POLITECHNIKA KRAKOWSKA im. T. Kościuszki
Wydział Inżynierii Lądowej
ul. Warszawska 24, 31-155 Kraków
tel. 012 628 2546/2929, fax: 012 628 2034, e-mail: L-5@pk.edu.pl
Instytut Technologii Informatycznych w Inżynierii Lądowej
(L-5)

MATERIAŁY POMOCNICZE

DO PRACY W ŚRODOWISKU

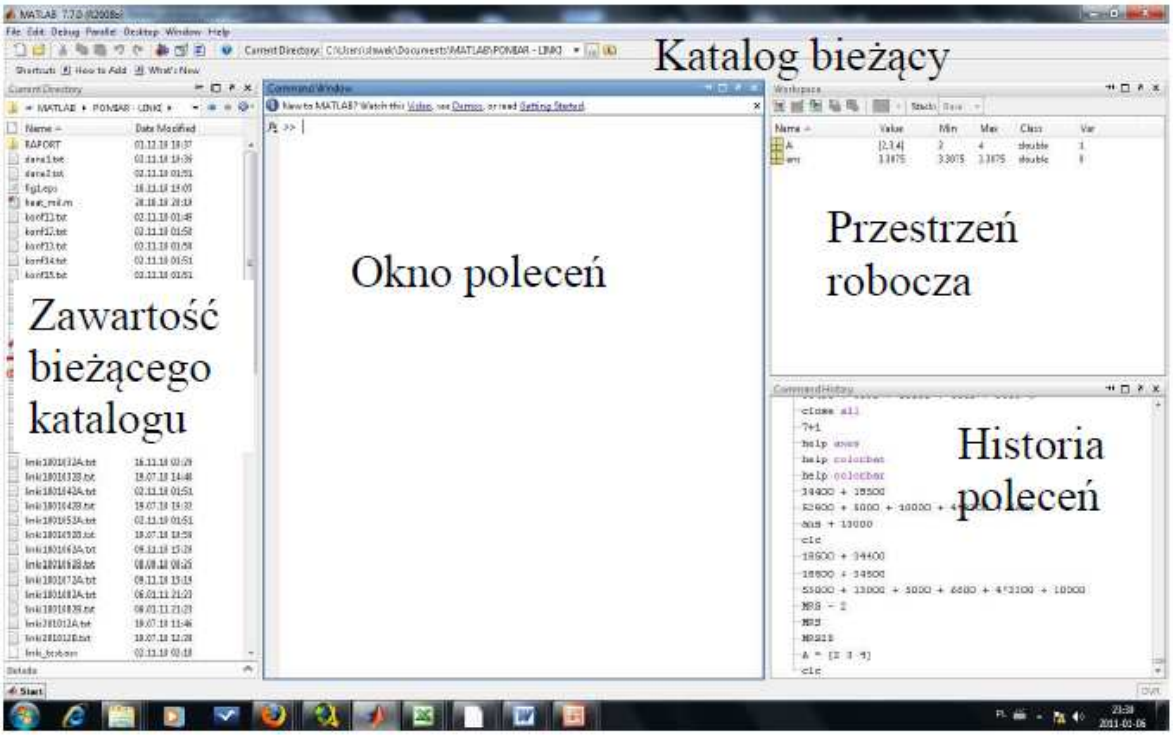
MATLAB

Przedmioty:
Informatyka, Matematyka Stosowana i metody numeryczne, Matematyka 2

Cz. I: WPROWADZENIE DO MATLABA

1. Wygląd interfejsu użytkownika

Po uruchomieniu programu Matlab w zależności od wersji i ustawień, pojawi się interfejs użytkownika. Przykładowy interfejs pokazano poniżej wraz z opisem najważniejszym składników okna programu. Domyślny wygląd interfejsu programu można zawsze przywrócić wybierając: Górne Menu: *Desktop: Desktop Layout: Default*.



2. Sposoby pracy w programie

Praca w Matlabie może mieć charakter

- wsadowy: wydawanie poleceń w oknie poleceń (Command Window), po znaku zachęty >> - polecenie jest natychmiastowo przetwarzane przez program i wykonywane: na ekranie może pojawić się rezultat lub komunikat o błędzie,
- skryptowy: poprzez tworzenie tzw. m-plików, czyli plików tekstowych zawierających ciąg poleceń Matlaba: nie są one wykonywane, dopóki użytkownik tego nie zażąda; nowy plik można utworzyć poprzez Górne Menu: File: New, a otworzyć istniejący z dysku: Górne Menu: File: Open. Za każdym razem pojawi się/zaktualizuje nowe (domyślnie niezadokowane) okno edytora z kolejnymi zakładkami oznaczającymi referencje do otwartych plików

Wydając polecenia w Matlabie można wykorzystywać: funkcje wbudowane w jądro (np. funkcje matematyczne), funkcje zebrane w tzw. przyborniki (toolbox) pogrupowane tematycznie lub własne funkcje (m-pliki) znajdujące się w katalogu bieżącym.

Na początku nauki Matlaba zalecane jest zaznajomienie z trybem pracy wsadowym, a potem stopniowe przechodzenie do tworzenia własnych, coraz bardziej złożonych m-plików.

3. Zasady pracy w trybie wsadowym

- podstawowym typem danych liczbowych w Matlabie jest tablica liczb zwana macierzą - jakakolwiek wartość liczbowa wpisana przez użytkownika w Oknie Poleceń (OP), będzie traktowana jako macierz

  ```
  >> a = 1
  ```

  Po napisaniu powyższego i naciśnięciu klawisza Enter, wykonanie polecenia poskutkuje utworzeniem zmiennej o nazwie "a" i nadaniem jej wartości "1". Jest to widoczne zarówno w postaci echa programu (wypisywanego z OP zaraz po wykonaniu polecenia) oraz w Przestrzeni Roboczej (Workspace) - na liście zmiennych widnieje zmienna "a", obok widnieje jej krótka charakterystyka symbol tablicy  , jej rozmiar (1x1), liczba bajtów (Bytes), na których zapisana jest zmienna (8), wartość (Value) oraz elementy ekstremalne (Max i Min). Widoczność tych pól i innych można włączyć/wyłączyć klikając PKM (Prawym Klawiszem Myszy) na nazwie jednego z nich.
- w Matlabie nie jest potrzebna deklaracja zmiennych, tzn. program sam niejawnie rozpoznaje, z jakim typem ma do czynienia. Typ liczbowy obejmuje liczby naturalne, całkowite, wymierne, niewymierne rzeczywiste i nierzeczywiste (zespolone). Wszystkie one zapisywane są na 8-16 bajtach. Konwersja na inny typ (np. całkowity - o mniejszej długości) musi być przeprowadzona przez użytkownika, np.

  ```
  >> c = int16(a)
  ```

- nieprzypisane wyniku działania do zmiennej nie skutkuje błędem: w takiej sytuacji tworzona lub aktualizowana jest zmienna standardowa o nazwie "ans", którą można posługiwać się jak każdą inną zmienną.

  ```
  >> a + 1
  >> ans + 3 + a
  ```

- rezultat operacji podstawienia będzie zawsze wypisywany przez program w postaci echa, chyba że linijka będzie zakończona znakiem średnika (;). W takiej sytuacji polecenie zostanie wykonane, ale rezultat nie będzie wyświetlony

```
>> b = a + 5;
```

Rezultat nie został wypisany, ale zmienna b została utworzona i ma wartość 6 - widać to chociażby w Przestrzeni Roboczej (Workspace). Wartość każdej zmiennej można też podglądnąć pisząc jej nazwę, np.

```
>> a
>> ans
>> b
```

- całą przestrzeń roboczą można zapisać na dysku poleceniem save

```
>> save zmienne
```

- usunąć jedną lub wiele zmiennych można poleceniem clear

```
>> clear a
>> clear all
```

Pierwsze clear usunie zmienną "a", drugie clear - całą Przestrzeń Roboczą.

- zapisaną przestrzeń roboczą można wczytać poleceniem load

```
>> load zmienne
```

- wszystkie wydane polecenia można łatwo przywracać w OP za pomocą strzałek góra - dół; strzałka górna zacznie przeszukiwanie historii poleceń od ostatnio wpisanego; jeżeli wcześniej wpisane zostaną jakieś znaki, strzałka górna zacznie wyszukiwać tylko te linijki, które się od tych znaków zaczynają. Dlatego też nie ma sensu wpisywać jeszcze raz poleceń, zwłaszcza tych długich, np. w sytuacji, gdy popełniło się wcześniej jakiś błąd składniowy. W tym przypadku należy przywrócić strzałką wcześniejszą linijkę i dokonać stosownej korekty.

- nazewnictwo zmiennych (jak również plików i funkcji) podlega takim samym ograniczeniom, jak w innych językach programowania: nie można zaczynać nazwy zmiennej od liczby, ani od żadnego innego znaku, który nie jest literą; wewnątrz nazwy również nie może być np. spacji. Poprawne nazwy zmiennych to np. a, a1, a_1, niepoprawne to z kolei 1a, a 1, _a1 itd. Należy również pamiętać, że Matlab rozróżnia wielkość liter, np. zmienne a i A to dwie różne zmienne.

4. Zmienne specjalne

- istnieje kilka nazw zarezerwowanych tzw. zmiennym specjalnym, wbudowanym w program, np. ans, computer, eps, i, j, Inf, Nan, nargin, nargout, pi, realmax, realmin

eps	- podaje dokładność maszynową (w zapisie wykładniczym) - jest to odległość od liczby 1 do następnej liczby rzeczywistej tej samej precyzji
i	- jednostka urojona (pierwiastek z -1)
j	- jw.
Inf	- symbol nieskończoności
Nan	- symbol nieoznaczony
nargin	- liczba argumentów aktualnych funkcji
nargout	- liczba wartości aktualnych funkcji
pi	- wartość liczby π
realmax	- największa liczba rzeczywista rozpoznawana przez program
realmin	- najmniejsza liczba rzeczywista rozpoznawana przez program

Wszystkie z nich (poza nargin i nargout, które są funkcjami logicznymi stosowanymi wewnątrz funkcji) można użyć bezpośrednio w OP. Np. zacznijmy od pi

```
>> pi
```

Wyświetlona zostanie wartość pi z dokładnością do czterech miejsc po przecinku, czyli 3.1416. Program pamięta 15 miejsc, ale wyświetla zgodnie z bieżącym formatem jedynie 4. Następnie wypiszmy eps

```
>> eps
```

Dokładność "eps" wynosi 2.2204e-016. Jak czytać taką liczbę? Mianowicie "e-016" oznacza to samo, co 10^{-16}, a liczba przed "e" to czynnik skalujący tę potęgę liczby 10. Dlatego też w klasycznym zapisie oznacza to liczbę $2.2204 \cdot 10^{-16}$ lub 0.0000000000000022204 (po przecinku jest 15 zer), czyli bardzo małą liczbę.

```
>> i
```

Wypisując "i" zobaczymy na ekranie zapis zespolony jednostki urojonej, 0 + 1.0000i, co oznacza część rzeczywistą równą "0", oraz część urojoną równą "1". "i" nie ma sensu liczbowego, jest to symboliczne określenie pierwiastka z "-1" (czyli $i = \sqrt{-1}$). Taki zapis pojawi się zawsze wtedy, gdy np. pierwiastek parzystego stopnia zostanie wyciągnięty z wartości ujemnej. Np. proszę wpisać

```
>> sqrt(-2)
```

Funkcja "sqrt" (ang. square root - pierwiastek kwadratowy) oblicza pierwiastek kwadratowy z argumentu podanego w nawiasie. Wartość "0 + 1.4142i" oznacza $i\sqrt{2}$. Zamiast funkcji "sqrt" można wykorzystać znak karety "^" ("daszek", "dzióbek"), czyli

```
>> (-2)^0.5
```

Przy tej okazji warto zwrócić uwagę na konieczność zastosowania nawiasów w podstawie potęgi. Potęgowanie "^" ma wyższy priorytet niż mnożenie przez "-

1", dlatego też wyrażenie bez nawiasów będzie oznaczało podniesienie "2" do potęgi "0.5", a następnie zmianę znaku wyniku

```
>> -2^0.5
```

Podobnie jeżeli zamiast wykładnika "0.5" napiszemy "1/2", to "1/2" musi znaleźć się w nawiasach

```
>> (-2)^(1/2)
```

Inaczej

```
>> (-2)^1/2
```

najpierw podniesiemy "-2" do potęgi "1", a potem wynik podzielimy przez "2".

