ĆWICZENIE 1

Cel: Zapoznanie studenta z podstawowymi informacjami o środowisku obliczeniowym SCILAB. Przedstawienie historii powstania i rozwoju programu oraz zapoznanie się z podstawowym interfejsem. Zostaną wymienione rodzaje zadań w realizacji których program SCILAB znajduje zastosowanie, oraz przykłady takiego zastosowania.

1. SCILAB Wprowadzenie

Scilab jest wolnym (open source) środowiskiem programowym, które w sposób interaktywny umożliwia prowadzenie obliczeń numerycznych o charakterze inżynierskim a także naukowym. Zarówno działanie programu opierające się na pracy z użyciem macierzy i wektorów, jak też budowa interfejsu są podobnie jak znanego systemu MATLAB(MathWorks). Funkcjonalnością SCILAB dorównuje innym znanym narzędziom obliczeniowym (Matlab, GNU Octave, Python (z bibliotekami NumPy/SciPy/Matplotlib) i in.). Wokół SCILAB-a powstało środowisko użytkowników i programistów, rozwijające biblioteki (m.in. algebrę liniową, optymalizację, przetwarzanie sygnałów) oraz narzędzia:

- Xcos – graficzne modelowanie systemów dynamicznych (następca Scicos),
- ATOMS – menedżer rozszerzeń (instalacja pakietów tworzonych i rozwijanych przez społeczność programistów),
- narzędzia do integracji z językami kompilowanymi (C/Fortran) i Javą.

SCILAB jest wyposażony w interpreter ukierunkowany na operacje wektorowo-macierzowe, z bogatą biblioteką funkcji matematycznych, narzędziami graficznymi oraz modułem do modelowania blokowego Xcos (odpowiednik Simulinka-MATLAB). Do podstawowych funkcjonalności programu należą:

- przeprowadzanie macierzowych obliczeń numerycznych – bezpośrednio lub poprzez programowanie,
- tworzenie wykresów,
- modelowanie systemów dynamicznych (blokowo),
- analiza danych, filtracja i przetwarzanie sygnałów oraz obrazów,
- optymalizacja i identyfikacja parametrów,
- projektowanie i analiza układów sterowania,
- rozwiązywanie równań różniczkowych i równań nieliniowych,
- prototypowanie algorytmów,
- tworzenie prostych interfejsów użytkownika.

SCILAB został stworzony na początku lat 90. przez francuski instytut badawczy INRIA (Institut National de Recherche en Informatique et en Automatique), oraz École des Ponts ParisTech (ENPC) jako otwarte środowisko obliczeń numerycznych dla inżynierii i nauk ścisłych. Rozwój systemu był wspierany przez INRIA Scilab Consortium, które koordynowało rozwój, popularyzację i współpracę z przemysłem. Aktualnie (od 2022) zespół SCILAB jest częścią Dassault Systèmes. Kolejne główne wersje utrzymują otwartą licencję GPL. Kolejne główne wersje przynosiły istotne zmiany: wersje 5.x wprowadziła m.in. nowy system grafiki i menedżer rozszerzeń ATOMS, a wersje 6.x – unowocześniony silnik wykonawczy, usprawnioną gospodarkę pamięcią i rozwój Xcos (narzędzia do symulacji blokowych).

Z oprogramowania SCILAB korzystają liczne uczelnie i instytuty badawcze na świecie (m.in. INRIA, École des Ponts ParisTech), a także organizacje z sektora publicznego i przemysłu, zwłaszcza w krajach, gdzie promuje się otwarty dostęp do informatycznych narzędzi inżynierskich. Na liście referencyjnej SCILAB można znaleźć podmioty z branż: energetycznej, lotniczej i kosmicznej, takie jak m. in.: EDF (Électricité de France), CNES (francuska agencja kosmiczna), Dassault Aviation, Airbus, Total, AcelorMittal, Peugeot, Hutchinson wykorzystujące Scilab/Xcos w analizie, symulacjach i pracach badawczo-rozwojowych. Powszechne jest także stosowanie Scilab w edukacji inżynierskiej jako alternatywy dla komercyjnych pakietów.

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 oraz animacji. 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 2025.1.0* , 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 interaktywną *Konsolę* umożliwiającą natychmiastowe uruchamianie poleceń oraz własny *Edytor* z wbudowanym debugerem umożliwiający porządkowanie kodu oraz tworzenie i zapisywanie własnych funkcji (.sci), skryptów (.sce) i bibliotek. Wśród niezbędnych narzędzi programistycznych, oprócz *Konsoli* i *Edytora* należy wymienić: Interpreter poleceń, Biblioteki funkcji, Interfejsy dla innych języków: m.in. C/Fortran , Tcl/Tk, Java, Modelica, LabVIEW, oraz narzędzia integracji i wymiany danych z Python i MATLAB.

