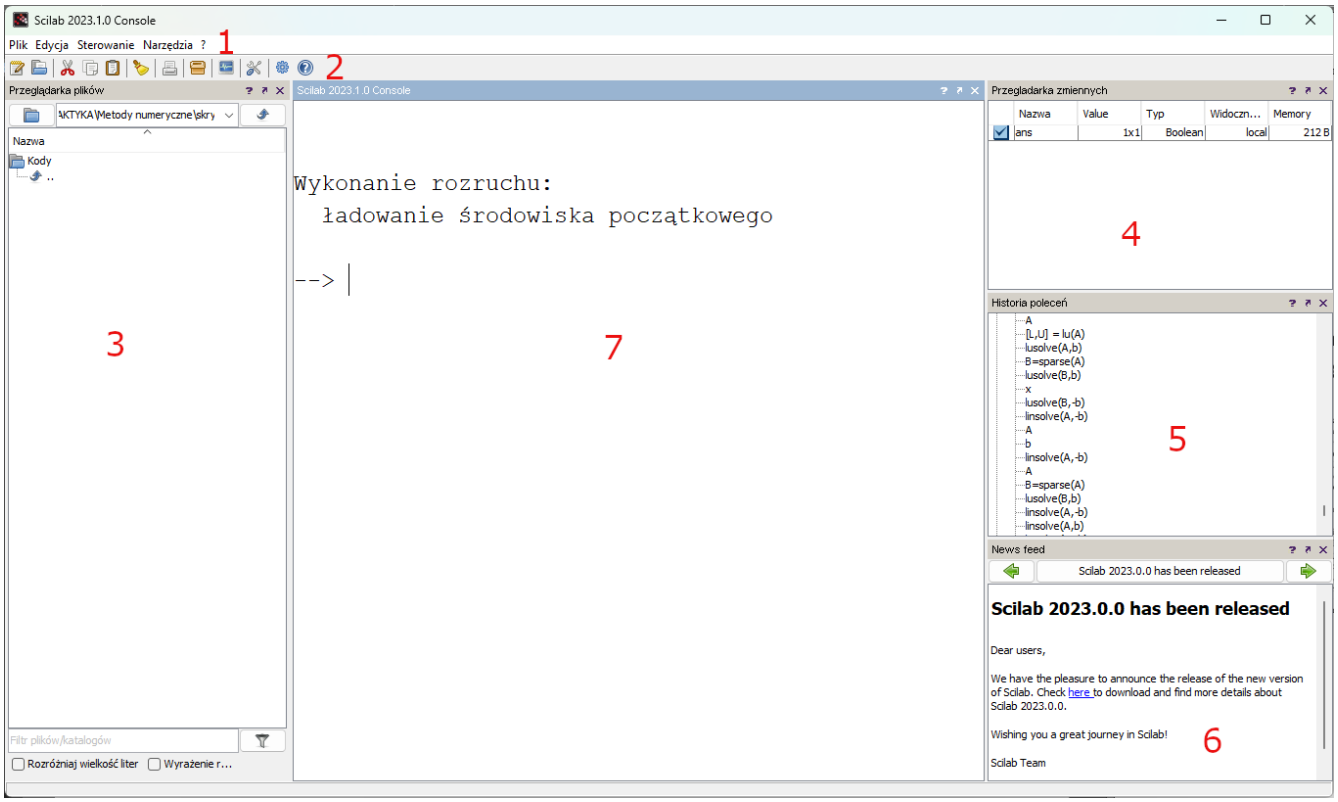
Zaprogramowanie i przeprowadzenie symulacji badanego obiektu lub zjawiska możliwe jest praktycznie w każdym języku programowania, jednakże znacznie wygodniejsze i efektywniejsze jest rozwiązanie danego problemu w dedykowanym środowisku obliczeniowym jak Matlab, Mathcad, Maple, Maxima czy Scilab. [1, 2]

program laboratorium metod numerycznych realizowany dla studentów kierunku Elektrotechnika na Wydziale Elektrotechniki i Informatyki realizowany jest w oparciu o środowisko obliczeniowe Scilab. [3] Jest to środowisko dystrybuowane na zasadach Open Source i w znacznym stopniu wzorowane na Matlab'ie. Możliwe jest importowanie kodu napisanego w Matlab'ie. Projekt jest rozwijany bardzo dynamicznie przez środowisko programistów projektu i społeczność. Jest on dostępny na wszystkie obecnie stosowane powszechnie systemy operacyjne.