Wróćmy do zmiennych specjalnych.

```
>> j
```

ma takie samo znaczenie jak "i", jest tylko "przezwiskiem" tej zmiennej.

Symbol "Inf" (ang. infinity - nieskończoność) pojawia się wszędzie tam, gdzie podzielimy przez zero, lub liczbę bardzo bliską zeru, np.

```
>> 1/0
```

```
>> 1/10^-400
```

Symbol nieoznaczony "NaN" (ang. Not a Number - "nieliczba") odpowiada większości sytuacji nierozstrzygniętych, np. przy liczeniu granic. Wtedy w zależności od przekształceń wynik finalny może być skończony (liczba) lub nieskończony.

```
>> 0/0
```

```
>> inf/inf
```

```
>> inf*0
```

```
>> Inf - inf
```

Natomiast wg Matlaba symbolami nieoznaczonymi nie są wyrażenia potęgowe z wykładnikiem "0", np.

```
>> 0^0
```

```
>> inf^0
```

Za każdym razem otrzymamy wartość "1". Wynika to z kodowania operacji potęgowania, w którym elementem startowym jest wartość "1", od której

iloczyn zwiększany jest o kolejne człony - w przypadku potęgi "0" liczba tych członów jest równa zeru.

```
>> realmax
```

```
>> realmin
```

Powyższe zmienne przechowują odpowiednio największą i najmniejszą dodatnią wartość rzeczywistą. Aby pokazać różnicę pomiędzy zmienną eps i zmienną realmin, wprowadźmy do programu dwie zmienne

```
>> a = 0.00000000010000000000000002
>> b = 0.0000000001
```

Po wprowadzeniu powyższych linijek pokażą się wartości zmiennych, każda po "1.0000e-010". Wygląda na to, iż są one równe sobie. Fakt ten potwierdzi się, jeżeli napiszemy porównanie (uwaga na podwójny znak równości "==" !)

```
>> a == b
```

Podwójny znak równości oznacza nie przypisanie, ale porównanie dwóch liczb ze sobą. Jeżeli będą one równe, wynik takiej operacji to "1" (prawda), jeżeli różne od siebie, to "0" (fałsz). W naszym przypadku wynik to "1". Liczby okazały się równe, mimo iż wprowadzone były jako różne. Jednakże różnica wystąpiła na tak dalekim miejscu, iż Matlab nie był w stanie zapamiętać tego niezerowego miejsca w liczbie "a" posługując się zmienną podwójnej precyzji (o dokładności "eps") - zapamiętał wykładnik potęgi o podstawie "10" (tu "-10") odpowiadający pierwszej niezerowej cyfrze po przecinku ("1") oraz czynnik skalujący potęgę do 15tu miejsc włącznie ("1"). Natomiast po wpisaniu

```
>> a = 0.000000000000000000000000002
```

program zapamięta miejsce "-27", oraz czynnik skalujący "2", gdyż mieści się to w granicach zmiennej "realmin".

- przypisanie innych wartości powyższym nazwom nie spowoduje błędu, np.

```
>> i = 1
```

ale od tej pory wartość zmiennej i będzie równa "1". Przywrócenie znaczenia domyślnego jest możliwe poprzez usunięcie zmiennej (clear)

```
>> i
>> clear i
>> i
```

5. Formatowanie wyświetlania wyników

- formatowanie postaci, z jaką będą się pojawiać na ekranie wartości wyrażeń lub podstawień, można ustawiać za pomocą funkcji "format"

```
>> format short
```

lub

```
>> format
```

oznacza powrót do formatu domyślnego, który obowiązuje od początku startu nowej sesji w Matlabie. Wprowadźmy do pamięci zmienną

```
>> a = 2/100
```

Ustawmy następujący format

```
>> format long
```

Nic się nie zmieniło, ale jeżeli teraz wyświetlimy jakąkolwiek wartość, zobaczymy 15, a nie 4 miejsca po przecinku

```
>> a
>> pi
```

Należy w tym miejscu nadmienić, iż taka wartość „pi” jest jednak wartością przybliżoną, gdy program nie pamięta wszystkich miejsc po przecinku (jest ich przecież nieskończenie wiele). Jest to jednak tak dobra dokładność, iż może być ona stosowana we wszystkich obliczeniach w Matlabie prowadzonych w podwójnej precyzji.
Ustawmy format

```
>> format long e
```

Od tej formy wartości będą wypisywane zgodnie z notacją wykładniczą ("e" - potęga o podstawie "10"), z 15toma miejscami po przecinku

```
>> a
>> pi
```

```
>> format short e
```

```
>> a
>> pi
```

lub na powrót czterema. Z kolei

```
>> format short g
```

oznacza format optymalny dla danej liczby z pominięciem zer od strony prawej począwszy nie stanowiących zaokrąglenia

```
>> a
>> pi
```

Kolejny format

```
>> format rat
```

ustawia wypisywanie liczb w postaci ilorazu dwóch możliwie małych liczb całkowitych (w postaci ułamka zwykłego)

```
>> a
>> pi
```

Wartość wymierna "355/113" oznacza jedynie przybliżenie liczby "pi" (większe przybliżenie niż samo „pi”), która jak wiemy jest liczba niewymierną i nie da się jej przedstawić w postaci ułamka zwykłego. Aby przekonać się, jak dobre jest to przybliżenie, wróćmy do formatu długiego

```
>> format long
```

i wypiszmy wartość przybliżoną oraz „dokładną”

```
>> 355/113
>> pi
```

Na pierwszy rzut oka bardzo podobne, ale jednak różniące się na 7mym miejscu po przecinku. W celu lepszego porównania można stworzyć normę błędu

```
>> abs(355/113 - pi)
```

a wynik pokaże od razu dokładność oszacowania liczby "pi". Wykładnik liczby "10" w notacji wykładniczej świadczy wprost o poziomie dokładności (błędu). Funkcja "abs" (ang. absolute value) oblicza wartość bezwzględną (moduł) danej liczby.

Powróćmy do standardowego formatu krótkiego

```
>> format short
```

Inne możliwe formaty, np.

```
>> format compact
>> a
>> format loose
>> a
```

odpowiednio odejmują lub dodają po jednej pustej linii pomiędzy odpowiedzią programu na wydane polecenie.

6. Podstawowe funkcje matematyczne

- funkcje matematyczne, które pojawiły się do tej pory, to "sqrt" i "abs". Pozostałe funkcje to np. funkcje trygonometryczne

```
>> sin(0)
>> sin(pi)
```

```
>> cos(0)
>> cos(pi/2)
>> tan(0)
>> tan(pi/2)
>> cot(0)
```

Argumenty tych funkcji podawane są w radianach. Jeżeli np. jest potrzeba policzenia wartości sin(2°), należy zamienić argument w stopniach na radiany, czyli

```
>> sin(2*pi/180)
```

Zauważmy, iż kilka wartości powyższych funkcji zostało podanych w przybliżeniu, np. "sin(pi)" to nie idealne 0, ale " 1.2246e-016". Przyczyna tkwi oczywiście w przybliżeniu liczby "pi", która jako niewymierna, nie może być pamiętana "w całości" przez program. Ta sama uwaga dotyczy wartości "tan(pi/2)", gdzie dla idealnej wartości π/2 powinna być odpowiedź "inf" lub "-inf", jako iż funkcja tangens posiada w wielokrotności całkowitej π/2 asymptotę pionową. Wartość "cot(0)" obliczana od idealnego zera, właśnie taką odpowiedź daje.

```
>> pow2(5)
>> pow2(1)
>> pow2(-0.5)
>> pow2(0.5)
```

to funkcja obliczająca 2 do danej potęgi, w tym przypadku do potęgi "5".

```
>> sign(-45)
>> sign(34)
>> sign(0)
```

to funkcja zwracająca znak ("1" lub "-1" lub "0" dla zera) liczby.

```
>> factorial(0)
>> factorial(1)
>> factorial(2)
>> factorial(3)
>> factorial(10)
```

to funkcja obliczająca silnię, czyli iloczyn kolejnych liczb naturalnych. Jest to funkcja określona tylko dla liczb naturalnych, dlatego też przy próbie wywołań

```
>> factorial(-5)
>> factorial(0.5)
```

pojawią się błędy.

```
>> asin(1)
```

to przykład wywołania funkcji arcus sinus, funkcji odwrotnej do funkcji sinus, którą należy rozumieć następująco: jeżeli znamy wartość sinus jakiegoś kąta i

chcemy dowiedzieć się, jaki to kąt, należy obliczyć arcus sinus z sinusa i w rezultacie otrzymamy wartość kąta w radianach. Np. jeżeli powyższy wynik pomnożymy przez "2"

```
>> 2*asin(1)
```

to otrzymamy kąt π, ponieważ dla kąta π/2 sinus wynosi "1".

```
>> sin(pi/2)
```

Podobnie działają funkcje acos i atan

```
>> acos(0)
>> atan(1)
>> 4*atan(1)
```

Ostatnia wartość to kolejne możliwe przybliżenie liczby π, jako że tg(π/4) = 1.

```
>> tan(pi/4)
```

Funkcja "exp" podnosi liczbę Eulera "e" do danej potęgi.

```
>> exp(2)
>> exp(-3)
```

Liczba Eulera (podstawa logarytmów naturalnych) to liczba niewymierna wynosząca ok. 2.7183. Nie ma w Matlabie zmiennej "e" zawierającej tę wartość, jednakże można ją stworzyć na własne potrzeby podnosząc "e" do potęgi "1".

```
>> e = exp(1)
```

Funkcja

```
>> log(100)
```

nie oznacza logarytmu o podstawie "10", gdyż wtedy wynik musiałby być równy 2. Jest to logarytm naturalny o podstawie "e".

```
>> log(e^-45)
>> log(e)
```

Logarytm dziesiętny należy obliczać za pomocą

```
>> log10(100)
>> log10(0.0001)
```

Istnieje jeszcze logarytm o podstawie "2"

```
>> log2(8)
>> log2(1/16)
```

Logarytmy o innych podstawach należy obliczać za pomocą wzoru na zmianę podstawy, czyli $\log_a b = \frac{\log_c b}{\log_c a}$. Np. do obliczenia log3(9) można wykorzystać jeden z istniejących już logarytmów (log, log10, log2):

```
>> log10(9)/log10(3)
>> log(9)/log(3)
>> log2(9)/log2(3)
```

Zauważmy gorszą dokładność wyniku („nieidealna” wartość „2”) opartego na logarytmie naturalnym, w którym podstawą jest liczba niewymierna "e" - Matlab również w tym przypadku opiera się jedynie na zaokrągleniu (reprezentacji skończonej) liczby „e”.

