Programowanie w środowisku obliczeniowym Scilab

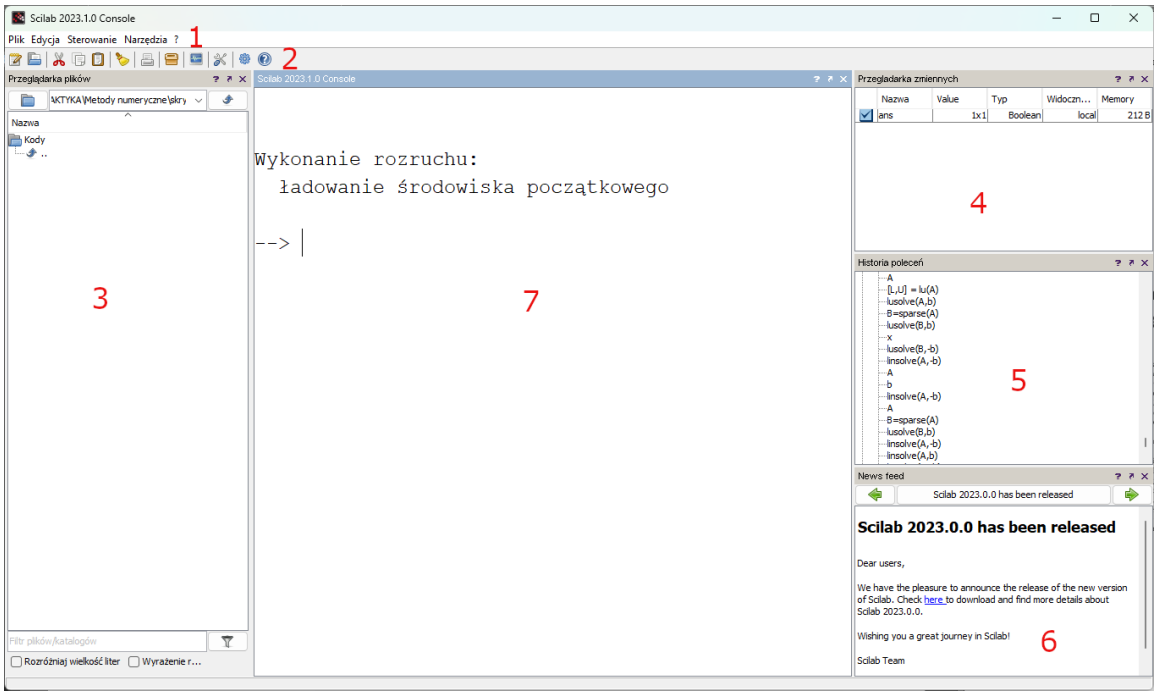
Wstęp

Metody numeryczne są dziedziną matematyki zajmującą się rozwiązywaniem problemów matematycznych za pomocą obliczeń. Wyniki określonych procedur obliczeniowych otrzymuje się zazwyczaj w postaci przybliżonej, z dokładnością przyjętą dla danego zadania. Metody numeryczne znalazły szerokie zastosowanie w analizie oraz rozwiązywaniu problemów inżynierskich i naukowych, szczególnie gdy badany problem nie ma rozwiązania analitycznego albo jego wyznaczenie jest czasochłonne lub skomplikowane. Symulację badanego obiektu lub zjawiska można zaprogramować praktycznie w każdym języku, jednak zwykle wygodniej i efektywniej rozwiązuje się takie problemy w dedykowanych środowiskach obliczeniowych, takich jak MATLAB, Mathcad, Maple, Maxima, Mathematica, GNU Octave czy Scilab [1, 2].

Laboratorium metod numerycznych dla studentów kierunku elektrotechnika na Wydziale Elektrotechniki i Informatyki jest prowadzone z wykorzystaniem środowiska Scilab [3]. Jest to oprogramowanie o otwartym kodzie źródłowym, w znacznym stopniu wzorowane na MATLABie; umożliwia również konwersję wielu skryptów napisanych w języku MATLAB. Projekt jest rozwijany przez programistów i społeczność związaną ze Scilabem. Program jest dostępny dla najczęściej używanych systemów operacyjnych.