Interfejs użytkownika pokazany na Rys.1 składa się z kilku elementów. Pozycje *Plik* i *Edit* zawierają standardowy zestaw skrótów, dla pracy w programie istotniejsze są zakładka *Sterowanie* zawierając zestaw poleceń zarządzających wykonywanie skryptów (przerwania, zatrzymanie i wznowienie), oraz *Narzędzia* z skrótami do podprogramów środowiska:

- **SciNotes** - edytor tekstowy dedykowany do tworzenia skryptów (programów obliczeniowych) dla środowiska ***Scilab***
- **XCOS** - moduł programowania graficznego, odpowiednik Simulink's z środowiska Malab'a,
- **ATOM** - interfejs zarządzania modułami rozszerzeń środowiska ***Scilab*** i ***XCOS***

- Moduł translatora projektów Matlab'a do środowiska ***Scilab***



Rys. 1: Interfejs programu Scilab (**1** - menu głowne, **2** - pasek skrótów, **3** - przeglądarka plików, **4** - okno podglądu zmiennych, **5** - historia poleceń, **6** - okno informacyjne, **7** - okno konsoli).

Konsola (7) jest głównym oknem programu umożliwiającym bezpośrednia prace na silniku obliczeniowym środowiska. Możliwe jet wprowadzanie pojedynczych poleceń lub ich sekwencji, a wyniki wyświetlane są w konsoli bezpośrednio po zatwierdzeniu linii polecenia. Jest ona bezpośredni sprzężona z oknami pomocniczymi, przeglądarką zmiennych (4) i historią (5). Konsola wspiera korzystanie z standardowych skrótów klawiaturowych (*wytnij*, *kopiuj* i *wklej*) oraz szybkiego dostępu do historii za pomocą klawiszy strzałek (góra i dół).

Polecenia środowiska ***Scilab***'a można podzielić na pięć głównych grup:

1. polecenia systemowe: *help*, *clear*, *clc*, *dir*, itp.
2. podstawowe działania i funkcje matematyczne: dodawanie, odejmowanie, dzieleni, mnożeni, *sin*, *cos*, *exp*, *sqrt* itp.
3. działania i funkcje rachunku macierzowego: *inv*, *det*, *diag*, *max* itp.
4. funkcje numerycznych: *Ode*, *solve*, *linsolve*, itp.
5. funkcje graficzne i tekstowe: *plot2d*, *plot3d*, *disp*, itp.

W ćwiczeniu omówione zostaną wybrane elementy spośród pierwszych trzech grup poleceń. Każdy wiersz wprowadzony konsoli zostanie wykonany a jego efekt wyświetlony poniżej. Jednakże zakończenie wiersza znakiem średnika ";" spowoduje zablokowanie wyświetlania wyniku działania w konsoli.

1) Polecenia systemowe

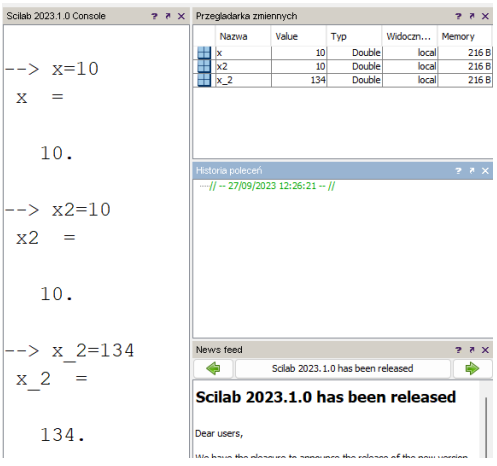
Polecenia z tej grupy umożliwiają zarządzanie praca konsoli i współpracą z innymi elementami środowiska ***Scilab***.