7. Sposoby zaokrąglania liczb rzeczywistych

- liczby rzeczywiste można zaokrąglać do postaci całkowitej za pomocą kilku funkcji. Klasyczne zaokrąglenie realizuje funkcja "round"

```
>> round(4.7)
>> round(4.2)
```

Zawsze w dół do mniejszej całkowitej zaokrągla "floor"

```
>> floor(4.7)
>> floor(4.2)
```

a zawsze w górę, do największej całkowitej - "ceil"

```
>> ceil(4.7)
>> ceil(4.2)
```

Jest jeszcze funkcja "fix", która dla liczb dodatnich działa tak samo, jak "floor"

```
>> fix(4.7)
>> fix(4.2)
```

ale inaczej dla ujemnych - zaokrągla zawsze w kierunku zera

```
>> floor(-4.7)
>> floor(-4.2)
>> fix(-4.7)
>> fix(-4.2)
```

Funkcja "rem" oblicza resztę z dzielenia - jest funkcją dwuargumentową - podajemy dzielną i dzielnik, oddzielając je znakiem przecinka - dlatego też w poniższym zapisie nie jest to jedna liczba 24.3, a dwie 24 i 3.

```
>> rem(24,3)
>> rem(25,3)
>> rem(26,3)
```

Z kolei dzielenie całkowite (bez reszty) można w Matlabie zawsze zrealizować przy pomocy funkcji "fix"

```
>> fix(26/3)
```

Warto wymienić znaki podstawowych działań: +, -, *, ^, /, \
Znak "/" oznacza klasyczne prawostronne dzielenie - pierwszego argumentu przez drugi

```
>> 2/5
```

Z kolei zapis odwrotny, za pomocą operatora lewostronnego dzielenia "\", powoduje zamianę dzielnika z dzielną:

```
>> 2\5
>> 5/2
```

8. Definiowanie macierzy

- deklarowanie tablicy, podstawowego nośnika danych w Matlabie, może odbywać się na kilka sposób. Pierwszy z nich to podanie wszystkich jej elementów w nawiasach prostokątnych [ ], wierszami, oddzielanymi od siebie znakiem średnia (";"), np.

```
>> A = [2 3 4; 5 6 7]
>> B = [1 2 3; -1 0 1; 2 4 6];
```

Tablice liczbowe w Matlabie stanowią informatyczny model tzw. macierzy, tablicy prostokątnej liczb rzeczywistych. A stanowi przykład macierzy prostokątnej (dwa wiersze, trzy kolumny), B - macierzy kwadratowej (trzy wiersze i trzy kolumny).
Wyrazy w wierszach mogą być oddzielane od siebie znakiem spacji lub za pomocą przecinka - można stosować notację jednorodną lub mieszaną

```
>> A = [2 3 4; 5 6 7]
>> A = [2,3,4; 5,6,7]
>> A = [2 3,4; 5,6 7]
```

Liczba wyrazów w każdym z wierszy musi być taka sama - w innym przypadku wystąpi błąd

```
>> A = [1 2 3; 4 5]
```

Istnieje w Matlabie sposób składowania takich zbiorów za pomocą tzw. tablic komórkowych

```
>> Z = {'123'; '45'}
```

lecz przewidziane są one przede wszystkim dla zmiennych znakowych, a nie liczbowych.

Drugi sposób definiowania tablic liczbowych wiąże się z wykorzystaniem tzw. zakresu. Jeżeli elementy tablicy układają się w szereg arytmetyczny, można to zapisać skrótowo, zamiast podawać wszystkie wyrazy szeregu, wystarczy podać wyraz pierwszy, różnicę i wyraz ostatni

```
>> a = 1:10
```

oznacza zbudowanie wektora wierszowego (tablicy jednowymiarowej) składającego się z liczb od 1 do 10. Jeżeli różnica szeregu ma być inna niż "1" (wartość domyślna), musi być to podane pomiędzy wartością pierwszą i ostatnią

```
>> a = 1:2:20
```

Zakres może być stosowany w wielu wierszach macierzy np.

```
>> A = [1:10; 1:2:20]
```

byleby liczba elementów w każdym z wierszy była taka sama. Dlatego też poniższy zapis jest nieprawidłowy:

```
>> A = [1:10; 1:2:25]
```

Różnica (skok) pomiędzy wyrazami może być ujemny, ale musi być wtedy jawnie określony

```
>> a = 10:-1:1
>> A = [10:-1:1; 20:-2:1]
```

Jeżeli o tym zapomnimy

```
>> a = 10:1
```

powstanie tzw. macierz pusta ([ ]).

Trzecim sposobem deklaracji macierzy jest zbudowanie jej z mniejszym przystających rozmiarami elementów

```
>> A = [1 2; 3 4];
>> B = [4 5; -1 2];

>> C = [A B]
>> D = [A; B]
```

Czwarty sposób wiąże się z wykorzystaniem różnych specjalnych funkcji tworzących gotowe tablice

```
>> zeros(4,3)
```

tworzy macierz zerową (wypełnioną samymi zerami) o wymiarach 4 wiersze i 3 kolumny. Zapis

```
>> zeros(4)
```

nie stworzy wektora o czterech elementach, ale macierz kwadratową 4x4. Jeżeli chcemy stworzyć wektor, należy wyraźnie zaznaczyć, który wymiar ma być równy "1"

```
>> zeros(4,1)
>> zeros(1,4)
```

Podobne funkcje to

```
>> ones(4,5)
>> ones(5)
>> ones(6,1)

>> eye(5,5)
>> eye(2,7)
>> eye(4)
>> eye(1,4)
```

"ones" tworzy macierz jedynkową (same jedynki), a "eye" - jednostkową (jedynki na przekątnej głównej, zera poza nią). Macierz jednostkowa ma specjalne znaczenie w rachunku macierzowym - stanowi element neutralny dla mnożenia (odpowiednik "1" przy mnożeniu liczb).
Funkcja "rand" produkuje zestaw rzeczywistych liczb pseudolosowych (generowanych wg danego klucza związanego np. z czasem systemowym) z przedziału otwartego (0,1)

```
>> rand(5,1)
>> rand(1,6)
>> rand(4,5)
>> rand(6,6)
```

"Surowe" wartości losowe można obrabiać na rozmaite sposoby, np. poprzez skalowanie lub/i zwiększanie, np.

```
>> rand(4,4)*10
```

oznacza zestaw 4x4=16 liczb rzeczywistych z przedziału (0,10)

```
>> round(rand(4,4)*10)
```

oznacza zestaw 4x4=16 liczb całkowitych z przedziału [0,10]

```
>> rand(4,4)*5 + 2
```

oznacza zestaw 4x4=16 liczb rzeczywistych z przedziału (2,7). Mnożenie tablicy przez liczbę oznacza pomnożenie każdego jej elementu przez tą liczbę, a dodanie do macierzy liczby oznacza zwiększenie każdego jej elementu o tę liczbę.
Funkcja ta jest bardzo pomocna przy testowaniu algorytmów macierzowych. Wygenerowane w ten sposób macierze stanowią duże wyzwanie dla takich algorytmów, gdyż składają się z przypadkowych wartości i często decydują o ich zbieżności i stabilności.

Funkcja "diag", jeżeli przyjmuje argument wektorowy, buduje macierz diagonalną, z elementami wektora ułożonymi na przekątnej głównej

```
>> diag([1 3 5])
```

Z kolei "diag" zastosowane dla argumentu macierzowego zwraca wektor złożony z wyrazów na przekątnej głównej tej macierzy

```
>> diag(A)
>> diag(B)
```

Stwórzmy macierz złożoną z losowych liczb całkowitych

```
>> C = round(10*rand(4,4))
```

Jak zmodyfikować macierz C, za pomocą instrukcji mieszczącej się w jednej linii, tak aby elementy przekątniowe macierzy C zwiększyć 10 razy?

```
>> C - diag(diag(C)) + diag( diag(C)*10 )
```

W powyższym zapisie "C - diag(diag(C))" pozwala na wyzerowanie elementów pozaprzekątniowych, a "+diag(diag(C)*10)" na umieszczenie tam elementów oryginalnych przemnożonych przez 10.
Jako ćwiczenie proszę spróbować teraz zwiększyć elementy przekątniowe o 10.

9. Edycja elementów macierzy

- z gotowych macierzy można wybierać elementy pojedynczo, wierszami, kolumnami, lub całymi zestawami wyrazów złożonych z wielu wierszy/kolumn. Na początek stwórzmy w Matlabie macierz C

```
>> C = [1 3 4 6; -1 5 6 9; 9 10 -4 0]
```

Dostęp do pojedynczych elementów jest możliwy poprzez zapis

```
>> C(2,3)
```

gdzie po nazwie macierzy w nawiasach okrągłych oddzielone przecinkiem (koniecznie!) występują numer wiersza i kolumny, z których to chcemy "wyciągnąć" pojedynczy wyraz.
Sięgnięcie po wyraz, który nie istnieje, poskutkuje błędem

```
>> C(5,2)
```

ale co ciekawe, wpisanie czegoś nowego do tego nieistniejącego miejsca błędem już nie będzie - Matlab sam zwiększy wymiar macierzy, wypełni dodatkowe miejsca zerami i wpisze do określonej lokalizacji podaną wartość

```
>> C(5,2) = -7
```

Wybranie za jednym zamachem wszystkich wyrazów jednego wiersza realizujemy za pomocą struktury ze znakiem dwukropek (":")

```
>> C(2,:)
>> C(5,:)
```

Podobnie z dostępem do kolumny

```
>> C(:,1)
>> C(:,3)
```

Znak ":" oznacza, iż chcemy wyciągnąć wszystkie wyrazy w danym wierszu/kolumnie.
Jeżeli na raz chcemy wydostać więcej niż jeden wiersz, numery wierszy należy objąć nawiasami kwadratowymi [ ]

```
>> C([2 5], :)
```

Podobnie kolumny

```
>> C(:, [1 4])
```

Jeżeli chcemy wyciągnąć numery wierszy z określonego zakresu, np. tylko wiersze o numerach nieparzystych, można napisać

```
>> C(1:2:5, :)
```

lub

```
>> C(1:2:end, :)
```

gdzie funkcja "end" sama sprawdzi, jaki numer ma ostatni wiersz.
Na miejscu ":" może pojawić się dodatkowa specyfikacja kolumnowa, np. chcemy wydostać wyrazy ze wszystkich wierszy parzystych, ale z kolumny pierwszej i ostatniej

```
>> C(2:2:end, [1 end])
```

W końcu możliwy jest zapis

```
>> C([1:2:end 2], :)
```

co oznacza wyciągnięcie wszystkich nieparzystych wierszy oraz dodatkowo wiersza nr 2, który jednak będzie dołączony na końcu nowej macierzy (ans).
Zapis

```
>> C(:)
```

oznacza zamianę macierzy na wektor kolumnowy, z wyrazami ułożonymi zgodnie z biegiem kolumnowym.

```
>> C(3)
```

nie oznacza wyboru trzeciego wiersza bądź kolumny, ale trzeciego wyrazu licząc wzdłuż kolumny.
Wyzerowanie całego wiersza wymaga napisania

```
>> C(5,:) = 0
```

a jego usunięcie (z pamięci) przypisania macierzy pustej ([ ])

```
>> C(5,:) = []
```

Taki zapis pozwala tylko na usuwanie całych wierszy bądź całych kolumn, ale nie na usuwanie pojedynczych wyrazów.