Możliwości jakie oferuje SCILAB sprawiają, że program dobrze sprawdza się w zadaniach inżynierskich, które wymagają obliczeń, wizualizacji wyników oraz automatyzacji działań.
Wśród możliwych zastosowań inżynierskich programu można wymienić:

1. Analiza numeryczna w typowych zadaniach projektowych
   - Układy równań liniowych: rozwiązywanie modeli równań wynikających z bilansów masy/energii, przepływów, obwodów (metody macierzowe).
   - Algebra liniowa: wyznaczniki, wartości własne, rozkłady, analizy stabilności układów.
   - Aproksymacja i interpolacja: dopasowanie krzywych, splajny
   - Całkowanie i różniczkowanie numeryczne
   - Równania różniczkowe zwyczajne (ODE) i różniczkowo-algebraiczne (DAE): symulacje procesów dynamicznych (mechanika, termodynamika, obwody).
2. Automatyka i sterowanie
   - Analiza układów: funkcje przejścia, reprezentacja stanu, wykresy Bodego i Nyquista, analiza stabilności.
   - Synteza regulatorów: strojenie PID,
   - Symulacja: odpowiedzi na wymuszenia, badanie wpływu zakłóceń i nieliniowości, badania odporności.
3. Przetwarzanie sygnałów i obrazów
   - Sygnały: filtracja, transformacje, spektrogramy resampling; analiza drgań, hałasu, komunikacji.
   - Obrazy: wczytywanie, filtracja (wygładzanie, wyostrzanie), wykrywanie krawędzi, morfologia.
   - Zastosowania: diagnostyka maszyn, analiza wibracji, wizja maszynowa.
4. Optymalizacja i planowanie
   - Minimalizacja/maksymalizacja funkcji z ograniczeniami i bez: dobór parametrów projektowych (masa vs. wytrzymałość), strojenie modeli. Optymalizacja nieliniowa, programowanie liniowe (alokacja zasobów).
5. Statystyka i analiza danych
   - Statystyki opisowe, testy podstawowe, korelacje, analiza wariancji.
   - Dopasowanie modeli: regresja liniowa i nieliniowa, walidacja krzyżowa.
   - Wizualizacja: histogramy, wykresy pudełkowe, mapy konturowe, 2D/3D.
6. Mechanika i wytrzymałość materiałów
   - Metody macierzowe w analizie prętów i belek (macierze sztywności, obciążenia).
   - Obliczenia drgań własnych prostych układów.
7. Inżynieria procesowa, energetyka, środowisko
   - Bilansy masy i energii, wymiana ciepła,
   - Szacowanie sprawności, charakterystyki pomp i wentylatorów, profile mocy w systemach OZE
   - Analiza danych pomiarowych (loggery, stacje meteo, liczniki energii)
8. Integracja z rzeczywistymi danymi i urządzeniami
   - Import/eksport: pliki CSV i tekstowe, arkusze kalkulacyjne, dane czasowo-rzeczywiste

- Współpraca z innymi narzędziami: wymiana danych przez standardowe formaty lub biblioteki; możliwość łączenia z kodem C/C++/Fortran, a także Java.

9. Modelowanie i symulacja układów dynamicznych w Xcos
   - Xcos to środowisko blokowe (diagramy), przydatne do budowania modeli bez pisania równań wprost. Obsługuje układy ciągłe, dyskretne i hybrydowe.
   - Przykłady: wahadło tłumione, układy RLC, zbiornik z regulacją poziomu, napęd silnik–sprzęgło–obciążenie i inn.
   - Zastosowania: szybkie prototypowanie, testowanie regulatorów (np. PID), analiza odpowiedzi skokowych i częstotliwościowych.

Scilab łączy funkcje kalkulatora macierzowego, narzędzia do symulacji (Xcos) i elastyczne możliwości programistyczne. Dzięki takim cechom umożliwia:

- efektywne wsparcie szerokiego zakresu pracy inżyniera: od modelu, przez symulację i optymalizację, po raport,
- automatyzowanie(programowanie) powtarzalnych zadań oraz budowaćwłasnych narzędzi (funkcje, skrypty, GUI),
- naukę programowania, podstaw inżynierii systemów i analizy danych.

2. Podstawowy interfejs Scilab – elementy, przeznaczenie i praktyczny sposób użycia

W pierwszym kontakcie z programem najważniejsze jest oswojenie się z oknem głównym i jego panelami. Interfejs programu Scilab jest zorganizowany z kilku współpracujących elementów:

- **Konsola** (miejsce wykonywania poleceń),
- **Przeglądarka zmiennych**,
- **Historia poleceń**,
- **Przeglądarka plików**,
- Edytor (SciNotes – do pisania skryptów i funkcji),
- Okno wykresów,
- Przeglądarka pomocy,
- Okno środowiska Xcos do modelowania blokowego.

