Pakiety obliczeniowe

Laboratorium 1 – Matlab – wprowadzenie

1. Praca w Command Window

Najprostszym sposobem pracy w środowisku Matlab jest wydawanie poleceń w Oknie poleceń (Command Window) po symbolu zachęty (>>).

1. Pomnóż 3 przez 7 wpisując polecenie **`3*7`** po symbolu zachęty.

Wyniki wykonywanych działań są domyślnie umieszczane w zmiennej **`ans`**.

Wykorzystując operator przypisania (=) można umieścić wynik obliczeń w dynamicznie utworzonej zmiennej.

2. Pomnóż 3 przez 7 umieszczając wynik działania w zmiennej **`p`** (**`p = 3*7`**).

Operator przypisania pozwala przypisać wartość wyrażenia z prawej strony do zmiennej umieszczonej po lewej.

3. Wprowadź polecenie **`p = p + 2`**, sprawdź co się stanie.

W oknie przestrzeni roboczej (Workspace) znajduje się podgląd wszystkich zmiennych.

4. Sprawdź jakie zmienne znajdują się w przestrzeni roboczej.

Jeśli polecenie zakończysz średnikiem (;), to wynik działania nie zostanie wyświetlony.

5. Wprowadź polecenie **`m = 13 - 8;`** i sprawdź czy w przestrzeni roboczej pojawiła się zmienna **`m`** oraz czy jej wartość jest zgodna z oczekiwaniem.

Pracując w Command window, możesz się łatwo cofnąć do wcześniejszych poleceń wykorzystując strzałkę w górę.

6. Cofnij się do polecenia **`p = 3*7`** i zmodyfikuj je następująco **`p = 3*m`**.

Nazwy zmiennych muszą zaczynać się literą oraz mogą zawierać litery, cyfry i znak podkreślenia.

7. Wprowadź polecenia **`a = 3`** oraz **`A = 4`**.
8. Wprowadź polecenie, które utworzy zmienną **`avgAa`** i przypisze jej wartość średniej zmiennych **`a`** oraz **`A`**.

Zawartość przestrzeni roboczej można łatwo zapisać do pliku poleceniem **`save`**.
Polecenie **`save nazwapliku`** *utworzy plik* **`nazwapliku.mat`** *i zapisze w nim wszystkie zmienne z przestrzeni roboczej.*

9. Zapisz zawartość przestrzeni roboczej do pliku **`dane.mat`**.

Zawartość przestrzeni roboczej można wyczyścić poleceniem **`clear`**.

10. Użyj polecenia **`clear`**, aby usunąć zmienne z przestrzeni roboczej.

Do odczytania zmiennych z pliku można wykorzystać polecenie **`load`**.
Polecenie **`load nazwapliku`**, *wczyta dane zapisane w pliku* **`nazwapliku.mat`**.

11. Wczytaj zmienne z utworzonego wcześniej pliku **`dane.mat`**.
12. Ponownie wyczyść przestrzeń roboczą, a następnie wprowadź polecenie **`load data p`**. Co się stało?
13. Wczytaj ponownie wszystkie zmienne z pliku **`dane.mat`**, a następnie wprowadź polecenie **`save tylkop p`**.

Poleceniem **`clc`** *możesz wyczyścić Command Window.*

14. Wprowadź polecenie **`clc`**.

W środowisku Matlab dostępne są wbudowane stałe (np. stała **`pi`**).

15. Wyczyść przestrzeń roboczą, a następnie utwórz zmienną **`x`** o wartości π/4.

W środowisku Matlab dostępnych jest wiele funkcji wbudowanych (np. funkcja **`sin`**). *Argumenty funkcji podaje się w nawisach zwykłych.*

16. Utwórz zmienną **`y`** i przypisz jej wartość cosinusa (**`cos`**) zmiennej **`x`**.

Operatorem potęgowania w Matlabie jest ^.

17. Oblicz wartość kwadratu zmiennej **`y`**.
18. Użyj funkcji **`sqrt`** do obliczenia pierwiastka z **`-16`**. Obliczoną wartość przypisz do zmiennej **`z`**. Zauważ, że wynik zawiera liczbę urojoną **`i`**, która jest stałą Matlaba.

Sposób wyświetlania danych można zmieniać funkcją **`format`**.

19. Wprowadź polecenie **`format long`**, następnie wyświetl zawartość zmiennej **`x`**.
20. Wprowadź polecenie **`format short`**, aby powrócić do standardowego formatu.