10. Działania macierzowe

- stwórzmy lub przywróćmy wcześniej stworzone macierze postaci

```
>> A = [1 2; 3 4]
>> B = [3 2;-2 3]
>> C = [3 4; 5 6; -1 2; 8 9]
>> D = [1 2 3; -1 0 1; 2 4 6]
```

- dodawanie i odejmowanie macierzy: dodawane i odejmowane mogą być tylko macierze o tych samych rozmiarach. Wynik w postaci kolejnej macierzy to zbiór sum lub różnic odpowiadających sobie elementów (czyli pierwszego z A i pierwszego z B, drugiego z A i drugiego z B itd.).

```
>> A + B
>> A – B
>> B – A
```

Ogólnie

$$c_{i,j} = a_{i,j} \pm b_{i,j} \quad , \quad i = 1...m \quad , \quad j = 1...n$$

Próba dodania do siebie lub odjęcia od siebie macierzy o różnych wymiarach poskutkuje błędem

```
>> A + C
>> B – D
```

- mnożenie macierzy przez liczbę to pomnożenie każdego jej elementu przez tę liczbę. Podobnie działa dzielenie macierzy przez liczbę

```
>> 2*A
>> 0.5*B
>> (1/pi)*C
>> D/4
```

- dodanie do macierzy liczby nie spowoduje błędu (dodajemy macierz do skalara), ale dodanie tej liczby do każdego elementu tej macierzy

```
>> D + 4
>> pi – A
```

- mnożenie macierzowe – tu sprawa jest bardziej skomplikowana, gdyż wyniku już nie można interpretować jako iloczynów odpowiadających sobie wyrazów

```
>> A*B
```

co więcej, mnożenie macierzy okazuje się nie być przemienne

```
>> B*A
```

w jaki więc sposób jest wykonywane? Mianowicie wiersze macierzy pierwszej mnożone są przez kolumny macierzy drugiej – dlatego też wyrazów w wierszach macierzy pierwszej musi być tyle samo, co wyrazów w kolumnach macierzy drugiej. Np. przyjrzyjmy się wynikowi A*B. Pierwszy wyraz macierzy wynikowej 2x2 to „-1”. Jak została ta wartość policzona? Wyraz ten leży w pierwszym wierszy i pierwszej kolumnie – to znaczy, że złożyły się na niego pierwszy wiersz A (czyli [1 2]) oraz pierwsza kolumna B (czyli [3;-2]). Wykonane przez program działanie to 1*3 + 2*(-2), co daje 3-4 = -1. Pozostałe elementy macierzy wynikowej są tworzone w ten sam sposób. Np. wartość „18” to złożenie drugiego wiersza A i drugiej kolumny B, czyli [3 4] z [2;3], co daje w iloczynie 3*2 + 4*3 = 6+12 = 18. Łatwo zauważyć, że za każdym razem mnożymy przez siebie skalarnie dwa wektory. I właśnie mnożenie macierzowe, jeżeli jest wykonalne, stanowi zbiór takich mnożeń skalarnych wektora (wierszowego) z wektorem (kolumnowym). Proszę spróbować samodzielnie zinterpretować wynik mnożenia B*A, przez sprawdzenie ręczne wszystkich wyrazów macierzy wynikowej.
Wykonywalność mnożenia macierzowe wiąże się zatem z wewnętrznymi rozmiarami czynników mnożenia. Można to zapisać w skrócie w sposób następujący $\underset{[m\times n]}{A}\cdot\underset{[n\times p]}{B}=\underset{[m\times p]}{C}$ w tzw. notacji globalnej (macierzowej) lub na poziomie elementów (notacja wskaźnikowa)

$$c_{i,j}=\sum_{k=1}^{n}a_{i,k}\cdot b_{k,j}\quad,\quad i=1...m\quad,\quad j=1...p$$

Warto zwrócić uwagę na fakt, iż macierz wynikowa C „nie pamięta” wymiaru „n” – tego, który musi być równy w macierzy A (liczba kolumn) i macierzy B (liczba wierszy).
Próba pomnożenia macierzy, które nie spełniają powyższej zasady, zakończy się błędem

```
>> A*C
>> B*C
>> D*C
>> C*D
```

ale można wykonać mnożenia

```
>> C*A
>> C*B
```

Warto jeszcze raz podkreślić: mnożenie macierzy nie jest przemienne. Nawet jeżeli uda się go wykonać „w dwie strony", to wynik w ogólnym przypadku będzie różny. Tylko dla specyficznych postaci macierzy czynnikowych można uzyskać to samo niezależnie od kolejności mnożenia (np. dla macierzy diagonalnych)

```
>> diag([1 2 3]) * diag([-1 0 3])
>> diag([-1 0 3]) * diag([1 2 3])
```

- transpozycja macierzy: oznacza wzajemną zamianę ze sobą wyrazów w wierszach z wyrazami w kolumnach. W zapisie algebraicznym transpozycję zapisujemy za pomocą notacji A^T, czego nie należy czytać „A to potęgi T", ale „A transponowane". Operacja ta zmienia wymiary macierzy $\underset{[m\times n]}{A} \rightarrow \underset{[n\times m]}{A^T}$. W Matlabie realizujemy ją nie za pomocą zapisu „A^T", ale za pomocą A' („A prim"), co nie oznacza w tym przypadku pochodnej

```
>> A
>> A'
>> B
>> B'
>> C
>> C'
>> D
>> D'
```

Powyżej wyświetlone są oryginale macierze oraz ich transpozycje. W przypadku macierzy kwadratowych (A, B, D) interpretacja transpozycji jest jeszcze prostsza – miejscami zamieniają się wyrazy pozaprzekątniowe, czyli wyraz np. A(1,2) z wyrazem A(2,1) itd. Wyrazy na przekątnej głównej pozostają bez zmian.

Macierz „niewrażliwa" na operację transponowania to np. macierz symetryczna (wtedy $A^T = A$). W takiej macierzy wyrazy pozaprzekątniowe są sobie parami równe (ogólny warunek symetrii wyrazów macierzy A wymiarze $m \times n$ wygląda następująco: $a_{i,j} = a_{j,i} \quad i = 1...m\,,\ j = 1...n$)

```
>> E = [1 2 5; 2 8 –3; 5 –3 –10]
>> E'
```

Macierz E stanowi przykład macierzy symetrycznej – macierz E^T do niej transponowana jest identyczna z macierzą E.
Warunek przeciwny spełnia tzw. macierz anty-symetryczna (lub skośnie-symetryczna). Wtedy $A^T = -A$ oraz na poziomie elementów: $a_{i,j} = -a_{j,i} \quad i = 1...m\,,\ j = 1...n$. Poniższa macierz

```
>> E = [1 2 5; –2 8 –3; –5 3 –10]
```

```
>> -E'
```

nie stanowi jednak poprawnego przykładu macierzy anty-symetrycznej, gdyż macierz do niej transponowana, ze znakiem „-„ nie jest identyczna z E. Co w takim razie należy poprawić? Zgodnie z wymogiem równości przeciwnej elementów, równość ta musi obowiązywać wszędzie, także na przekątnej głównej. I to właśnie tam musi być taka liczba, aby $a_{1,1} = -a_{1,1}$, $a_{2,2} = -a_{2,2}$ itd. Jedyną taką liczbą jest zero, i to właśnie zero musi być wszędzie na przekątnej głównej macierzy, jeżeli ma być ona anty-symetryczna.

```
>> E = [0 2 5; –2 0 –3; –5 3 0]
>> -E'
```

Uwaga: sprawdzanie równości dwóch macierzy (w tym przypadku E oraz $-E^T$) nie musi odbywać się na zasadzie „na oko” – porównywanie element po elemencie przez ich wyświetlenie w OP. Można np. zbudować następującą konstrukcję

```
>> E + E'
```

która powinna dać w rezultacie (jeżeli macierze E i –E' są sobie równe) same zera, co łatwiej zaobserwować. Innym sposobem jest przytoczone wcześniej porównanie za pomocą podwójnego znaku równości „==”

```
>> E == -E'
```

które, jeżeli jest prawdziwe, wyprodukuje macierz składającą się z samych jedynek, co oznacza, że porównanie równości każdej pary elementów daje wartość logiczną prawdziwą.

Wróćmy do mnożenia macierzowego. Zdefiniujmy dwa wektory wierszowe („leżące”)

```
>> a = [1 2 3]
>> b = [-1 0 1]
```

Zadanie jest następujące: należy obliczyć ich iloczyn skalarny, czyli sumę iloczynów odpowiadających sobie elementów tych wektorów.

$$a \circ b = \sum_{i=1}^{n} a_i b_i$$

Rezultatem iloczynu skalarnego dla wektorów a i b powinna być liczba 1*(-1) + 2*0 + 3*1 = 2. Jeżeli pomnożymy te wektory przez siebie bezpośrednio, czyli

```
>> a*b
```

to działanie nie wykona się (wyświetli się komunikat o błędzie), gdyż wymiary tych wektorów [1x3] oraz [1x3] nie pozwalają na ich pomnożenie – zgodnie z podanymi wcześniej regułami mnożenia macierzowego, które dotyczy także wektorów. W takim razie należy te wektory ustawić względem siebie tak, aby a) działanie powiodło się b) dało w rezultacie skalar (tablicę 1x1). Należałoby

zapewnić następującą konstrukcję iloczynu [1x3] z [3x1]. Wtedy wymiar „3” będzie się zgadzał wewnętrznie, a wynik będzie miał żądany wymiar [1x1]. Jak to zrobić bez przepisywania elementów wektora „b”? Za pomocą operacji transpozycji

```
>> a*b'
```

Wynikiem jest liczba 2. Możliwe są też inne ustawienia wzajemne tych wektorów. Sprawdźmy

```
>> b*a
```

Próba mnożenia wektorów [1x3] z [1x3] – ponownie uzyskamy błąd. Ale już np.

```
>> b*a'
```

ponownie obliczy iloczyn skalarny z racji tego, iż jest to działanie przemienne ($a \circ b = b \circ a$)

Inne możliwości to

```
>> b'*a
```

To z kolei nastąpiło rozmnożenie elementów do tablicy [3x3]. Jest to jednak zgodne z zasadami mnożenia, czyli [3x1] z [1x3] daje macierz [3x3]. Elementy wyniku to wszystkie możliwe iloczyny (każdy z każdym) elementów czynników mnożenia.

Z kolei

```
>> a'*b'
>> b'*a'
```

stanowią kolejne przykłady operacji niezgodnych z zasadami mnożenia. Wniosek jest następujący: w zależności od wzajemnego ustawienia dwóch wektorów można uzyskać coś mniejszego (skalar czyli liczba), coś większego (macierz kwadratowa) albo nic nie uzyskać, jeżeli działanie jest niewykonalne.