Podstawowym narzędziem wsparcia w trakcie pracy w środowisku obliczeniowym ***Scilab*** jest pomoc, baza danych z omówieniem środowiska, jego dokumentacja. Dostęp do niej możliwy jest poprzez interfejs programu oraz wywołanie odpowiedniego polecenia w konsoli:

- `help` - aktywacja okno pomocy
- `help print3d` - wyświetlanie okna pomocy dotyczącego podanego polecenia, przykładowo funkcji print3d
- `help("differential equations")` - wyświetlenie okna pomocy i pozycji w bazie danych skojarzonych z zadanym tematem

Korzystanie z pomocy pozwala na szybkie odnalezienie narzędzi (funkcji) niezbędnych do rozwiązania analizowanego problemu obliczeniowego i programistycznego. Pomimo pełnego spolszczenia interfejsu programu ***Scilab***, pomoc na chwilę obecną dostępna jest tylko w języku Angielskim.

W trakcie pracy w konsoli może pojawić się konieczność wyczyszczenia jej okna. Można to zrealizować z poziomu menu podręcznego (prawy przycisk myszki w oknie konsoli) lub za pomocą polecenia `clc`. Instrukcja ta pozwala na ograniczenie czyszczenia okna konsoli do określonej liczby ostatnio wprowadzonych poleceń `clc(liczba);`.



Rys. 2: Zmienne

Podstawowym narzędziem stosowanym trakcie obliczeń realizowanych w konsoli są zmienne. W środowisku ***Scilab*** zmienną definiuje się poprzez jej inicjacje, jej typ określany jest na podstawie typu przypisywanej wartości. W Środowisku ***Scilab*** stosowane są trzy typy do opisu zmiennych: **logiczny** (*Boolean*), **tekstowy** i **liczbowy** (*Double*). Typ Double jet także stosowany dla zmiennych zespolonych. Nazwa zmiennej może być dowolna, składająca się z dowolnych znaków (litery, cyfry, inne znaki), ale musi rozpoczynać się od litery: `x=10`, `x2=10`, `x_2=10`, jak pokazano na Rys.2. Nazwa może być taka sama jak już dostępna w środowisko, jednak wtedy przykrywa

ta już stosowaną. Może to z oczywistych względów spowodować problemy. Utworzenie zmiennej `sin=10` spowoduje że nie będzie możliwe wywołanie funkcji sinusa (`sin(20)`), system zwróci błąd. ***Scilab*** rozróżnia stosowaną w nazewnictwie zmiennych wielkość liter, a więc zmienna `Alfa=10` i `alfa=5` to dwie różne zmienne.

Kontrola nad istniejącymi zmiennymi możliwa jest poprzez okno podglądu zmiennych i jej interfejs podręczny lub za posłannictwem poleceń sprawdzenia istnienia zmiennej i kasowania pamięci.

Funkcja `isdef("zmienna")` zwraca informacje w postaci wartości logicznej `true` lub `false` w zależności od istnienia w pamięci zmiennej podanej jako atrybut polecenia. jak pokazano poniżej.

```
--> isdef("A")
ans =
      F
--> A=23;
--> isdef("A")
ans =
      T
```

Zmienne można skasować za pomocą polecenia `clear`, które po wywołaniu czyści pamięć programu z wszystkich zdefiniowanych zmiennych. Możliwe jest jednak także selektywne kasowanie wybranych zmiennych, jak pokazano poniżej.

```
--> A=20; B=30; C=100;
--> clear A;
--> A
Undefined variable: A
--> clear("B","C")
--> B
Undefined variable: B
```

Informacje wyświetlane w konsoli można prezentować w sposób sformatowany, jest to szczególne istotne przy prezentacji wyników obliczeń realizowanych za pomocą zaprogramowanych skryptów. Można to zrealizować za pomocą dwóch funkcji:

- `disp()` - funkcja wyświetla ciąg tekstowy lub zawartość zmiennej na ekranie. Ciąg tekstowy może być budowany z wykorzystaniem funkcji konwersji wartości liczbowych do ciągu tekstowego `string()`, jak pokazano poniżej:

  ```
  --> A=23; B=23;
  --> disp(A);
     23.
  --> disp("Zmienna A="+string(A)+", a zmienna B="+string(B));
    "Zmienna A=23, a zmienna B=23"
  ```

- `mprintf()`- funkcja w swojej składni i funkcjonalności analogiczna do znanej z języka `C` funkcji `printf()`. Wyświetla sformatowany ciag tekstowy na ekranie konsoli. Pozwala na wstawienie w budowany ciąg zawartości zmiennych, wartości zwracanych przez funkcje lub wyrażenia, jak pokazano poniżej.

```
--> A=20; B=30;
--> mprintf("Suma liczb %f i %f wynosi %f",A,B,A+B);
Suma liczb 20.000000 i 30.000000 wynosi 50.000000
--> mprintf("Suma liczb %.0f i %.0f wynosi %.0f",A,B,A+B);
Suma liczb 20 i 30 wynosi 50
--> mprintf("Suma liczb %i i %i wynosi %i",A,B,A+B);
Suma liczb 20 i 30 wynosi 50
```

Dane liczbowe prezentowane są na dwa sposoby:

- algebraiczna - za pomocą maksymalnie 10 znaków, z uwzględnieniem znaku liczby ($\pm$) i kropki oddzielającej cześć ułamkową,
- potęgową - postać zmiennopozycyjna z podstawa potęgowa 10.

Dobór sposoby wyświetlania liczby realizowany jest automatycznie i zależy o jej wielkości. Możliwe jest jednak ręczne wymuszenie prezentacji wartości liczbowej za pomocą funkcji `format(p1,p2)`. Ma ona dwa atrybuty wejściowe, pierwszy określa wybór formy (`"v"` - algebraiczny, `"e"` - potęgowy), a drugi określa liczbę znaków dedykowaną do wyświetlenia oknie konsoli. Należy zwrócić uwagę że nie zawsze narzucone formatowanie jest realizowane. Nadrzędnym kryterium jest wyświetlenie danych z jak największym zasobem informacji, co widać w poniższym przykładzie przy zbyt dużym ograniczeniu liczby znaków w trybie algebraicznym. ***Scilab*** automatycznie przełączył na ryb potęgowy.

```
--> A=-123.123456789
 A  =
  -123.12346
--> format("v",15)
--> A
 A  =
  -123.123456789
--> format("e",12)
--> A
 A  =
  -1.23123D+02
--> format("v",4)
--> A
 A  =
```

```
  -1.D+02
--> format("v",5)
--> A
 A  =
  -123.
```

Ostatni zestaw funkcji w tej upie stanowią polecenia operacji plikowych. Pozwalają na realizacje operacji na folderach i plikach. Znajdują one zastosowanie przy realizacji zadań związanych z importowaniem danych lub funkcji z plików zewnętrznych. Do grupy tej mozna między innymi zaliczyć:

- **chdir** - zmiana aktualnego katalogu
- **isdir** — sprawdza czy podany katalog istnieje
- **pwd** — zwraca aktualny katalog
- **home** — zwraca domyślny katalog użytkownika
- **filebrowser** — otwarcie okna dialogowego otwarcia pliku

2) Podstawowe działania matematyczne

W środowisku obliczeniowym ***Scilab*** obliczenia wykonywane są na wartościach liczbowych (rzeczywistych i zespolonych), aczkolwiek możliwe jest także realizacja rachunku symbolicznego, po doinstalowaniu wymaganego plugin'a. Podobnie jak większości środowisk obliczeniowych miarę kata należy wyrażać w radianach, aczkolwiek dostępnych jest szereg funkcji które operują na mierze stopniowej kąta. Część ułamkowa należy oddzielać za pomocą znaku kropki ".", przecinek stosowany jet do oddzielenia atrybutów funkcji. W środowisku zdefiniowano szereg stałych:

- liczba π - %pi
- stała e - %e
- nieskończoność ∞ - %inf
- operator i części urojonej liczby zespolonej - %i

Podstawowe operacje matematyczne realizuje się za pomocą standardowych znaków +,-,*,\ i ^ (potęga) i nawiasów organizujących kolejność działań, jak pokazano poniżej.

```
--> (23-4^(7/34))*(12+123)
 ans  =
   2925.407872
```

Tab. 1: Wybrane funkcje matematyczne.