Interfejs użytkownika pokazany na ryc. 1 składa się z kilku elementów. Pozycje *Plik* i *Edycja* zawierają standardowy zestaw poleceń. W pracy z programem szczególnie przydatne są pozycje *Sterowanie*, zawierająca polecenia zarządzające wykonywaniem skryptów (przerwanie, zatrzymanie i wznowienie), oraz *Narzędzia*, ze skrótami do podprogramów środowiska:

- **SciNotes** – edytor tekstowy przeznaczony do tworzenia skryptów (programów obliczeniowych) dla środowiska ***Scilab***,
- **Xcos** – moduł programowania graficznego, odpowiednik Simulinka ze środowiska MATLAB,
- **ATOMS** – interfejs zarządzania modułami rozszerzeń środowisk ***Scilab*** i ***Xcos***,
- moduł konwersji projektów MATLAB do środowiska ***Scilab***.



Ryc. 1. Interfejs programu Scilab (**1** – menu główne, **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 i umożliwia bezpośrednią pracę z silnikiem obliczeniowym środowiska. Można w niej wprowadzać pojedyncze polecenia lub ich sekwencje. Wynik pojawia się po zatwierdzeniu wiersza klawiszem Enter. Konsola jest bezpośrednio powiązana z oknami pomocniczymi: przeglądarką zmiennych (4) i historią poleceń (5). Obsługuje standardowe skróty klawiaturowe (*wytnij*, *kopiuj* i *wklej*) oraz szybki dostęp do historii za pomocą klawiszy strzałek w górę i w 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, dzielenie, mnożenie, *sin*, *cos*, *exp*, *sqrt* itp.,
3. działania i funkcje rachunku macierzowego: *inv*, *det*, *diag*, *max* itp.,
4. funkcje numeryczne: *ode*, *fsolve*, *linsolve* itp.,
5. funkcje graficzne i tekstowe: *plot2d*, *plot3d*, *disp*, itp.

W niniejszym rozdziale omówiono wybrane elementy pierwszych trzech grup poleceń. Każdy wiersz wprowadzony w konsoli zostanie wykonany, a jego wynik wyświetlony poniżej. Zakończenie wiersza średnikiem (*;*) blokuje wyświetlanie wyniku w konsoli.

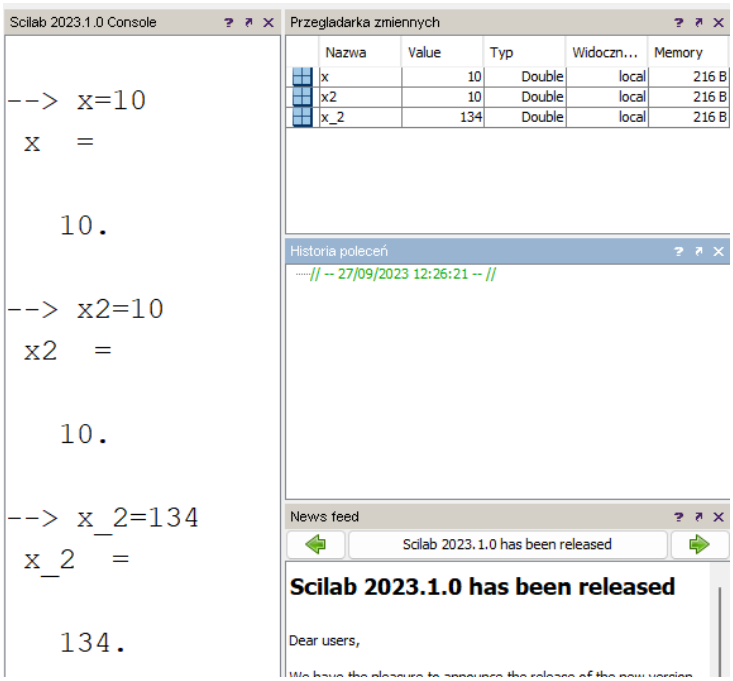
1 Polecenia systemowe

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

Podstawowym narzędziem wsparcia podczas pracy w środowisku obliczeniowym ***Scilab*** jest system pomocy oraz dokumentacja. Dostęp do nich jest możliwy przez interfejs programu lub przez wywołanie odpowiedniego polecenia w konsoli:

- `doc` – otwarcie przeglądarki dokumentacji,
- `doc plot3d` – otwarcie strony dokumentacji polecenia `plot3d`,
- `doc("differential equations")` – wyszukanie w dokumentacji stron związanych z podanym tematem,
- `help("plot3d")` – wyświetlenie skróconego opisu polecenia w konsoli w najnowszych wersjach Scilaba.

Korzystanie z dokumentacji pozwala szybko odnaleźć funkcje potrzebne do rozwiązania analizowanego problemu obliczeniowego lub programistycznego. Zakres dostępnych tłumaczeń zależy od wersji Scilaba; dokumentacja angielska jest wersją referencyjną.