2. Praca z edytorem

Praca w Command Window pozwala łatwo i szybko sprawdzić działanie wybranych poleceń. Do pracy z bardziej złożonymi problemami wykorzystuje się okno edytora (Editor).

21. Utwórz nowy skrypt i zapisz go pod nazwą **`po_101.m`**.
22. Umieść w skrypcie polecenia, które utworzą zmienną **`r`** o wartości **`0.5`**, a następnie obliczą pole koła o promieniu **`r`**. Uruchom skrypt.
23. Dodaj do skryptu polecenie, które obliczy obwód koła o promieniu **`r`**. Zmodyfikuj wartość **`r`**. Uruchom skrypt ponownie.
24. Wprowadź polecenie **`po_101`** w Command Window.

3. Tworzenie tablic

Jedna liczba (skalar) jest w Matlabie tablicą rozmiaru 1 na 1.

25. Wyczyść przestrzeń roboczą i utwórz zmienną **`x`** o wartości 3. Następnie wywołaj funkcję **`size`** przekazując **`x`** jako argument.

Tablicę można utworzyć wykorzystując nawiasy kwadratowe. Polecenie **`x = [2 3]`***, utworzy tablicę o zawierającą dwa elementy: 2 oraz 3.*

26. Utwórz zmienną **`x`** – tablicę zawierającą dwa elementy: 4 oraz 7.

Elementy tablicy oddzielone spacją lub przecinkiem będą umieszczone w wierszu. Natomiast elementy rozdzielone średnikiem będą umieszczone w kolumnie.

27. Utwórz zmienną **`x`** – wektor wierszowy zawierający dwa elementy: 5 oraz 6.
28. Utwórz zmienną **`x`** – wektor kolumnowy zawierający dwa elementy: 4 oraz 9.
29. Utwórz zmienną **`x`** – wektor wierszowy zawierający trzy elementy: -1, 2 oraz -9.
30. Utwórz zmienną **`x`** – wektor kolumnowy zawierający trzy elementy: 1, 0.1 oraz 0.5.

Wykorzystując spacje i średniki można utworzyć tablice prostokątne. Przykładowo polecenie **`x = [1 1 2; 3 5 8]`** *utworzy zmienną* **`x`** *o dwóch wierszach i trzech kolumnach.*

31. Utwórz zmienną **`x`** – tablicę rozmiaru 2 na 3 zawierającą elementy: 1, 2, 3; 4, 5, 6.

Elementy tablicy mogą być podawane jako wyrażenia.

32. Utwórz zmienną **`x`** – wektor wierszowy zawierający: $\sqrt{2}$ oraz π^3.

Często będziemy pracować wektorami zawierającymi równoodległe liczby.

33. Utwórz zmienną **`x`** – wektor wierszowy zawierający kolejno liczby: 1, 2, 3.

W przypadku długich wektorów można wykorzystać skróconą notację z dwukropkiem (:).

34. Sprawdź działanie polecenia: **`x = 3:7`**.

Domyślnie liczby odległe są o 1. Można podać własną wartość kroku. Polecenie **`x = [3:2:7]`** *utworzy zmienną x o elementach: 3, 5, 7.*

35. Utwórz zmienną **`x`** – wektor wierszowy zawierający liczby od 0 do 5 z krokiem 0.5.

Innym sposobem generowania wektorów jest wykorzystanie funkcji **`linspace`***, która pobiera trzy argumenty: pierwszy element, ostatni element oraz liczbę wszystkich elementów. Argumenty funkcji są oddzielane przecinkiem.*

36. Utwórz zmienną **`x`** – wektor wierszowy zawierający 11 liczb od 0 do 5.

Wektor wierszowy można zamienić w wektor kolumnowy wykorzystując operator transpozycji ‘.

37. Sprawdź działanie polecenia **`x = x'`**.

W środowisku Matlab dostępne są również funkcje, które pozwalają tworzyć przydatne macierze.

38. Sprawdź działanie polecenia **`x = rand(2)`**.
39. Sprawdź działanie polecenia **`x = rand(5)`**.
40. Sprawdź działanie polecenia **`x = rand(2,3)`**.
41. Sprawdź działanie polecenia **`x = ones(2,3)`**.
42. Sprawdź działanie polecenia **`x = ones(3)`**.
43. Sprawdź działanie polecenia **`x = zeros(2,3)`**.

44. Sprawdź działanie polecenia **`x = zeros(3)`**.
45. Sprawdź działanie serii poleceń
    `x = rand(2)`,
    `y = rand(size(x))`.