Lp.	Funkcja	Przykład	Opis

1	`sin(x), sind(x)`	`--> sin(%pi/3)`
		`ans = 0.8660254`
		`--> sind(60)`
		`ans = 0.8660254`
			Wyznaczenie sinusa kąta radianach lub stopniach.

2	`log(x)`	`--> log(10)` `ans = 2.3025851`	Logarytm naturalny liczby
3	`log10(x)`	`--> log10(100)` `ans = 2.`	Logarytm dziesiętny liczby
4	`sqrt(x)`	`--> sqrt(1256)` `ans = 35.44009`	Pierwiastek kwadratowy
5	`exp(x)`	`--> exp(2)` `ans = 7.3890561`	Potęga wykładnicza

Poza działaniami podstawowymi w środowisku zdefiniowanych jest szereg funkcji matematycznych. Wybrane z nich zestawiono w tabeli 1.

Pierwiastki i potęgi poza pierwiastkiem kwadratowym należny realizować z wykorzystaniem operatora "^". Działania mogą być wykonywane zarówno dla liczb rzeczywistych jak i zespolonych, jak pokazano poniżej.

```
--> x=2.13;y=5-3*%i;
--> x^3
 ans  =
   9.663597
--> ans^(1/3)
 ans  =
   2.13
--> y^5
 ans  =
  -6100. - 2868.i
--> ans^(1/5)
 ans  =
   5. - 3.i
```

Scilab dostarcza także szereg funkcji dedykowanych do przetwarzania liczb zespolonych, wybrane z nich zestawiono w tabeli 2.

Tab. 2: Wybrane funkcje dla liczb zespolonych.

Lp.	Funkcja	Przykład	Opis

1	`complex(a,b)`	`--> a=20;b=-35;`
		`--> c=complex(a,b);`
		`--> c`
		` c  =`
		`  20. - 35.i`
			Tworzy zmienną zespoloną na podstawie dwóch wartości rzeczywistych

2	`conj(Z)`	`--> conj(c)`
		` ans  =`
		`  20. + 35.i`
			Zwraca liczbę sprzężoną do wejściowej liczby Z

3	`isreal(Z)`	`--> isreal(c)`
		` ans  =`
		`  F`
		`--> isreal(a)`
		` ans  =`
		`  T`
			funkcja tstowa, zwraca wartość true jeżeli liczba jest rzeczywista

4	`real(Z)`	`--> real(c)`
		` ans  =`
		`  20.`
			Zwraca część rzeczywistą wejściowej liczby zespolonej Z

5	`imag(Z)`	`--> imag(c)`
		` ans  =`
		`  -35.`
			Zwraca część urojoną wejściowej liczby zespolonej Z

Problematyczne jest określanie argumentu liczby zespolonej w postaci wykładniczej. Nie ma dedykowanej do tego funkcji, konieczne jest wyznaczenie kata oparciu o funkcje trygonometryczne, zgodnie z poniższym wzorem.

$\varphi = atg\left(\frac{Im(Z)}{Re(Z)}\right)$

W środowisku ***Scilab*** wartość a można wyznaczyć w stopniach jak i radianach, jak pokazano poniżej.

```
--> atan(imag(c)/real(c))
 ans  =
  -1.051650213
--> b=-c;
--> atan(imag(b)/real(b))
 ans  =
  -1.051650213
```