Układ paneli można dopasować do własnych potrzeb – panele da się dokować, odrywać i zmieniać ich rozmiary. Poniżej opis najważniejszych części interfejsu wraz z przykładowym użyciem.

Podstawowy interfejs użytkownika zawiera cztery okna (wypisane powyżej czcionką pogrubioną) (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. Polecenia są wpisywane za znakiem zachęty **-->** , a po zatwierdzeniu Enterem Scilab wykonuje polecenie. 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 z *Historii poleceń* do *Konsoli* i poprawiając błąd przed zaakceptowaniem polecenia. Szybki dostęp do poleceń z *Historii poleceń* można też uzyskać wciskając klawisz kierunkowy (strzałka w górę).

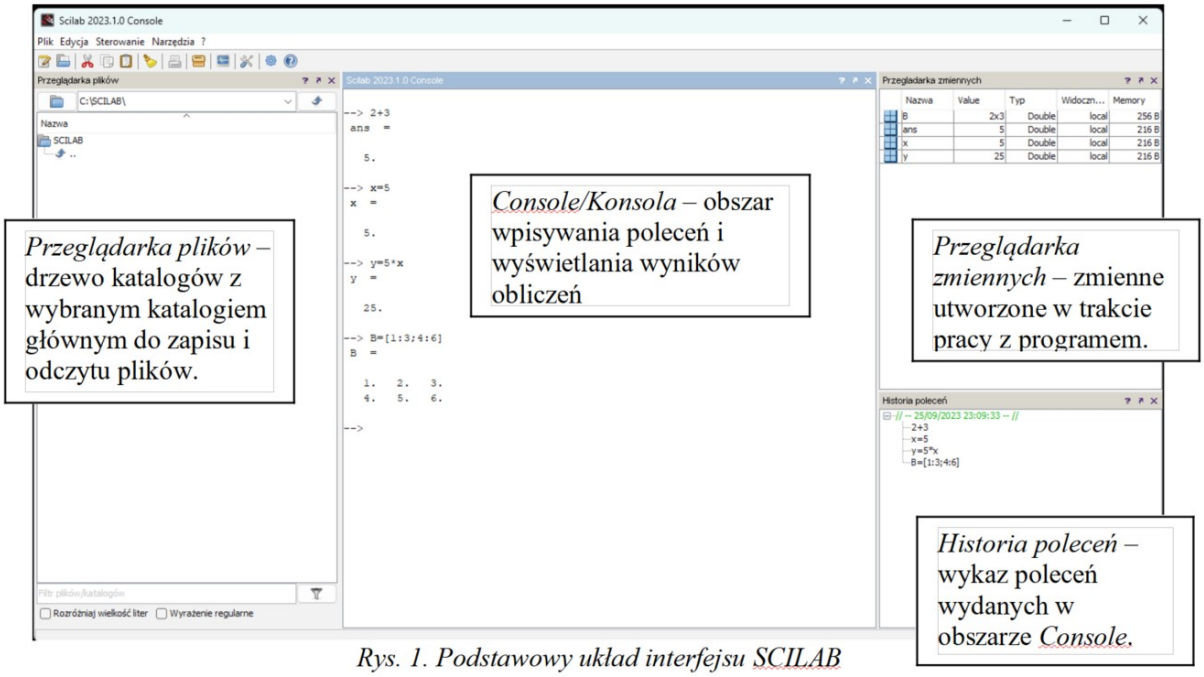
Do czego służy Konsola:

- szybkie obliczenia i testy,
- uruchamianie skryptów i funkcji,
- komunikaty o błędach i ostrzeżenia.

Jak korzystać z Konsoli:

- jako kalkulator macierzowy: wpisz wyrażenie i naciśnij Enter,

- używaj strzałek w górę/dół, aby przewijać ostatnie polecenia,
- skorzystaj z auto-uzupełniania klawiszem Tab (np. wpisz pl i naciśnij Tab →wybierz plot),
- podstawowe polecenia porządkowe:
  - clc // czyści tekst w konsoli,
  - who // lista zmiennych,
  - whos // lista zmiennych z typami,
  - clear x y // usuwa zmienne x i y, clear // usuwa wszystkie zmienne



Rys. 1. Podstawowy układ interfejsu SCILAB

Przeglądarka plików (*File Browser*): przeglądarka zorganizowana w postaci drzewa katalogów z ustawioną ścieżką do katalogu bieżącego (*Current Directory*), w którym są zapisywane pliki utworzone podczas pracy z programem, i w którym w pierwszej kolejności poszukiwane są pliki wczytywane.
Do czego służy przeglądarka plików:

- szybkie nawigowanie po plikach projektowych (dane, skrypty, funkcje),
- ustawianie bieżącego katalogu pracy.