Ryc. 2. Zmienne

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);`.

Podstawowym narzędziem obliczeń wykonywanych w konsoli są zmienne. W środowisku ***Scilab*** zmienną definiuje się przez przypisanie jej pierwszej wartości, a typ zmiennej jest określany na podstawie typu tej wartości. Do podstawowych typów należą: **logiczny** (*Boolean*), **tekstowy** (*String*) i **liczbowy** (*Double*); typ *Double* jest używany także dla liczb zespolonych. Nazwa zmiennej może zawierać litery, cyfry i znak podkreślenia, ale musi rozpoczynać się od litery, np. `x=10`, `x2=10` lub `x_2=10`, jak pokazano na ryc. 2. Nazwa może być taka sama jak identyfikator już dostępny w środowisku, jednak wtedy go przesłania. Na przykład przypisanie `sin=10` uniemożliwi wywołanie funkcji sinus `sin(20)` do czasu usunięcia tej zmiennej. ***Scilab*** rozróżnia wielkość liter, dlatego `Alfa=10` i `alfa=5` definiują dwie różne zmienne. Kontrola istniejących zmiennych jest możliwa w przeglądarce zmiennych lub za pomocą poleceń konsoli.

Funkcja `isdef("zmienna")` zwraca wartość logiczną `%t` lub `%f` w zależności od tego, czy w pamięci istnieje zmienna o nazwie podanej jako argument, 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 o składni i działaniu analogicznym do funkcji `printf()` z języka C. Wyświetla sformatowany ciąg tekstowy w konsoli oraz umożliwia wstawianie do niego wartości zmiennych, wyników funkcji lub wyrażeń, 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 część ułamkową,
- potęgowa – postać zmiennopozycyjna z podstawą potęgową 10.

Sposób wyświetlania liczby jest dobierany automatycznie i zależy od jej wartości. Prezentację można ustawić ręcznie za pomocą funkcji `format(p1,p2)`. Pierwszy argument określa postać zapisu (`"v"` – algebraiczna, `"e"` – wykładnicza), a drugi – liczbę znaków wyświetlanych w oknie konsoli. Należy zwrócić uwagę, że zbyt mała szerokość pola może wymusić zapis wykładniczy. ***Scilab*** przełącza wtedy wynik na odpowiedni tryb, aby zachować możliwie pełną informację o wartości.

```
--> 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 grupie stanowią polecenia operacji plikowych. Pozwalają one na realizację operacji na folderach i plikach. Są przydatne między innymi podczas importowania danych lub funkcji z plików zewnętrznych. Do tej grupy można zaliczyć:

- **chdir** – zmiana aktualnego katalogu,
- **isdir** – sprawdzenie, czy podany katalog istnieje,
- **pwd** – zwrócenie aktualnego katalogu,
- **home** – zwrócenie domyślnego katalogu użytkownika,
- **filebrowser** – otwarcie przeglądarki plików.

2 Podstawowe działania matematyczne

W środowisku obliczeniowym ***Scilab*** obliczenia wykonuje się na liczbach rzeczywistych i zespolonych. Rachunek symboliczny jest możliwy po zainstalowaniu odpowiedniego modułu rozszerzającego. Podobnie jak w większości środowisk obliczeniowych, miarę kąta podaje się domyślnie w radianach, choć dostępne są również funkcje operujące na stopniach. Część ułamkową oddziela się kropką, natomiast przecinek służy do rozdzielania argumentów funkcji. W środowisku zdefiniowano szereg stałych:

- liczba π - %pi
- stała e - %e
- nieskończoność ∞ - %inf

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 w radianach (`sin`) lub stopniach (`sind`).

2	`log(x)`	`-> log(10)` `ans = 2.3025851`	Logarytm naturalny liczby.
3	`log10(x)`	`-> log10(100)` `ans = 2.000000`	Wyznaczenie logarytmu dziesiętnego z liczby.
4	`sqrt(x)`	`-> sqrt(1256)` `ans = 35.440090`	Pierwiastek kwadratowy liczby.
5	`exp(x)`	`-> exp(2)` `ans = 7.3890561`	Obliczenie potęgi wykładniczej liczby.

- operator i części urojonej liczby zespolonej - %i

Podstawowe operacje matematyczne realizuje się za pomocą znaków `+`, `-`, `*`, `/` i `^` oraz nawiasów ustalających kolejność działań, jak pokazano poniżej. Operator `\` nie jest zwykłym operatorem dzielenia: dla skalarów `a\b` odpowiada ilorazowi `b/a`, natomiast dla macierzy zapis `A\B` wyznacza rozwiązanie układu $AX = B$ bez jawnego obliczania macierzy odwrotnej.

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

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

Pierwiastki dowolnego stopnia można obliczać za pomocą operatora `^`, np. `4^0.5`. Funkcja `sqrt` jest wygodnym odpowiednikiem dla pierwiastka kwadratowego. 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 przeznaczone między innymi do rachunku macierzowego. Należy zwrócić uwagę, że wiele problemów rozwiązywanych metodami numerycznymi zapisuje się w postaci macierzowej. ***Scilab*** traktuje skalar jak macierz jednoelementową, a wektor jak macierz jednokolumnową lub jednowierszową. Macierz deklaruje się przez podanie listy elementów w nawiasach kwadratowych; elementy wiersza rozdziela się przecinkami lub spacjami, a wiersze – średnikami, 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 z wykorzystaniem ciągu liniowego lub logarytmicznego. Ciąg liniowy definiuje się, podając wartość początkową, przyrost i wartość końcową. Parametry definiujące ciąg rozdziela się dwukropkiem, a wartość przyrostu można pominąć; w takim przypadku domyślnie przyjmowana jest wartość 1. Poniżej pokazano przykład deklaracji macierzy za pomocą ciągu 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 zapewnienia regularnego kształtu macierzy, są ustawiane 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 ze zgodnością ich wymiarów. 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` — liczba kolumn macierzy `A` musi być równa liczbie wierszy macierzy `B`; mnożenie macierzy nie jest przemienne,