Mnożenie skalarne można również uzyskać poprzez zastosowanie funkcji o nazwie „dot”

```
>> dot(a,b)
>> dot(b,a)
```

W tym przypadku nie trzeba się przejmować ustawieniem wektorów – funkcja “dot” policzy iloczyn skalarny niezależnie od ich ustawienia, byleby wektory miały ten sam rozmiar.

Jest jeszcze jeden typ wyniku, który można uzyskać w wyniku mnożenia dwóch wektorów: można tak pomnożyć dwa wektory (o tej samej liczbie elementów wynoszącej 3), aby uzyskać w rezultacie kolejny, trzeci wektor. Taki rodzaj działania nazywa się iloczynem wektorowym. Jeżeli należałoby obliczyć iloczyn

wektorowy dwóch wektorów $[a_1 \quad a_2 \quad a_3]$ oraz $[b_1 \quad b_2 \quad b_3]$ ręcznie, można skorzystać z następującej formuły wyznacznikowej

$$\begin{bmatrix} \hat{i} & \hat{j} & \hat{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{bmatrix} = \hat{i}(a_2 b_3 - b_2 a_3) + \hat{j}(b_1 a_3 - a_1 b_3) + \hat{k}(a_1 b_2 - b_1 a_2)$$

W powyższym zapisie $\hat{i}$, $\hat{j}$, $\hat{k}$ oznaczają wersory (wektory jednostkowe) trzech osi (x, y, z) przestrzennego układu współrzędnych. Natomiast w Matlabie iloczyn wektorowy można zrealizować za pomocą funkcji o nazwie „cross”

```
>> cross(a,b)
>> cross(b,a)
```

Mnożenie wektorowe nie jest przemienne, można natomiast zapisać następującą własność $a \otimes b = -b \otimes a$.
Aby lepiej zrozumieć i zinterpretować wynik działania mnożenia wektorowego, stwórzmy dwa wektory współpłaszczyznowe (leżące na płaszczyźnie x,y)

```
>> a = [1 2 0]
>> b = [-4 5 0]
```

Ich iloczyn wektorowy

```
>> c = cross(a,b)
```

jest wektorem mającym jedyną niezerową składową w kierunku osi “z” – czyli jest wektorem prostopadłym do płaszczyzny, w której leżą wektory a i b. Można to dodatkowo sprawdzić, obliczając iloczyny skalarne wektora c z wektorami a oraz b

```
>> dot(a,c)
>> dot(b,c)
```

Zerowy iloczyn skalarny a i c oraz b i c oznacza prostopadłość tych wektorów (jest to wartość cosinusa kąta prostego $\pi/2$). Z kolei zwrot wektora c jest zależny od kolejności mnożenia wektorowego a i b.

Operacja transpozycji może być także użyta przy definiowaniu wektora, np. chcemy zdefiniować wektor kolumnowy („stojący”), ale zapomnieliśmy o wstawianiu średnika pomiędzy wierszami. Nie trzeba w takiej sytuacji wracać na początek linii i zamieniać wszystkie spacje (bądź przecinki) na średniki, wystarczy finalny wierszowy wektor „leżący” transponować

```
>> c = [1 4 5 6 –1 –9 10]'
```

Ta sama uwaga dotyczy generowania wektora poprzez operator zakresu (":"). Domyślnie powstaje wektor wierszowy

```
>> c = 1:10
```

natomiast, aby uczynić wynik wektorem kolumnowym, wystarczy napisać

```
>> c = (1:10)'
```

Użycie nawiasów powyżej jest niezbędne, gdyż w innym przypadku transpozycja będzie dotyczyła wartości końcowej "10", a nie całego wektora generowanego przez zakres.

- kolejnym omawianym działaniem będzie potęgowanie macierzy, w pierwszej kolejności w potędze naturalnej. Po wpisaniu poleceń

```
>> A^2
>> D^3
```

pojawią się wyniki, ale nie oznacza ją one prostych zbiorów potęg elementów macierzy A i D. A^2 oznacza bowiem to samo, co A*A, a D^3 to D*D*D.

```
>> A*A
>> D*D*D
```

Potęgowanie sprowadza się zatem do pomnożenia tej samej macierzy przez nią samą określoną liczbę razy: $\underset{(n\times n)}{A^p} = \underbrace{A\cdot A\cdot ...\cdot A}_{p}$. Stąd następujący wniosek: aby można było podnieść macierz do potęgi (każdej, nie tylko naturalnej), macierz musi być kwadratowa. Próba podniesienia do dowolnej potęgi macierzy prostokątnej zakończy się błędem

```
>> C^3
```

Podnoszenie macierzy kwadratowych do potęg innych niż naturalne też jest możliwe, chociaż trudniej nadać wynikowi odpowiednią interpretację

```
>> A^0.5
>> B^(1/3)
>> D^(-1/4)
```

W pierwszym i drugim przypadku wynik ostateczny jest zbiorem liczb zespolonych - proszę zwrócić uwagę na zapis każdego z elementu macierzy wynikowych (w postaci x + i*y), składających się z części rzeczywistej (x) i urojonej (y). Podnoszenie do takich potęg wymaga operacji transformacji macierzy do postaci diagonalnej - czyli takiej, w której macierz ma elementy niezerowe jedynie na przekątnej głównej. W takim specjalnym przypadku potęgowanie macierzy sprowadza się do potęgowania wyrazów na przekątnej głównej. Dla przykładu stwórzmy macierz diagonalną E

```
>> E = diag([16 9 4])
```

oraz wykonajmy działania

```
>> E^0.5
```

```
>> E^(1/3)
>> E^(-1/4)
```

Sprowadzenie macierzy do takiej postaci jest możliwe dzięki rozwiązaniu tzw. problemu własnego macierzy, który oznacza wyznaczenie wartości własnych oraz wektorów własnych macierzy. Zagadnienie to będzie rozważane w dalszej części opracowania.
Specyficzną potęgą macierzy jest potęga o wykładniku "-1". Zanim jednak omówimy własności takiej macierzy, wprowadzone zostanie pojęcie wyznacznika macierzy.

- wyznacznikiem (determinantem) macierzy kwadratowej nazywamy liczbę rzeczywistą reprezentującą macierz

$$\det(A) = |A| \in \Re$$

Z pojęciem tym mogliśmy się już wcześniej spotkać przy okazji rozwiązywania układów równań 2x2 metodą wyznaczników. Dla macierzy kwadratowych 2x2 oraz 3x3 jest kilka prostych sposobów obliczania ich wyznaczników

$$\left\| \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \right\| = a_{11}a_{22} - a_{12}a_{21}$$

$$\left\| \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \right\| = \begin{matrix} a_{11}a_{22}a_{33} + a_{21}a_{32}a_{13} + a_{31}a_{12}a_{23} + \\ -a_{31}a_{22}a_{13} - a_{11}a_{32}a_{23} - a_{21}a_{12}a_{33} \end{matrix}$$

$$\begin{matrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{matrix}$$

Dla macierzy o większych rozmiarach (ale także dla dwóch powyższych) obowiązuje ogólny sposób zwany rozwinięciem Laplace'a wyznacznika względem dowolnej kolumny lub wiersza. Sprowadza się on do tego, iż zamiast obliczać wyznacznik rzędu n-tego, obliczamy n wyznaczników rzędu o jeden mniejszych (n-1). Wzór ogólny na rozwinięcie względem kolumny k-tej

$$\left\| \begin{bmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \dots & \dots & \dots & \dots \\ a_{n,1} & a_{n,2} & \dots & a_{n,n} \end{bmatrix} \right\| = \sum_{i=1}^{n} (-1)^{i+k} a_{i,k} M_{i,k} =$$

$$= (-1)^{1+k} a_{1,k} M_{1,k} + (-1)^{2+k} a_{2,k} M_{2,k} + \dots + (-1)^{n+k} a_{n,k} M_{n,k}$$

Symbol $M_{i,k}$ oznacza tzw. minor, czyli podwyznacznik powstały z wykreślenia z oryginalnego wyznacznika określonej liczby wierszy i kolumn, w tym przypadku

i-tego wiersza i k-tej kolumny. Często wyrażenie $(-1)^{i+k} M_{i,k}$ oznacza się $A_{i,k}$ i nazywa się dopełnieniem algebraicznym elementu $a_{i,k}$ macierzy.

Dla przykładu obliczmy wyznacznik macierzy, którą od razu wprowadzimy do Matlaba

```
>> E = [0 1 2 3; -1 0 1 2; 0 2 3 4; 2 -1 2 0]
```

W pierwszej kolejności obliczymy wyznacznik ręcznie, posługując się rozwinięciem Laplace'a wyznacznika względem pierwszej kolumny. Wybór numeru wiersza lub kolumny jest w zasadzie dowolny (końcowy wynik powinien być taki sam), ale warto jest wybierać takie kolumny (lub wiersze), w których jest jak najwięcej elementów zerowych - stąd wybór pierwszej kolumny w omawianym przypadku. Rozwinięcie wyznacznika wygląda następująco

$$|E| = (-1)^{1+1} \cdot 0 \cdot \begin{vmatrix} 0 & 1 & 2 \\ 2 & 3 & 4 \\ -1 & 2 & 0 \end{vmatrix} + (-1)^{2+1} \cdot (-1) \cdot \begin{vmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ -1 & 2 & 0 \end{vmatrix} +$$
$$(-1)^{3+1} \cdot 0 \cdot \begin{vmatrix} 1 & 2 & 3 \\ 0 & 1 & 2 \\ -1 & 2 & 0 \end{vmatrix} + (-1)^{4+1} \cdot 2 \cdot \begin{vmatrix} 1 & 2 & 3 \\ 0 & 1 & 2 \\ 2 & 3 & 4 \end{vmatrix} = \ldots$$

Składników pierwszego i trzeciego w powyższej sumie nie ma sensu dalej obliczać, bowiem każdy z nich da w rezultacie "0" (poprzez mnożenie przez zerowy wyraz macierzy z pierwszej kolumny). Pozostałe podwyznaczniki (w składniku drugim i czwartym) można obliczyć albo stosując rozwinięcia do podwyznaczników rzędu 2giego (poniżej przykład ogólny z rozwinięciem względem pierwszego wiersza)

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11} \begin{vmatrix} a_{12} & a_{23} \\ a_{22} & a_{33} \end{vmatrix} - a_{12} \begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} + a_{13} \begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix}$$

albo stosując podany wcześniej wzór.

$$\begin{vmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ -1 & 2 & 0 \end{vmatrix} = 1 \cdot (3 \cdot 0 - 2 \cdot 4) - 2(2 \cdot 0 + 4 \cdot 1) + 3(2 \cdot 2 + 3 \cdot 1) = -8 - 8 + 21 = 5$$

$$\begin{vmatrix} 1 & 2 & 3 \\ 0 & 1 & 2 \\ 2 & 3 & 4 \end{vmatrix} = 1(1 \cdot 4 - 3 \cdot 2) - (0 \cdot 4 - 2 \cdot 2) + (0 \cdot 3 - 2 \cdot 1) = -2 + 4 - 2 = 0$$

wracając do obliczeń wyznacznika macierzy E otrzymujemy końcową wartość

$$\ldots = 0 + (-1) \cdot (-1) \cdot 5 + 0 + (-1) \cdot 2 \cdot 0 = 5$$

Zweryfikujmy tę wartość w Matlabie wydając polecenie

```
>> det(E)
```

W ten sam sposób możemy policzyć wyznaczniki pozostałych wprowadzonych wcześniej macierzy kwadratowych

```
>> det(A)
>> det(B)
>> det(D)
```