Jak korzystać z przeglądarki plików:

- wybór katalogu klikając w drzewie folderów lub podając ścieżkę,
- dwuklik pliku .sce lub .sci otwiera go w Edytorze (SciNotes),
- polecenia konsolowe związane z przeglądarką plików:
  - pwd() // pokaż katalog,
  - chdir("C:\ ścieżka do nowego katalogu") // zmień katalog,
  - dir() // lista plików.

Przeglądarka zmiennych (*Variable Browser*): okno z wyszczególnionymi zmiennymi z podaniem nazwy, typu, wymiaru i przykładowych wartości zmiennych. Jest to aktualna „zawartość pamięci” Scilab.
Do czego służy Przeglądarka zmiennych:

- podgląd i szybka kontrola zawartości przestrzeni roboczej programu,
- otwieranie i przeglądanie zmiennych w widoku tabelarycznym.

Jak korzystać z Przeglądarki zmiennych:

- dwuklik na zmiennej otwiera ją w edytorze danych,
- klik prawym przyciskiem umożliwia m.in. eksport do pliku,
- szybkie wizualne sprawdzanie ‘istnienia’ zmiennej.

Historia poleceń (*Command History*): rejestr poleceń wpisywanych w *Konsoli* podczas bieżącej sesji programu, a także podczas sesji historycznych – z podziałem wg dat sesji.
Do czego służy Historia poleceń:

- ponowne uruchamianie często wykorzystywanych komend,
- kopiowanie poleceń do konsoli lub edytora.

Jak korzystać z Historii poleceń:

- dwuklik na wpisie w Historii poleceń, powoduje wysłanie polecenia konsoli i wykonanie go,
- przeciągnięcie wpisu z Historii poleceń do konsoli lub do edytora pozwala zmodyfikować/poprawić wpis przed jego wykonaniem
- użycie klawiszy strzałek ↑ ↓ w konsoli, pozwala przewijać historię.

Dodatkowo uaktywnione mogą być dodatkowe okna:

Edytor (SciNotes): Osobne okno zawierające narzędzie programistyczne do pisania kodu programów - (funkcji, skryptów) z kolorowaniem składni, numerami linii i przyciskami uruchamiania. Edytor wywoływany poleceniem ***edit*** wpisanym w *konsoli,* lub wybierając ikonę na pasku narzędzi przy aktywnym oknie Console,
Do czego służy SciNotes:

- tworzenie i uruchamianie skryptów (.sce),
- tworzenie funkcji (.sci),
- wygodna edycja dłuższych zadań, możliwość jednoczesnego wywoływania wielu poleceń,
- dokumentowanie kodu komentarzami - //.

Jak korzystać z SciNotes:

- tworzenie nowego pliku i zapisywanie w bieżącym katalogu,
- przenoszenie do konsoli i uruchamianie całych skryptów lub zaznaczonych fragmentów kodu odpowiednim przyciskiem (Execute/Wykonaj),
- komunikaty błędów kompilacji/wykonania wyświetlane w konsoli pokażą numer linii z błędem co pozwala na powrót do edytora, poprawienie kodu we wskazanym miejscu i ponowne uruchomienie.

Wskazówka: skrypt można uruchomić także poleceniem exec("plik.sce") wpisanym w konsoli.

Okno graficzne (*Figure*): służące do wyświetlania grafiki np. wykresów. Każde polecenie rysowania wykresu tworzy okno typu Figure z osią/osiami oraz paskiem narzędzi umożliwiającym podstawowe działania na wykresie: przybliżanie, przesuwanie, wskazówki danych.
Do czego służy okno graficzne:

- wizualizacja wyników 2D/3D, interaktywna analiza punktów,
- eksport wykresów do plików graficznych.