Należy jednak takim wypadku zauważyć, że przy wyznaczeniu kąta tracona jest informacja przynależności znaku do cześć rzeczywistej i urojonej. W takim wypadku

lepiej jest skorzystać z innej postaci funkcji `atan()` i `atand()` dedykowanej do wyznaczania kąta na podstawie elementów liczby zespolonej, jak pokazano poniżej.

```
--> atan(imag(c),real(c))
 ans  =
  -1.051650213
--> atan(imag(b),real(b))
 ans  =
   2.089942441
```

3) Rachunek macierzowy

Środowisko obliczeniowe ***Scilab*** podobnie jak ***Matlab*** jest dedykowane do rachunku macierzowego. Należy zwróci uwagę że większość problemów obliczeniowych rozwiązywanych za pomocą metod numerycznych oparta jest na rachunku macierzowym. ***Scilab*** zmienną traktuje jak macierz jednoelementową. Wektor jest macierzą jednokolumnowa lub jednowierszową. Macierz deklaruje się podając listę elementów w nawiasie kwadratowym, elementy wiersza rozdziela się przecinkiem, a wiersze średnikiem, jak pokazano poniżej.

```
--> A=[1,2,3]
 A  =
   1.   2.   3.
--> B=[1;2;3]
 B  =
   1.
   2.
   3.
--> C=[1,2,3;4,5,6;7,8,9]
 C  =
   1.   2.   3.
   4.   5.   6.
   7.   8.   9.
```

Elementy macierzy można deklarować wykorzystaniem rozkładów liniowego lub logarytmicznego. Rozkład liniowy definiuje się podając wartość początkową, przyrost i wartość końcową. Elementy definiując rozkład rozdziela się dwukropkiem, a wartość przyrostu można pominąć. W takim przypadku przyjmowana jest wartość domyślna równa 1. Poniżej pokazano przykład deklaracji macierzy za pomocą rozkładu liniowego.

```
--> A=[1:5;1:2:10]
 A  =
   1.   2.   3.   4.   5.
   1.   3.   5.   7.   9.
```

Należy zwracać uwagę aby liczba wygenerowanych elementów w obu wierszach była taka sama, w przypadku błędnie dobranych parametrów uzyskujemy różne liczby elementów w wierszach, a taka macierz nie może być utworzona, jak pokazano przykładzie poniżej.

```
--> A=[1:5;1:1.5:10]
inconsistent row/column dimensions
```

W pewnych sytuacjach wygodniejsze może się okazać zastosowanie funkcji generującej wektor rozkładu liniowego w zadanym przedziale wartości, składający się z określonej liczby elementów. Realizuje to funkcja `linspace(start,koniec,liczba)`, jak pokazano na poniższym przykładzie.

```
--> A=[linspace(1,5,5);linspace(1,10,5)]
 A  =
   1.   2.     3.    4.     5.
   1.   3.25   5.5   7.75   10.
```

Rozkład logarytmiczny realizowany jest przez funkcję `logspace(start,koniec, liczba)` w oparciu o rozkład potęgowy na bazie liczby 10 i przedział wartości wykładnika w zakresie od `start` i `koniec`. Parametr `liczba` określa liczbę elementów wygenerowanego wektora, jak pokazano poniżej.

```
 A  =
   10.    17.7827941    31.6227766    56.23413252    100.
   1.     1.122018454   1.258925412   1.412537545    1.584893192
```

Odwołanie do elementu macierzy realizowane jest poprzez podanie współrzędnych danego elementu. W przypadku wektora należy podać tylko jego pozycję indeksowaną od 1. Natomiast w przypadku macierzy podaje się numer wiersza i numer kolumny, jak pokazano poniżej.

```
--> A=[1,2,3,4];
--> A(3)
 ans  =
   3.
--> B=[1,2,3,4;4,3,2,1;5,6,7,8];
--> B(3,2)
 ans  =
   6.
```

Macierze w środowisku ***Scilab*** są skalowalne, to znaczy że do juz zainicjowanej macierzy można dodawać nowe elementy zwiększając jednocześnie jej rozmiar, jak pokazano poniżej. Elementy macierzy indeksowane są od jedynki (wiersze i kolumny).

```
--> A=[1:3]
 A  =
   1.   2.   3.
--> A(4)=12
 A  =
   1.   2.   3.   12.
```