Z kolei próba obliczenia wyznacznika macierzy prostokątnej C poskutkuje błędem

```
>> det(C)
```

Wartość "0" wyznacznika uzyskana dla macierzy D ma specyficzne znaczenie. Macierz o wyznaczniku zerowym nazywa się macierzą osobliwą (np. macierz D), natomiast macierz o wyznaczniku niezerowym - macierzą nieosobliwą (np. macierze A i B). Co takiego szczególnego jest w macierzy D, iż jej wyznacznik jest równy "0"? W końcu tylko jeden z jej elementów jest zerowy. Otóż proszę przyjrzeć się pierwszemu i trzeciemu wierszowi macierzy D

```
>> D
```

Wiersz trzeci ([2 4 6]) to wynik przemnożenia wiersza pierwszego ([1 2 3]) przez "2". Taka sytuacja nazywa się liniową zależnością wierszy lub ich proporcjonalnością i w przypadku macierzy oznacza natychmiastową osobliwość - wyznacznik takiej macierzy musi być równy zero. Oczywiście ta sama własność dotyczy kolumn - jeżeli znajdziemy w macierzy dwie kolumny, z których jedna stanowi wielokrotność drugiej, to wyznacznik takiej macierzy jest równy "0". Wyznaczniki mają o wiele więcej ciekawych własności, są one wynotowane poniżej, wraz z odpowiednim ćwiczeniem do wykonania w Matlabie

(i) wyznacznik macierzy diagonalnej jest iloczynem wyrazów głównej przekątnej

```
>> F = diag([1 -2 5 6])
>> det(F)
>> prod(diag(F))
```

Funkcja "prod" oblicza iloczyn (ang. product), w tym przypadku wyrazów na przekątnej głównej macierzy F. Obydwa wyniki, uzyskane za pomocą funkcji "det" oraz "prod", powinny być takie same.

(ii) wyznacznik macierzy dolno- lub górno-trójkątnej jest iloczynem wyrazów głównej przekątnej. Macierz ma postać dolno- lub górno-trójkątną, jeżeli odpowiednio nad lub pod przekątną główną są same zera.

```
>> L = [1 0 0; 2 3 0; -1 4 5]
>> U = [2 3 4; 0 -3 5; 0 0 1]
```

L to przykład macierzy dolno-trójkątnej, a U - macierzy górno-trójkątnej. Sprawdźmy własności ich wyznaczników

```
>> det(L)
>> prod(diag(L))
>> det(U)
>> prod(diag(U))
```

(iii) wyznacznik macierzy, w której jeden wiersz lub kolumna składa się z samych zer jest równy 0

```
>> F = diag([1 0 5 6])
>> det(F)
```

(iv) wyznacznik macierzy jest równy wyznacznikowi jej macierzy transponowanej

```
>> E
>> det(E)
>> E'
>> det(E')
```

(v) jeżeli macierz B powstaje z macierzy A przez pomnożenie wszystkich elementów jednego wiersza lub jednej kolumny przez liczbę c to $\det(B) = c\det(A)$. Aby przetestować tę własność, pomnóżmy drugi wiersz macierzy E przez "-3"

```
>> E
>> E(2,:)  = -3*E(2,:)
>> E
```

a następnie obliczmy wyznacznik nowej macierzy E

```
>> det(E)
```

Wynik ("-15") faktycznie stanowi iloczyn "-3" poprzez wartość "starego" wyznacznika, czyli "5". Z twierdzenia tego płynie prosty wniosek, że jeżeli całą macierz pomnożymy przez c, to wyznacznik nowej macierzy zmieni się c^n razy, (czyli $\det(B) = c^n \det(A)$)gdzie *n* stanowi wymiar macierzy.
Przywróćmy oryginalną macierz E

```
>> E = [0 1 2 3; -1 0 1 2; 0 2 3 4; 2 -1 2 0]
```

i obliczmy

```
>> det(E)
>> det(-3*E)
>> det(E)*(-3)^4
```

Dwa ostatnie wyniki powinny być, w myśl poprzedniego wniosku, identyczne.

(vi) jeżeli macierz powstała w wyniku iloczynu dwóch macierzy kwadratowych, to jej wyznacznik jest iloczynem macierzy - czynników.

```
>> A
>> det(A)
>> B
>> det(B)
>> A*B
>> det(A*B)
>> det(A)*det(B)
```

(vii) gdy macierz B powstaje z macierzy A przez zamianę miejscami 2 wierszy lub 2 kolumn to wyznacznik zmienia znak na przeciwny, czyli $\det(B) = -\det(A)$. Spróbujmy zamienić miejscami w oryginalnej macierzy E drugi wiersz z wierszem trzecim

```
>> E
>> det(E)
```

W tym celu należy np. wiersz drugi zapamiętać do jakiegoś pomocniczego wektora

```
>> c = E(2,:)
```

a następnie na jego miejsce wstawić wiersz trzeci

```
>> E(2,:) = E(3,:)
```

Na koniec w miejsce przestawionego miejsca trzeciego wstawiamy zawartość wektora c

```
>> E(3,:) = c
```

Zamiana wierszy z wykorzystaniem trzeciego wektora była konieczna, gdyż inaczej, bez jego wykorzystania, oryginalna zawartość wiersza drugiego nie zostałaby zapamiętana i w rezultacie otrzymalibyśmy macierz złożoną z dwóch "wierszy trzecich". Podobne zagadnienie występuje w codziennym życiu np. przy problemie zamiany zawartości dwóch szklanek wypełnionych w całości dwoma różnymi płynami. Jeżeli zawartości jednej ze szklanek nie przejemy chwilowo do trzeciej szklanki, to taka zamiana nie będzie możliwa.
Teraz obliczmy wyznacznik nowej macierzy

```
>> det(E)
```

Powinniśmy otrzymać wartość "-5", przeciwną do oryginalnej wartości "5". Ta sama własność dotyczy oczywiście kolumn. Na koniec przywróćmy oryginalną macierz E

```
>> E = [0 1 2 3; -1 0 1 2; 0 2 3 4; 2 -1 2 0]
```

(viii) wyznacznik nie zmienia wartości, gdy do wiersza (lub kolumny) dodamy odpowiednie elementy innego wiersza (lub kolumny) pomnożone przez dowolną stałą. Innymi słowy dodanie do wybranego wiersza (lub kolumny) dowolnej kombinacji liniowej innych wierszy (lub innych kolumn) nie zmienia wartości wyznacznika. Własność tę przetestujmy dodając do trzeciej kolumny oryginalnej macierzy E

```
>> E
>> det(E)
```

kombinację liniową typu "2*pierwsza kolumna -3*czwarta kolumna"

```
>> E(:,3) = E(:,3) + 2*E(:,1) -3*E(:,4)
```

Obliczmy wyznacznik nowej macierzy

```
>> det(E)
```

Powinien on nie zmienić swej wartości ("-5"). Przywróćmy oryginalną postać macierzy E

```
>> E = [0 1 2 3; -1 0 1 2; 0 2 3 4; 2 -1 2 0]
```

Własność ta jest bardzo przydatna przy obliczaniu ręcznym wyznacznika. Można bowiem tak dodatkowo zmodyfikować wybrany wiersz (lub kolumnę) macierzy, aby pojawiło się w nim (w niej) jeszcze więcej wyrazów zerowych - co znacznie ułatwi jego obliczanie. Dla przykładu wybranie do rozwinięcia wyznacznika macierzy E kolumny pierwszej (przykład obliczeniowy zaprezentowany jakiś czas temu) pozwala nam zredukować liczbę koniecznych do obliczenia podwyznaczników rzędu 3x3 z czterech na dwa z uwagi na dwa zerowe wyrazy pierwszej kolumny. Ale można postarać się tak przekształcić macierz, aby w kolumnie pierwszej pojawiło się jeszcze jedno zero. Np. dodajmy do czwartego wiersza macierzy E wiersz drugi pomnożony przez "2"

```
>> E(4,:) = E(4,:) + 2*E(2,:)
```

Macierz po tym przekształceniu powinna mieć postać

```
E =

     0     1     2     3
    -1     0     1     2
     0     2     3     4
     0    -1     4     4
```

oraz wyznacznik

```
>> det(E)
```

Natomiast w obliczeniach ręcznych przy rozwinięciu - tak jak poprzednio - względem pierwszej kolumny, liczba koniecznych obliczeń pośrednich zmniejsza się do jednego podwyznacznika

$$|E| = \begin{vmatrix} 0 & 1 & 2 & 3 \\ -1 & 0 & 1 & 2 \\ 0 & 2 & 3 & 4 \\ 0 & -1 & 4 & 4 \end{vmatrix} = 0 + (-1)^{2+1} \cdot (-1) \cdot \begin{vmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ -1 & 4 & 4 \end{vmatrix} + 0 + 0 =$$

$$= (12-16) - 2(8+4) + 3(8+3) = -4 - 24 + 33 = 5$$

Mimo tych udogodnień, obliczanie wyznacznika macierzy przy wykorzystaniu rozwinięcia w podwyznaczniki jest bardzo kosztowne (pamięć komputerowa przy lokowaniu nowych wyznaczników) i uciążliwe. Programy komputerowe (w tym Matlab) stosują nieco inny sposób obliczania wyznaczników - przekształcają macierz drogą eliminacji do postaci trójkątnej (dolno- lub górno-), lub diagonalnej, a następnie obliczają iloczyn elementów na przekątnej głównej tak przekształconej macierzy.

- odwracanie macierzy: specyficzna potęga o wykładniku "-1" oznacza uogólnienie odwrotności liczby (skalara) w rachunku macierzowym. Warunkiem koniecznym istnienia macierzy odwrotnej do macierzy kwadratowej jest jej niezerowy wyznacznik ($\underset{(n\times n)}{A} \rightarrow \underset{(n\times n)}{A}^{-1} \Leftrightarrow \det(A) \neq 0$). Ogólny algebraiczny wzór na obliczanie elementów macierzy odwrotnej wykorzystuje pojęcie wyznacznika oraz podwyznaczników, omawianych w poprzednim podpunkcie opracowania

$$a_{i,j}^{"-1"} = \frac{A_{j,i}}{\det(A)} = \frac{(-1)^{i+j} M_{j,i}}{\det(A)}$$

Schematyczny sposób wyznaczenia macierzy odwrotnej wymaga (i) obliczenia wyznacznika macierzy odwracanej (ii) wyznaczenia macierzy minorów wszystkich jej wyrazów (iii) wyznaczenia macierzy dopełnień algebraicznych wszystkich jej wyrazów i (iv) wyznaczenia tzw. macierzy dołączonej (transponowanej macierzy dopełnień algebraicznych) (v) podzielenia wyrazów końcowej macierzy przez wyznacznik. Dla przykładu obliczymy macierze odwrotne dla dwóch macierzy kwadratowych - 2x2 oraz 3x3.