4. Odwoływanie się do elementów tablic.

W celu odwołania się do elementów tablicy możemy wykorzystać indeksy podawane w kolejności wiersz, kolumna. Indeksy są liczone od 1.

46. Utwórz zmienną **`x`** – tablicę liczb losowych rozmiaru 7 na 3. Następnie odczytaj wartość elementu znajdującego się w szóstym wierszu, w drugiej kolumnie.

Wykorzystując słowo kluczowe **`end`***, można łatwo odnieść się do ostatniego elementu wiersza lub kolumny.*

47. Utwórz zmienną **`y`** i przypisz jej wartość elementu tablicy **`x`** znajdującego się w ostatnim wiersze w drugiej kolumnie.

Słowo kluczowe **`end`** *można umieszczać w wyrażeniach arytmetycznych.*

48. Utwórz zmienną **`y`** i przypisz jej wartość elementu tablicy **`x`** znajdującego się w przedostatnim wierszu w drugiej od końca kolumnie.

Zastąpienie wybranego indeksu znakiem dwukropka powoduje zwrócenie wszystkich elementów w tym wymiarze.

49. Utwórz zmienną **`y`** i przypisz jej wartości z drugiej kolumny tablicy **`x`**.
50. Utwórz zmienną **`y`** i przypisz jej wartości z drugiego wiersza tablicy **`x`**.

Operator dwukropka pozwala definiować zakresy przy indeksowaniu.

51. Utwórz zmienną **`y`** i przypisz jej wartości z wierszy od 2 do 4 tablicy **`x`**.
52. Utwórz zmienną **`y`** i przypisz jej wartości z wierszy od 3 do ostatniego tablicy **`x`**.

Indeksy nie muszą być kolejnymi liczbami.

53. Utwórz zmienną **`y`** i przypisz jej wartości z wierszy 1, 3 oraz 7 tablicy **`x`**.

Indeksy mogą być wykorzystane do zmieniania wartości elementów w tablicy.

54. W tablicy **`x`** umieść wartość 3 w czwartym wierszu w drugiej kolumnie.
55. W tablicy **`x`** umieść wartość 3 w drugim wierszu w czwartej kolumnie.

5. Praca z tablicami

W środowisku Matlab wiele operacji można wykonywać na tablicach. Można do tablicy dodać skalar.

56. Wyczyść przestrzeń roboczą. Utwórz zmienną **`x`** zawierającą liczby 1, 2, 3. Następnie wykonaj polecenie **`y = x + 2`**.

Można również dodawać do siebie tablice (o takich samych rozmiarach).

57. Utwórz zmienną **`z`** zawierająca sumę tablic **`x`** oraz **`y`**.

Tablice można mnożyć i dzielić przez skalary.

58. Wykonaj polecenie **`a = x*3`**.
59. Wykonaj polecenie **`b = z/5`**.

Tablice mogą być argumentami funkcji (np. funkcji **`max`***).*

60. Wykonaj polecenie **`xMax = max(x)`**.
61. Wykonaj polecenie **`xSqrt = sqrt(x)`**.

Operacja mnożenie dla tablic jest mnożeniem macierzowym.

62. Spróbuj wykonać polecenie **`x = [2 3] * [4 5]`**. Czy operacja zakończyła się sukcesem?

*Mnożenie element po elemencie uzyskuje się operatorem .**

63. Wykonaj polecenie **`x = [2 3] .* [4 5]`**. Czy tym razem operacja zakończyła się sukcesem?

6. Wywoływanie funkcji

Niektóre funkcje mogą zwracać więcej niż jedną wartość. Funkcja **`size`** *może zwracać dwuelementowy wektor liczb lub dwie liczby zawierające liczbę wierszy oraz kolumn.*

64. Wyczyść przestrzeń roboczą. Wykorzystując funkcję **`rand`** utwórz dwie zmienne: **`x`** – siedmioelementowy wektor wierszowy oraz **`y`** – tablicę liczb rozmiaru 3 na 5.
    Następnie wykonaj polecenia:
    ```
    sx = size(x)
    sy = size(y)
    [xr, xc] = size(x)
    [yr, yc] = size(y)
    ```

Podobnie może zachowywać się funkcja **`max`***.*

65. Sprawdź działanie poleceń:
    ```
    [xMax, ixMax] = max(x)
    [yMax, iyMax] = max(y)
    ```

7. System pomocy

Informacje o działaniu funkcji wbudowanych można uzyskać z rozbudowanego systemu pomocy.