W analogiczny sposób można dodawać nowe elementy do macierzy, jak pokazano poniżej.

```
--> A=[1:3;3:5]
 A  =
   1.   2.   3.
   3.   4.   5.
--> A(2,5)=-3
 A  =
   1.   2.   3.   0.   0.
   3.   4.   5.   0.  -3.
--> A(1,4)=-7
 A  =
   1.   2.   3.  -7.   0.
   3.   4.   5.   0.  -3.
```

Dodatkowe elementy które muszą powstać w celu zapienienia stabilności macierzy ustawiane są na wartość 0.

Środowisko ***Scilab***'a pozwala na odczytanie wybranego wiersza lub kolumny z macierzy. Realizuje się to poprzez wskazanie wymiaru, w drugim atrybucie wywołania podaje się znak dwukropka, jak pokazano poniżej.

```
--> C=B(2,:)
 C  =
   4.   3.   2.   1.
--> D=B(:,1)
 D  =
   1.
   4.
   5.
```

Skasowanie wybranego wiersza lub kolumny z macierzy realizuje się poprzez odwołanie się do wybranego elementu i przypisanie do niego pustego nawiasu kwadratowego, jak pokazano poniżej.

```
--> B
 B  =
```

```
   1.   2.   3.   4.
   4.   3.   2.   1.
   5.   6.   7.   8.
--> E=B;E(:,2)=[];E
 E  =
   1.   3.   4.
   4.   2.   1.
   5.   7.   8.
```

Możliwe jest także wyciągnięcie z macierzy wybranych wartości na podstawie określonego zakresu wierszy i kolumn. Czynność tę można także wykonać odwołując się do ostatniego elementu macierzy reprezentowane przez znak $, jak pokazano poniżej.

```
--> A=rand(4,5)
 A  =
   0.068374    0.1985144   0.2164633   0.9329616   0.2922267
   0.5608486   0.5442573   0.8833888   0.2146008   0.5664249
   0.6623569   0.2320748   0.6525135   0.312642    0.4826472
   0.7263507   0.2312237   0.3076091   0.3616361   0.3321719
--> A(1:2,1:2)
 ans  =
   0.068374    0.1985144
   0.5608486   0.5442573
--> A($-2:$,$-1:$)
 ans  =
   0.2146008   0.5664249
   0.312642    0.4826472
   0.3616361   0.3321719
```

W środowisku dostępnych jest kilka funkcji tworzących standardowe typy macierzy:

- macierz diagonalna - funkcja tworzy macierz diagonalną na podstawie wektora wejściowego umieszczając jego elementy na przekątnej diagonalnej

  ```
  --> A=diag(linspace(1,5,5))
   A  =
     1.   0.   0.   0.   0.
     0.   2.   0.   0.   0.
     0.   0.   3.   0.   0.
     0.   0.   0.   4.   0.
     0.   0.   0.   0.   5.
  ```

- macierz jednostkowa - funkcja tworzy macierz diagonalną o zadanym rozmiarze z jedynkami na przekątnej diagonalnej

```
--> B=eye(4,4)
 B  =
   1.   0.   0.   0.
   0.   1.   0.   0.
   0.   0.   1.   0.
   0.   0.   0.   1.
```

- macierz jedynkowa i zerowa - funkcje tworzą macierze o zadanym wymiarze wypełnione jedynkami lub zerami

```
--> A=ones(3,3)
 A  =
   1.   1.   1.
   1.   1.   1.
   1.   1.   1.
--> B=zeros(3,3)
 B  =
   0.   0.   0.
   0.   0.   0.
   0.   0.   0.
```

- macierz losowa - macierz o zadanym rozmiarze zapełniona wartościami losowymi z przedziału (0,1)

```
--> A=rand(3,3)
 A  =
   0.2113249   0.3303271   0.8497452
   0.7560439   0.6653811   0.685731
   0.0002211   0.6283918   0.8782165
```