(i) wyznacznik macierzy

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad , \quad \det(A) = 4 - 6 = -2 \neq 0$$

(ii) macierz minorów wyrazów macierzy A (obliczając np. minor wyrazu $a_{1,1} = 1$ zasłaniamy pierwszy wiersz oraz pierwszą kolumnę, a następnie obliczamy wyznacznik tego, co zostało, czyli $\det(4) = |4| = 4$ itd.)

$$A^M = \begin{bmatrix} 4 & 3 \\ 2 & 1 \end{bmatrix}$$

(iii) macierz dopełnień algebraicznych macierzy A (zmieniamy znaki elementów macierzy minorów A^M wtedy, gdy suma numeru wiersza i kolumny, w których leży dany wyraz, jest nieparzysta, czyli np. 1+1 = 2 - nie zmieniamy znaku wyrazu $a_{1,1}^M = 4$, 1+2 = 3 - zmieniamy znak wyrazu $a_{1,2}^M = 3$ itd.)

$$A^{DA} = \begin{bmatrix} 4 & -3 \\ -2 & 1 \end{bmatrix}$$

(iv) transponujemy macierz dopełnień (tzw. macierz dołączona A^D)

$$A^D = \left(A^{DA}\right)^T = \begin{bmatrix} 4 & -2 \\ -3 & 1 \end{bmatrix}$$

(v) po podzieleniu macierzy dołączonej przez wyznacznik otrzymamy ostatecznie macierz odwrotną do A

$$A^{-1} = \frac{1}{\det(A)} A^D = -\frac{1}{2}\begin{bmatrix} 4 & -2 \\ -3 & 1 \end{bmatrix} = \begin{bmatrix} -2 & 1 \\ 1.5 & -0.5 \end{bmatrix}$$

Dokonamy kontroli uzyskanych wyników w programie Matlab pisząc

```
>> A^-1
```

lub przy zastosowaniu funkcji "inv" (ang. inversion - odwrotność)

```
>> inv(A)
```

Iloczyn macierzy i macierzy do niej odwrotnej (o ile istnieje) powinien dawać w rezultacie macierz jednostkową (złożenie może odbywać się w dowolnej kolejności)

```
>> inv(A)*A
>> A*A^-1
```

Sprawdźmy powyższe także dla macierzy większej E - 4x4

```
>> inv(E)*E
>> E*inv(E)
```

W przypadku powyższego przykładu charakterystyczne jest to, iż wynik obydwu mnożeń nie stanowi idealnej macierzy jednostkowej (złożonej tylko z liczb "1" oraz "0"). Reprezentacja skończona liczb rzeczywistych oraz dużo operacji dzielenia, a zwłaszcza odejmowania (które drastycznie obniżają precyzję wyniku) wprowadza na tyle duży błąd obcięcia, iż Matlab przechowuje niezerowe miejsca po przecinku. Przekonajmy się o tym chwilowo włączając format "długi"

```
>> format long
>> E*inv(E)
```

```
>> format short
```

Obliczmy jeszcze ręcznie macierz odwrotną do macierzy 3x3

$$F=\begin{bmatrix}1 & 0 & -1\\ 0 & 1 & 0\\ -1 & 0 & -1\end{bmatrix}\quad,\quad \det(F)=-1-0-1=-2\neq 0$$

Macierz minorów wygląda następująco

$$F^{M}=\begin{bmatrix}\begin{vmatrix}1 & 0\\ 0 & -1\end{vmatrix} & \begin{vmatrix}0 & 0\\ -1 & -1\end{vmatrix} & \begin{vmatrix}0 & 1\\ -1 & 0\end{vmatrix}\\ \begin{vmatrix}0 & -1\\ 0 & -1\end{vmatrix} & \begin{vmatrix}1 & -1\\ -1 & -1\end{vmatrix} & \begin{vmatrix}1 & 0\\ -1 & 0\end{vmatrix}\\ \begin{vmatrix}0 & -1\\ 1 & 0\end{vmatrix} & \begin{vmatrix}1 & -1\\ 0 & 0\end{vmatrix} & \begin{vmatrix}1 & 0\\ 0 & 1\end{vmatrix}\end{bmatrix}=\begin{bmatrix}-1 & 0 & 1\\ 0 & -2 & 0\\ 1 & 0 & 1\end{bmatrix}$$

Macierz dopełnień algebraicznych (bez zmian, bo wyraz na "nieparzystych" miejscach są równe "0")

$$F^{DA}=\begin{bmatrix}-1 & 0 & 1\\ 0 & -2 & 0\\ 1 & 0 & 1\end{bmatrix}$$

Macierz dołączona F^{D} (F^{DA} transponowana - bez zmian, bo macierz dopełnień jest symetryczna)

$$F^{D}=\left(F^{DA}\right)^{T}=\begin{bmatrix}-1 & 0 & 1\\ 0 & -2 & 0\\ 1 & 0 & 1\end{bmatrix}$$

Macierz odwrotna

$$F^{-1}=-\frac{1}{2}\begin{bmatrix}-1 & 0 & 1\\ 0 & -2 & 0\\ 1 & 0 & 1\end{bmatrix}=\begin{bmatrix}0.5 & 0 & -0.5\\ 0 & 1 & 0\\ -0.5 & 0 & -0.5\end{bmatrix}$$

Proszę sprawdzić poprawność wyniku poprzez (i) dokonanie ręczne mnożeń FF^{-1} oraz $F^{-1}F$ (ii) obliczenie macierzy odwrotnej do F w Matlabie.
To nie przypadek, że odwrotność macierzy symetrycznej jest również macierzą symetryczną. Symetria jest bowiem zachowywana w wyniku odwracania macierzy. Z kolei macierze anty-symetryczne są zawsze macierzami osobliwymi i dlatego nie istnieją macierze odwrotne do nich.
Odwrotnością macierz diagonalnej stanowi macierz złożona z odwrotności elementów na przekątnej głównej

```
>> diag([1 2 3])
>> inv(diag([1 2 3]))
```

Wyznaczniki macierzy i macierz odwrotnej spełniają następującą relację $\underset{(n\times n)}{A}, \underset{(n\times n)}{A^{-1}} \rightarrow \det(A) = \frac{1}{\det(A^{-1})}$.

```
>> E
>> E^-1
>> det(E)
>> det(inv(E))
>> 1/ det(inv(E))
```

Próba obliczenia odwrotności macierzy prostokątnej zakończy się błędem

```
>> inv(C)
```

Z kolei próba obliczenia odwrotności macierzy kwadratowej, ale osobliwej da wynik

```
>> inv(D)
```

złożony z samych symboli nieskończoności ("inf") oraz ostrzeżenia (nie stanowiącego błędu krytycznego przerywającego obliczenia): "Matrix is singular to working precision" co oznacza "Macierz jest osobliwa dla precyzji roboczej". Należy to rozumieć w taki sposób, iż wyznacznik macierzy zerowy lub na tyle mały (ale różny od idealnego zera), iż próba obliczenia jego odwrotności dała w rezultacie "inf" (tak samo, jak przy odwracaniu zera lub liczby bardzo bliskiej zeru). Macierz wynikowa złożona z "inf" nie ma żadnego sensu liczbowego, a jedynie posiada pewien charakter informacyjny, iż coś nie w porządku jest z macierzą odwracaną. Macierze bardzo bliskie osobliwym nazywają się macierzami źle uwarunkowanymi, a zagadnienia inżynierskie, w których występują takie macierze, stanowią jeden z najważniejszych problemów współczesnej numeryki.
Macierze odwrotne są wykorzystywane m.in. do rozwiązywania układów równań, o czym będzie mowa w kolejnych rozdziałach opracowania.

- ostatnim omawianym działaniem macierzowym będzie dzielenie macierzy przez macierz. Taki rodzaj działania nie jest zdefiniowany w algebrze. W Matlabie podzielić macierz przez macierz oznacza pomnożyć macierz przez macierz odwrotną, a kolejność takiego mnożenia zależy od skierowania znaku dzielenia ("/" - dzielenie prawostronne, "\" - dzielenie lewostronne). I tak:

$$\underset{(n\times m)}{A} / \underset{(m\times m)}{B} = \underset{(n\times m)}{A} \cdot \underset{(m\times m)}{B}^{-1} = \underset{(n\times m)}{C}$$

oraz

$$\underset{(n\times n)}{A} \setminus \underset{(n\times m)}{B} = \underset{(n\times n)}{A^{-1}} \cdot \underset{(n\times m)}{B} = \underset{(n\times m)}{C}$$

Kolejności mnożenia macierzy w powyższych wzorach nie można zmieniać, gdyż jak wiadomo, mnożenie macierzy nie jest przemienne. Wykonajmy kilka obliczeń na istniejących macierzach A i B

```
>> A
>> B
>> A / B
```

powinno dać to samo, co

```
>> A*B^-1
```

Z kolei

```
>> A \ B
```

stanowi inny sposób zapisu

```
>> A^-1*B
```

11. Działania tablicowe

W Matlabie, oprócz omawianych szczegółowo w poprzednim rozdziale operacji macierzowych (dodawanie, odejmowanie, mnożenie przez skalar, mnożenie przez macierz, potęgowanie, transpozycja, odwracanie, dzielenie) istnieje grupa operacji zwanych tablicowymi. Różnią się one od macierzowych tym, że prowadzone są one na poziomie elementów macierzy, mimo iż zapisuje się "globalnie", z użyciem symboli macierzowych, bez specyfikacji numerów elementów. Wywołuje się je wstawiając kropkę (".") pomiędzy pierwszy argument takiego działania, a symbol działania. Zacznijmy od mnożenia tablicowego. Przywróćmy definicję macierzy A i B

```
>> A = [1 2; 3 4]
>> B = [3 2; -2 3]
```

Napiszmy (uwaga na kropkę pomiędzy "A" i "*")

```
>> A.*B
```

Wynik jest zupełnie inny niż w przypadku operacji macierzowej bez kropki

```
>> A*B
```

Przypomnijmy: mnożenie macierzowe (bez kropki) oznaczało pomnożenie macierzy przez macierz w sposób następujący: wiersze macierzy A są mnożone skalarnie przez kolumny macierzy B. Natomiast w operacji tablicowej (z kropką) mnożone są przez siebie odpowiadające sobie elementy macierzy A i B, czyli 1*3, 2*2, 3*(-2) oraz 4*3. Stąd oczywisty wniosek, iż do tego rodzaju operacji tablicowych, w których argumentami są macierze, macierze te muszą mieć dokładnie takie same rozmiary (aby możliwe było ułożenie wszystkich par elementów pomiędzy macierzami). Jest to warunek silniejszy, dlatego też możliwe do pomnożenia macierzowego, macierze C i A

```
>> C*A
```

nie mnożą się tablicowo - pojawi się komunikat o błędzie

```
>> C.*A
```

Co więcej, mnożenie tablicowe (jeżeli jest wykonywalne) jest przemienne - jak zwykłe mnożenie liczb

```
>> A.*B
>> B.*A
```

Uwaga dla osób mających doświadczenie w programowaniu, zwłaszcza w stosowaniu pętli "for": mnożenie tablicowe macierzy (jak i wszystkie działania tego typu) zastępują w Matlabie zapis podwójnie zagnieżdżonej pętli "for": A(i,j) * B(i,j) na poziomie elementów z pominięciem potrzeby definiowania zakresu parametrów sterujących pętli "i" oraz "j".