Jak korzystać z okna graficznego:

- standardowe polecenia: plot, plot2d, subplot, plot3d, mesh, contour, powodują utworzenie określonego typu wykresu
- uchwyty graficzne do modyfikacji wyglądu: gce(); gca(),
- zapis wykresu w pliku graficznym: w menu wykresu: Menu: *Plik - Exportuj do...*
- użycie 'scroll' umożliwia przybliżanie wykresu,
- wciśnięty lewy klawisz myszy umożliwia przesuwanie przybliżonego wykresu,

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

Przeglądarka pomocy (*Help Browser*) Wbudowana pomoc/dokumentacja do programu uaktywniana poleceniem *help*, zawierająca opisy funkcji, przykłady i odsyłacze. Dostęp do przeglądarki pomocy można uzyskać z każdego okna programu wybierając jeden z przycisków na pasku tytułowym . Jednak podstawowym

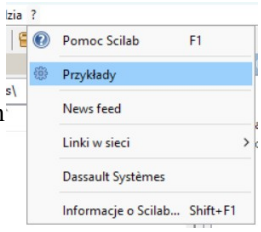
sposobem wywoływania jest polecenie **help** *temat* wpisywane w *Konsoli*, co spowoduje otwarcie okna *Przeglądarki pomocy* zawierającego szczegółowe objaśnienia dotyczące słowa *temat* (warto korzystać z polecenia **help** w celu lepszego poznania działania funkcji Scilaba);
Do czego służy Przeglądarka pomocy:

- szybkie znalezienie składni polecenia/funkcji i przykładów użycia,
- przegląd tematyczny możliwości programu (grafika, optymalizacja, sygnały, Xcos, ...).

Jak korzystać z Przeglądarki pomocy:

- przeglądarkę pomocy można otworzyć z menu Pomoc / klikając ikonę ze znakiem zapytania, wpisując help w konsoli lub wciskając klawisz funkcyjny F1,
- wyszukiwanie jest możliwe po słowach kluczowych wpisywanych w wyszukiwarce Okna pomocy,

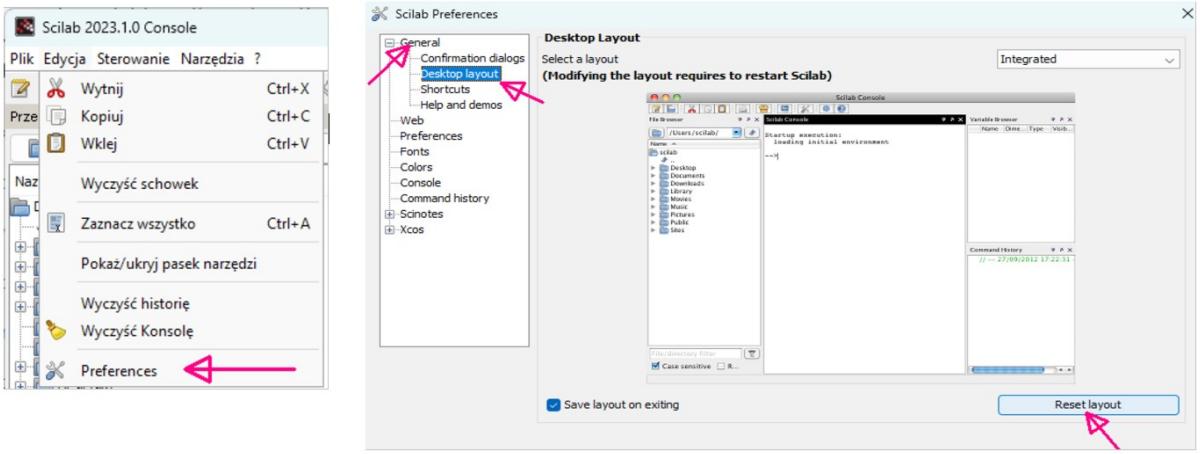
Demonstracje:
Dostęp do przykładów demonstracyjnych programu SCILAB jest możliwy z menu tekstowego okna *Console* wybierając symbol ‘ *?* ’ a następnie ‘*Przykłady*’.
W dodatkowym oknie można będzie wybrać przykłady realizacji różnorodnych działań z wykorzystaniem funkcjonalności programu.