```
--> 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

Celem ćwiczenia jest poznanie podstawowych poleceń środowiska Scilab na przykładzie obliczeń typowych dla elektrotechniki. Obliczenia należy wykonać w sposób odtwarzalny. Każdy student realizuje jeden zestaw z poniższej tabeli; k_R, k_L i k_C mnożą odpowiednio wszystkie wartości R, L i C danych bazowych.

Zestaw	f [Hz]	k_R	k_L	k_C	E_1 [V]
1	50	1,00	1,00	1,00	230
2	60	0,90	1,10	1,00	120
3	40	1,10	0,90	1,20	240
4	75	1,20	1,00	0,80	110
5	100	0,80	1,20	1,10	100
6	45	1,30	0,80	0,90	220
7	55	0,95	1,15	1,25	230
8	80	1,15	0,95	0,85	200
9	25	1,40	1,10	1,00	24
10	120	0,75	0,90	1,30	48
11	50	1,05	1,25	0,95	325
12	90	0,85	1,05	1,15	150
13	30	1,25	0,85	1,05	230
14	150	0,70	1,30	0,75	60
15	65	1,10	1,10	0,90	180

Dane bazowe elementów wynoszą: $R = 120\,\Omega$, $L = 80$ mH, $C = 47\,\mu$F, $RL = (35\,\Omega, 120$ mH) i $RC = (68\,\Omega, 100\,\mu$F). Obwód trzyoczkowy tworzą gałęzie

Gałąź	oczko lewe	oczko prawe	R [Ω]	L [mH]	C [μF]
B1	1	0	30	20	0
B2	1	2	20	10	0
B3	2	0	25	0	100
B4	2	3	15	5	0
B5	3	0	40	0	47
B6	1	0	10	0	0

Źródło E_1 znajduje się w gałęzi B1 i ma zwrot dodatni w oczku 1; pozostałe źródła są równe zeru.

1. Dla parametrów swojego zestawu obliczyć zespolone impedancje elementów R, L, C, RL i RC. Wyniki przedstawić w postaci algebraicznej i biegunowej.
2. Na podstawie tabeli gałęzi utworzyć macierz impedancji oczkowych Z i wektor napięć źródłowych E. Wyznaczyć wektor prądów poleceniem

   ```
   I = Z\E;
   ```

   i sprawdzić normę residuum $r = ZI - E$.
3. Na utworzonych danych przećwiczyć wybieranie elementów, wierszy i kolumn, łączenie macierzy, transpozycję sprzężoną oraz zmianę formatu wyświetlania liczb.
4. Zestawić parametry elementów, impedancje, prądy oraz normę residuum w czytelnej tabeli i zapisać wyniki do pliku.

Zadanie rozszerzające. Porównać rozwiązania otrzymane za pomocą operatora `Z\E` oraz wyrażenia `inv(Z)*E`. Wyjaśnić, dlaczego do rozwiązywania układów równań nie zaleca się jawnego obliczania macierzy odwrotnej.

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.* 2024. URL: `http://www.scilab.org`.