66. Sprawdź działanie poleceń:
    ```
    help randi
    doc randi
    ```
    Na podstawie informacji uzyskanych z systemu pomocy, stwórz zmienną **`x`** zawierającą tablicę losowych liczb całkowitych (z zakresu od 1 do 17) o rozmiarze 5 na 9.

8. Rysowanie wykresów

Do tworzenia wykresów danych można wykorzystać funkcję **`plot`**.

67. Wyczyść przestrzeń roboczą. Stwórz zmienną **`x1`** – wektor wierszowy zawierający liczby od -2 do 2 z krokiem 0.5. Następnie wykonaj polecenia:
```
y1 = x1.^2
plot(x1,y1)
```

Funkcja **`plot`** *może przyjmować dodatkowe argumenty, które określają kolor, styl linii oraz rodzaj znaczników punktów.*

68. Wykorzystując system pomocy narysuj wykres **`y1`** względem **`x1`** używając czerwonej linii przerywanej i znaczników kwadratowych.
69. Wykonaj następujące polecenia:
```
x2 = -2:2
y2 = x2.^3
plot(x2,y2,'k*')
```
Zwróć uwagę, że nowy wykres został umieszczony w tym samym oknie, ale poprzedni wykres nie jest już dostępny.

Wykorzystują funkcję **`hold`** *można spowodować zatrzymanie poprzednich wykresów.*

70. Wykonaj następujące polecenia:
```
hold on
plot(x1,y1,'r--s')
hold off
```

Funkcja **`plot`** *może przyjmować jeden wektor danych, wtedy na osi x umieszcza odpowiednie indeksy.*

71. Sprawdź działania polecenia **`plot(y2)`**
72. Sprawdź działania polecenia **`plot(y2,'LineWidth',3)`**
73. Sprawdź działania polecenia **`plot(y2,'ro-','LineWidth',5)`**

Do wykresów można łatwo dodawać opisy.

74. Zapoznaj się z dokumentacją funkcji **`title`**. Przetestuj jej działanie.
75. Zapoznaj się z dokumentacją funkcji **`xlabel`** oraz **`ylabel`**. Przetestuj ich działanie.
76. Zapoznaj się z dokumentacją funkcji **`legend`**. Przetestuj jej działanie.

9. Indeksowanie z wykorzystaniem wartości logicznych

W środowisku Matlab można wykorzystywać operatory porównania **`<, >, <=, >=, ==, ~=`**.

77. Wyczyść przestrzeń roboczą, a następnie utwórz zmienną **`test`**, która zawiera wynik sprawdzenia, czy π jest większe od 3.

Operatory porównania mogą działać na całym wektorze.

78. Utwórz (funkcją **`rand`**) pięcioelementowy wektor wierszowy **`x`**. Sprawdź działanie polecenia:
```
x < 0.5
```

Wektor wartości logicznych może być wykorzystany do indeksowania elementów w wektorze.

79. Sprawdź działanie polecenia: **`x(x < 0.5)`**.

W ten sposób można też modyfikować wartości elementów.

80. Sprawdź działania polecenie: **`x(x < 0.5) = 0`**.

Wektory wartości logicznych można łączyć wykorzystując operatory logiczne koniunkcji (&) oraz alternatywy (|).

81. Sprawdź działanie poleceń:
```
x = rand(1,5)
y = x(x > 0.2 & x < 0.8)
```

10. Proste struktury programistyczne

Do sterowania przebiegiem programów można wykorzystywać instrukcję warunkową **`if`** *oraz pętlę* **`for`**.

82. Napisz skrypt, który przypisuje zmiennej **`x`** losową wartość (z wykorzystaniem funkcji **`rand`**). Następnie przypisuje wartość zmiennej **`y`**: jeśli **`x > 0.5`**, to **`y = 1`** w przeciwnym przypadku **`y = -1`**. Uruchom skrypt kilka razy i sprawdź jego działanie.
83. Przenieś poniższe instrukcje do skryptu
```
hold on
for idx = 1:10
    plot(rand(),rand())
    pause(1)
end
```
Sprawdź jego działanie uruchamiając skrypt kilka razy.

Materiały opracowane na podstawie kursu on-line MATLAB Onramp dostępnego w ramach Matlab Academy.

https://matlabacademy.mathworks.com/R2020b/portal.html?course=gettingstarted

Karol Tarnowski
Wrocław, 2021