Komunikaty błędów:
Komunikaty błędów są 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
```

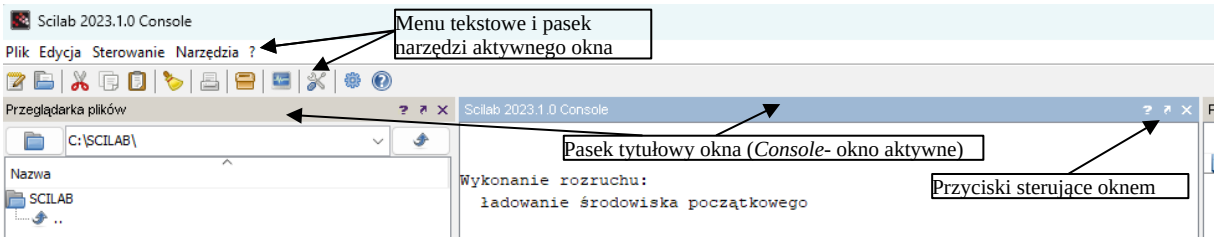
Wygląd interfejsu użytkownika może być dowolnie konfigurowany. Każde z okien można „oddokować” z głównego okna SCILAB, zmienić jego położenie lub zamknąć jeśli nie będzie nam w danej sesji potrzebne. W każdym czasie możemy przywrócić podstawowy układ okien interfejsu wybierając polecenia z menu górnego *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.



Rys. 2. Przywracanie podstawowego interfejsu SCILAB

UWAGA: w danej chwili pracy programu 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 okna, 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).



Rys. 3. Elementy okien interfejsu SCILAB

3. Typy danych w Scilab

W Scilab wszystkie zmienne mają typ (określa „rodzaj” danych) oraz kształt (określa sposób zapisu: skalar, wektor, macierz, hipermacierz). Podstawową strukturą jest **macierz/tablica**, a większość operacji działa tak samo dla skalarów, wektorów i macierzy. Typ zmiennej w Scilab-ie nie jest deklarowany. Określanie typu zmiennej odbywa się dynamicznie – ta sama nazwa zmiennej może w kolejnych liniach przechowywać różne typy. 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).

*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.*

Najczęściej używane typy i konstrukcje zmiennych:

(1) **Liczby rzeczywiste** (double, typ „constant”) to domyślny typ liczbowy w Scilab; obliczenia odbywają się w podwójnej precyzji zmiennoprzecinkowej. Pojedyncza liczba jest traktowana jak macierz o jednym wierszu i jednej kolumnie.

Przykłady:

```
-->a = 3.14 b = 2 // całkowita stała, ale typu double
-->c = %pi // stała pi
-->eps = %eps // najmniejsza rozróżnialna różnica (dokładność obliczeniowa)
-->typeof(a) // "constant" – sprawdzenie typy zmiennej a
```

(2) **Liczby zespolone** Znak %i oznacza jednostkę urojoną. Funkcje real(), imag(), abs(), arg(), conj() pomagają w analizie liczb zespolonych.

Przykłady:

```
-->z = 2 - 3*%i
-->modul = abs(z)   // długość wektora w płaszczyźnie zespolonej
-->faza = atan(imag(z), real(z))
-->typeof(z)   // "constant"
```

(3) **Wektory i macierze (2D)** Struktura tablicy złożonej z wierszy i kolumn. Szczególnym przypadkiem macierzy jest wektor (wierszowy lub kolumnowy).

Tworząc wektor wierszowy używamy przecinków lub spacji, kolumnowy średników; macierze łączą oba.

Przykłady:

-->v = [1 2 3], v1 = [1, 2, 3] // wektor wierszowy
-->w = [1; 2; 3] // wektor kolumnowy
-->A = [1 2; 3 4] // macierz 2x2
-->x = 0:0.1:1 // wektor równych kroków

(4) **Hipermacierze (N-wymiarowe)** Rozszerzenie macierzy 2D na więcej wymiarów; przydatne np. w analizie obrazów kolorowych (wysokość x szerokość x kanały).

Przykłady:
-->H = ones(3,4,2) // hipermacierz 3x4x2 wypełniona jedynkami
-->H(:,:,2) = 2*H(:,:,2) // operacja na „drugiej warstwie"
-->size(H) // rozmiar hipermacierzy [3 4 2]

(5) **Wartości specjalne liczbowe**

%nan (Not a Number): wynik nieokreślonych operacji.
%inf, -%inf: dodatnia/ujemna nieskończoność.
Przykłady:
-->y = 0/0 // %nan
-->z = 1/0 // %inf
-->isnan(y), isinf(z) //sprawdzenie wartości specjalnej

(6) **Typ logiczny (boolean)** Wartości prawda to %t , %T a fałsz to %f , %F

Przykłady:
-->x = [-2 -1 0 1 2] // utworzenie wektora wierszowego x
-->mask = x > 0 // [F F F T T] - maska logiczna wektora x dla warunku logicznego x>0
-->pos = x(mask) // [1 2] -filtrowanie z wektora x elementów wg warunku maski logicznej

(7) **Ciągi znaków (string)** Tekst/Napisy zapisujemy w cudzysłowach. Można je łączyć, dzielić, konwertować liczby na tekst i odwrotnie.

Przykłady:
-->n = 5, s = "Pomiar nr " + string(n) // "Pomiar nr 5"
-->typeof(s) // "string"
-->slowa = ["Ala","ma","kota"] //wektor słów
-->zdanie = strcat(parts, " ") // "Ala ma kota" - łączenie tekstu w jedną linię

(8) **Wielomiany (poly)** Scilab przechowuje wielomiany jako obiekty symboliczne – można na nich wykonywać operacje sumowania, mnożenia, obliczania pierwiastków.

Przykłady:
-->s = poly(0, "s") // zmienna wielomianowa „s"
-->p = s^2 + 2*s + 1 // p = 1 +2s +s^2 - definiowanie wielomianu
-->r = roots(p) // obliczenie pierwiastków : [-1; -1] - pierwiastki mogą być podane w postaci zespolonej
-->val = horner(p, 2) // 9 -obliczenie wartości wielomianu dla s=2
-->typeof(p) // "poly" – sprawdzenie typu zmiennej p

(9) **Wyrażenia wymierne (rational)** Iloraz dwóch wielomianów – przydatny w teorii układów (np. funkcje przejścia).

Przykłady:
-->s = poly(0, "s"), R = (s + 1) / (s^2 + 2*s + 1), - definiowanie ilorazu wielomianów
-->typeof(R) // "rational"– sprawdzenie typu zmiennej R

(10) **Listy heterogeniczne (list)** Lista elementów różnych typów z dostępem indeksowym, liczonym od 1.

Przykłady:
-->L = list(42, "czujnik", %t, [1 2 3]) // utworzenie listy L
-->typeof(L) // sprawdzenie typu zmiennej L
-->L(2) // "czujnik" wybór elementu listy z pozycji 2
-->L($) // ostatni element listy: [1 2 3]
-->L(1) = 100 // podmiana elementu z pozycji 1

(11) **Funkcje (function)** Funkcje można tworzyć, przekazywać jako argumenty i przechowywać w zmiennych.

Przykłady:
-->deff("y=g(x)","y=sin(x)") // definicja funkcji „w jednej linii"
-->typeof(f) // "function" - sprawdzenie typu zmiennej f

(12) **Uchwyt graficzny (handle)** Dostęp do właściwości obiektów składowych na wykresie; pozwala programowo zmieniać wygląd wykresu/grafiki.

Przykłady:
-->t = 0:0.01:1; y = sin(2*%pi*10*t); // wartości zmiennych do utworzenia wykresu,
-->plot(t,y) // rysowanie wykresu,
-->h = gce() // „ostatnio utworzony element" – compound
-->set(h.children, "thickness", 2) // zmiana grubości linii
-->ax = gca(); set(ax, "grid", [1 1]) // dodanie siatki na wykresie

Typy wielomianów, wyrażeń wymiernych, uchwytów graficznych oraz list sprawiają, że Scilab dobrze nadaje się do zadań inżynierskich. Dzięki listom z łatwością organizujemy dane i wyniki w czytelne, opisane zbiory. Opis „naturalnych" cech obiektów (np. funkcje przejścia, modele) może być przechowywany wprost w zmiennych, z których można następnie korzystać przy wykonywaniu operacji algebraicznych oraz wizualizacji.

4. Wybrane elementy środowiska Scilab i podstawowe funkcje wbudowane.

Pisanie poleceń w konsoli Scilab opiera się na kilku prostych, ale bardzo konsekwentnych zasadach składniowych. Elementy takie jak nawiasy, znaki interpunkcyjne i krótkie polecenia konfiguracyjne decydują o tym, czy polecenie zadziała zgodnie z intencją. Poniżej omówiono najważniejsze elementy, ilustrując je krótkimi przykładami i komentarzem objaśniającym ich stosowanie.

--> 'znak zachęty' wiersza poleceń w oknie *Console* – miejsce gdzie należy wpisywać polecenia, które będą realizowane w programie po ich zaakceptowaniu 'enterem'.

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 *'enter'*
1 2 3 1]

- wielokrotna podstawa - Jeżeli funkcja zwraca wiele wielkości, , można je bezpośrednio przypisać do kilku zmiennych zapisanych w wektorze przy użyciu nawiafów kwadratowych;

--> [m, n] = size(A)

Nawiasy ( ) :

Nawiasy okrągłe mają trzy główne zastosowania: grupowanie działań, wywoływanie funkcji oraz indeksowanie (odwoływanie się do elementów tablic i macierzy).

- Określenie kolejności wykonywania działań matematycznych pozwala wymusić kolejność obliczeń, co bywa kluczowe w dłuższych wyrażeniach: **-->**x=(2+6)*7
- Nawiasy po nazwie funkcji zawierają argumenty, oddzielone przecinkami. Nawet jeśli funkcja nie ma argumentów, nawiasy puste mogą być wymagane: **-->**sin(%pi/4)
- Oznaczanie indeksów wektorów i tablic - wskazujemy element (i, j) macierzy A, lub zakres wierszy lub kolumn, itp. **-->**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.
- Pola struktur: s.pole, jeśli pracujemy ze strukturami, dostęp do pola odbywa się przez kropkę:
  --> s = struct('imie','Ada','rok',2023)
  --> s.rok // 2023

Przecinek , :

- W wywołaniach funkcji przecinek rozdziela argumenty: f(a, b, c)
- W wektorach/macierzach przecinek (albo spacja) rozdziela elementy w wierszu: **-->** v = [10, 20, 30]
- Oddziela indeksy tablic: A(1,3)
- Na poziomie konsoli przecinek może także oddzielać kolejne instrukcje na jednej linii: **-->** a = 1, b = 2

Średnik ; :

- Oddziela kolejne wiersze macierzy.
- Umieszczony na końcu wyrażenia powoduje, że wynik wyrażenia nie jest wyświetlany na ekranie a wynik jest zapisywany do *Przeglądarki zmiennych.*
- Umożliwia umieszczenie kilku poleceń w jednym wierszu: **-->** a = 5; b = 6; c = a + b

Znak // :

Komentarz liniowy - tekst po znaku // aż do końca linii jest traktowany jako komentarz.

Dwukropek : :

Dwukropek to wygodny generator zakresów i sposób adresowania fragmentów tablic.

- Służy do konstrukcji wektorów o elementach liniowo narastających lub malejących:
  -->X=1:10
  -->X1=1:2:10
  -->X2=20:-3:0
- W indeksowaniu tablic pozwala na wybór zakresów wierszy/kolumn:
  A(2:4, 1) // wybór wierszy od 2 do 4, oraz kolumny 1

diary('nazwaPliku'):

Polecenie *diary* służy do zapisywania do pliku tekstowego tego, co pojawia się w konsoli (polecenia i odpowiedzi). To przydatne, gdy chcemy udokumentować działania lub dołączyć je do sprawozdania.
Rejestrowanie kończymy poleceniem diary(0). Plik zostanie utworzony w katalogu bieżącym.