Działania na macierzach podlegają określonym regułom związanym z zgodnością wymiarów macierzy. Zazwyczaj wykonuje się jedno z poniższych działań:

- dodawanie stałej do macierzy - `n+A=A+n` - do każdego elementu macierzy `A` dodawana jest wartość liczby `n`. W analogiczny sposób postępuje się przy odejmowaniu.

```
--> A=[1,2,3;4,3,2]
 A  =
   1.   2.   3.
   4.   3.   2.
--> A+10
 ans  =
   11.   12.   13.
```

```
   14.   13.   12.
--> 10+A
 ans  =
   11.   12.   13.
   14.   13.   12.
```

- dodawanie dwóch macierzy - A+B=B+A - obydwie macierze muszą mieć ten sam wymiar

```
--> A=[1,2,3;4,3,2]
 A  =
   1.   2.   3.
   4.   3.   2.
--> A+eye(2,3)
 ans  =
   2.   2.   3.
   4.   4.   2.
```

- mnożenie macierzy przez liczbę `n*A=A*n` - każdy element macierzy `A` przemnażany jest przez liczę `n`

```
--> 3*A
 ans  =
   3.    6.   9.
   12.   9.   6.
```

- mnożenie macierzy A*B - warunkiem koniecznym do wykonania działania jest zgodna liczba wierszy w macierzy A z liczbą kolumn B. Mnożenie macierzy nie jest działaniem przemiennym

```
--> A=[2,3,4;3,2,1];
--> B=[2,7;1,3;5,5];
--> A*B
 ans  =
   27.   43.
   13.   32.
--> B*A
 ans  =
   25.   20.   15.
   11.   9.    7.
   25.   25.   25.
```

Poza działaniami podstawowymi na macierzach w Scilab'e dostępne są także:

- transponowanie macierzy
- macierz odwrotna
- wyznacznik macierzy

```
--> A
 A  =
   3.    1.3   4.6
   2.5   3.2   0.2
   2.2   2.    2.4
--> A'
 ans  =
   3.    2.5   2.2
   1.3   3.2   2.
   4.6   0.2   2.4
--> inv(A)
 ans  =
   1.3   1.1  -2.6
  -1.   -0.5   2.
  -0.3  -0.6   1.1
--> det(A)
 ans  =
   5.4
```

Zadania

W oparciu o materiał wykładowy omówiony w powyższym rozdziale należy wykonać ćwiczenia zestawione poniżej:

1. Korzystając z pomocy systemu Scilab przeanalizować dostępne informacje o omówionych w ćwiczeniu poleceniach.
2. Utworzyć trzy zmienne:
   - macierz o wymiarze 3x4 zawierającą losowo wygenerowane liczby z zakresu <-10,10>,
   - macierz o wymiarze 4x2 zawierającą losowo wygenerowane liczby z zakresu <-2,10>,
   - macierz o wymiarze 5x4 zawierającą w kolumnach liczby w ciągu logarytmicznym w zakresie <1,100>.
3. Utworzyć dwie zmienne zespolone, w wartościach wygenerowanych losowo:

- wektor o długości 5,
- macierz o wymiarze 3x5.

4. Zmienić format wyświetlania liczb na potęgowy, format 11 znaków, a następnie na algebraiczny zdefiniowany przez 6 znaków. Wyświetlić zmienne w obu formatach.

5. W oparciu o zdefiniowane zmienne przetestować omówione w ćwiczeniu działania i funkcje

   - dodawanie i kasowanie (elementów macierzy, wierszy, kolumn)
   - tworzeni macierzy na podstawie wybranych elementów innej macierzy
   - dodawanie i mnożenie macierzy, transpozycję, inwersję i wyznacznik

Bibliografia

[1] A. Brozi. *Scilab w przykładach.* Poznań: Nakom, 2010.

[2] C. . Lachowicz. *Matlab, Scilab, Maxima. Opis i przykłady zastosowań.* Wydawnictwo Politechniki Opolskiej, 2005.

[3] *Strona Scilab Enterprises S.A.S.* 2023. URL: `http://www.scilab.org`.