```
diary('sesja.txt') // rozpocznij zapis do pliku sesja.txt
diary(0) // zakończ zapis
diary('sesja.txt','append') // dopisuj do istniejącego pliku
```

UWAGA: Istniejący plik o takiej samej nazwie zostanie, bez uprzedzenia nadpisany. W zależności od wersji Scilab składnia zatrzymywania zapisu może się nieznacznie różnić. Dobrze jest sprawdź pomoc: help diary.

format : *postać_liczby* :

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

Format nie zmienia sposobu obliczeń, a jedynie sposób prezentacji wyniku liczbowego w konsoli (liczba cyfr, zapis wykładniczy itp.). Dzięki temu możemy np. szybciej „podglądać" wyniki lub, przeciwnie, zobaczyć więcej cyfr znaczących.

Przykład:

Przedstaw wyrażenie 4/3 w różnej postaci używając funkcji format.

--> format v // tryb domyślny

--> 4/3

--> format e // zapis wykładniczy

--> 4/3

--> format ('e',12) // zapis wykładniczy z 12 pozycjami

--> 4/3

Podstawowe funkcje wbudowane w Scilab to gotowe „klocki", z których można zbudować rozwiązania zadań obliczeniowych, które wywołujemy po prostu nazwą funkcji i nawiasami z argumentami z którymi chcemy daną funkcję wywołać. Do obliczeń matematycznych użyjemy m.in. funkcji trygonometrycznych:sin, cos, tan, cotg, wykładniczych:exp, sqrt, logarytmicznych: log, log10 , do pracy na wektorach i macierzach – size, length, sum, mean, min, max, sort. Do tworzenia danych startowych przydadzą się zeros, ones, eye (macierz jednostkowa), linspace (wektor równomierny) i rand (liczby losowe). Scilab oferuje też proste funkcje wejścia/wyjścia, np. disp do wyświetlania oraz input do pobierania wartości od użytkownika, a także narzędzia do zaokrąglania i konwersji, takie jak round, floor, ceil czy string. Większość z tych funkcji poznamy w kolejnych ćwiczeniach stosując je w konkretnych zadaniach. Wiele funkcji potrafi zwracać kilka wyników naraz, które uzyskujemy w nawiasach kwadratowych, np. [m, n] = size(A). Ilość dostępnych funkcji powoduje, że niemożliwe jest zapamiętanie składni każdej z nich. Jeśli nie pamiętasz dokładnie składni, wpisz help nazwa_funkcji (np. help linspace) – dokumentacja wprost pokazuje wymagane argumenty, przykłady i zwracane wartości. Dzięki tym kilku grupom funkcji można szybko zacząć liczyć, analizować i porządkować dane już od pierwszej sesji z Scilabem.

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 $i=\sqrt{-1}$
%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)	-->b=b^2;	-->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:

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