Podobnie działa dzielenie tablicowe - mimo zapisu na poziomie macierzy dzielenie dotyczy odpowiadających sobie wyrazów. Tutaj także możemy wyróżnić mnożenie prawo- i lewostronne. W przypadku prawostronnego wyrazy pierwszej macierzy są dzielone przez wyrazy macierzy drugiej, a w przypadku lewostronnego - odwrotnie.

```
>> A./B
>> A.\B
```

Dzielenie tablicowe nie jest przemienne, chyba że zmienimy "skośność" kreski dzielenia na przeciwną

```
>> A./B
>> B.\A
```

Skoro istnieją tablicowe wersje mnożenia i dzielenia, to zapewne istnieją także tablicowe dodawanie i odejmowanie. Sprawdźmy

```
>> A.+B
>> A.-B
```

I tutaj następuje zaskoczenie: dlaczego wystąpił komunikat o błędzie składniowym? Przecież macierze mają takie same rozmiary… Wyjaśnienie jest następujące: dokładnie to samo robią zwykłe operacje macierzowego dodawania i odejmowania (dodają lub odejmują odpowiadające sobie elementy macierzy), dlatego też nie było potrzeby wprowadzać dodatkowego zapisu tych podstawowych działań

```
>> A+B
>> A-B
```

Ale za to istnieją operacje tablicowe potęgowania. Np. można podnieść tablicowo macierz do potęgi macierzowej

```
>> A.^B
>> B.^A
```

W obydwu przypadkach podnoszone są oczywiście wyrazy pierwszej macierzy do potęg określonych przez wyrazy drugiej macierzy. Jeżeli natomiast chcemy podnieść wszystkie elementy macierzy (tu już może być macierzą prostokątną) do określonej potęgi, wystarczy napisać

```
>> A.^2
>> C.^3
>> D.^(-1)
```

W rezultacie otrzymujemy odpowiednio kwadraty wyrazów macierzy A, sześciany wyrazów macierzy C oraz odwrotności wyrazów macierzy D.

Poniżej zamieszczono listę wszystkich omówionych operacji tablicowych wraz z ich interpretacją na poziomie elementów

(i) mnożenie macierzy przez macierz $\underset{(n\times m)}{A} .* \underset{(n\times m)}{B} = \underset{(n\times m)}{C} \quad \left[c_{ij} = a_{ij} \cdot b_{ij}\right]$

(ii) dzielenie (prawostronne) macierzy przez macierz $\underset{(n\times m)}{A} ./ \underset{(n\times m)}{B} = \underset{(n\times m)}{C} \quad \left[c_{ij} = a_{ij} / b_{ij}\right]$

(iii) dzielenie (lewostronne) macierzy przez macierz $\underset{(n\times m)}{A} .\backslash \underset{(n\times m)}{B} = \underset{(n\times m)}{C} \quad \left[c_{ij} = b_{ij} / a_{ij}\right]$

(iv) potęgowanie macierzy w liczbie $\underset{(n\times m)}{A}^{p} = \underset{(n\times m)}{C} \quad \left[c_{ij} = a_{ij}{}^{p}\right]$

(v) potęgowanie macierzy w macierzy $\underset{(n\times m)}{A}^{\underset{(n\times m)}{B}} = \underset{(n\times m)}{C} \quad \left[c_{ij} = a_{ij}^{\,b_{ij}}\right]$

Mnożenie tablicowe ma podstawowe znaczenie przy konstruowaniu np. zbioru wartości funkcji dla wektora argumentów (zmiennych niezależnych). Np. mając wzór funkcji

$$y = \frac{x^2 + 1}{x^{\frac{5}{4}} - \frac{5}{x+1}}$$

i chcąc obliczyć jej wartość dla jednej określonej wartości "x" wystarczy napisać

```
>> x = 5
>> y = (x^2 + 1) / (x^(5/4) - 5/(x+1))
```

Jednakże, jeżeli "x" będzie wektorem - zbiorem zmiennych niezależnych, dla których będziemy chcieli obliczyć "za jednym zamachem" zbiór wartości tej samej funkcji, to bezpośrednie przeniesienie wzoru poskutkuje błędem

```
>> x = (10:20)'
>> y = (x^2 + 1) / (x^(5/4) - 5/(x+1))
```

Skąd się ten błąd bierze i jaka "macierz musi być kwadratowa" (wg komunikatu Matlaba)? Otóż popatrzmy na strukturę powyższego wzoru "y = …". Wszystkie potęgi, a także dzielenia będą traktowane przez Matlab jako operacje macierzowe, czyli np. x^2 będzie oznaczało próbę pomnożenia x*x, co jak wiemy, jest zarezerwowane dla macierzy kwadratowych. A "x" jest wektorem kolumnowym 11x1. Jak należy więc to działanie zapisać? Pamiętajmy, iż w rezultacie mamy otrzymać zbiór wartości funkcji dla każdego argumentu (elementu wektora "x") osobno. To oznacza, że wszystkie operacje muszą być

prowadzone na poziomie elementu (czyli element do potęgi oraz element przez element). Dlatego też należy wszystkie działania macierzowe zastąpić tablicowymi

```
>> y = (x.^2 + 1). / (x.^(5/4) - 5./(x+1))
```

W sumie przybyły cztery kropki - przed każdym dzieleniem macierzowym i potęgowaniem macierzowym. Wstawianie kropki w działanie "5/4" nie ma sensu, bo wykonywane jest ono na liczbach. Również działań dodawania i odejmowania nie trzeba modyfikować w żaden sposób, gdyż jak pamiętamy, dodanie liczby do macierzy to dodanie tej liczby do każdego jej elementu (podobnie z odejmowaniem).

12. Wybrane funkcje i zagadnienia macierzowe

W pierwszej kolejności omówione zostaną układy równań liniowych w ogólnej postaci

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases}$$

o takiej samej liczbie równań i niewiadomych (układy kwadratowe, układy określone). Powyżej pokazano tradycyjny "szkolny" zapis polegający na zapisaniu pod znakiem klamry równania po równaniu składających się z wyrazów liniowych ze względu na wszystkie niewiadome $x_1, x_2, ..., x_n$, ich znanych mnożników (współczynniki układu równań) $a_{1,1}, a_{1,2}, ..., a_{1,n}, a_{2,1}, ..., a_{n,n}$ oraz znanych wyrazów wolnych $b_1, b_2, ..., b_n$. Istnieją trzy podstawowe ("szkolne") metody rozwiązywania małych układów równań (dla n=2 lub n=3): metoda podstawiania, metoda przeciwnych współczynników oraz metoda wyznaczników. Wszystkie te metody są dosyć uciążliwie w obliczeniach ręcznych, zwłaszcza, jeżeli rozmiar układu równań jest znaczny (n>3). Dlatego też istnieje dużo algorytmów macierzowych (dokładnych i przybliżonych), stanowiących pewne uogólnienia wspomnianych metod, dla układów o dużych rozmiarach. Wymagają one innego zapisu układu równań w formie macierzowej, mianowicie

$$\underset{[n\times n]}{A}\ \underset{[n\times 1]}{x} = \underset{[n\times 1]}{b}$$

w którym to $\underset{[n\times n]}{A} = \begin{bmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \dots & \dots & \dots & \dots \\ a_{n,1} & a_{n,2} & \dots & a_{n,n} \end{bmatrix}$ to kwadratowa macierz współczynników (macierz główna układu), $\underset{[n\times 1]}{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ - wektor niewiadomych, oraz $\underset{[n\times 1]}{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$ - wektor wyrazów wolnych (tzw. wektor prawej strony). Z podanej wyżej definicji kwadratowego układu równań wynika wzór na jego rozwiązanie

$$\underset{[n\times 1]}{x} = \underset{[n\times n]}{A^{-1}}\,\underset{[n\times 1]}{b}$$

Wymaga on obliczenia macierzy odwrotnej (o ile macierz A nie jest osobliwa - musi mieć wyznacznik niezerowy) oraz jej pomnożenia przez wektor wyrazów wolnych b. Jest to jednak uciążliwy sposób wyznaczania rozwiązania w sposób ręczny, gdyż obliczanie ręczne macierzy odwrotnej jest złożone obliczeniowo (wymaga liczenia wielu podwyznaczników). Jednakże właśnie ten sposób jest najczęściej stosowany w Matlabie to rozwiązywania układów równań. Np. rozważmy układ równań o tradycyjnym zapisie

$$\begin{cases} 2x_1 + 3x_2 = 8 \\ x_1 + x_2 + x_3 = 6 \\ -3x_2 + x_3 = -3 \end{cases}$$

Na tej podstawie należy skonstruować w Matlabie znane jego elementy, czyli macierz współczynników (np. o nazwie "A") oraz wektor prawej strony (np. o nazwie "b"). Wyrazy macierzy "A" wprowadzamy wierszami wzdłuż każdego z równań, a przy definiowaniu wektora "b" pamiętajmy o tym, iż musi on być wektorem kolumnowym ("stojącym") - w innym przypadku nie będą zgadzały się wymiary w przyszłym mnożeniu macierzowym

```
>> A = [2 3 0; 1 1 1; 0 -3 1]
>> b = [8; 6; -3]
```

Aby rozwiązać ten układ, a rozwiązanie zapamiętać w wektorze "x", należy napisać

```
>> x = A^-1*b
```

lub

```
>> x = inv(A)*b
```

Współczynniki układu zostały tak dobrane, aby rozwiązaniem ścisłym były idealne liczby 1,2 i 3. Jednakże poprzez błędy zaokrągleń podczas odejmowania i dzielenia nie udało się takich otrzymać - wprawdzie przy obecnym formacie nie widać niezerowych pozycji po przecinku, ale prawdopodobnie gdzieś jest coś niezerowego, skoro Matlab wypisuje zera po przecinku. Sprawdzenie poprawności rozwiązania układu polega na pomnożeniu macierzy "A" przez wektor rozwiązań "x" - jeżeli układ został rozwiązany poprawnie, rezultatem takiego mnożenia powinien być oryginalny wektor "b" (lub zestaw liczb bardzo mu bliski)

```
>> A*x
>> b
```

Dlaczego używamy w Matlabie tego właśnie zapisu, skoro jest on tak kosztowny obliczeniowo? Matlab nie oblicza macierzy odwrotnej "z definicji" - tak, jak to podano w jednym z poprzednich rozdziałów. Do obliczania macierzy odwrotnej stosowana jest odpowiednia metoda eliminacji np. do postaci diagonalnej.
Rozwiązanie układu równań w Matlabie można zapisać w jeszcze inny sposób, mianowicie z wykorzystaniem dzielenia lewostronnego

```
>> x = A\b
```

O zaletach tego najbardziej ogólnego w Matlabie zapisu rozwiązania układu równań porozmawiamy bardziej szczegółowo przy okazji omawiania prostokątnych układów równań.
Wracając jeszcze do wzoru na rozwiązanie układu równań wynikającego z jego definicji - obliczenia ręczne wymagają następujących przekształceń: zapisania macierzy A, wektora b, macierzy odwrotnej A^{-1} oraz pomnożenia A^{-1} przez b

$$A = \begin{bmatrix} 2 & 3 & 0 \\ 1 & 1 & 1 \\ 0 & -3 & 1 \end{bmatrix} \quad , \quad b = \begin{bmatrix} 8 \\ 6 \\ -3 \end{bmatrix}$$

$$\det(A) = 5 \quad \rightarrow \quad A^{-1} = \begin{bmatrix} 0.8 & -0.6 & 0.6 \\ -0.2 & 0.4 & -0.4 \\ -0.6 & 1.2 & -0.2 \end{bmatrix}$$

$$x = A^{-1}b = \begin{bmatrix} 0.8 & -0.6 & 0.6 \\ -0.2 & 0.4 & -0.4 \\ -0.6 & 1.2 & -0.2 \end{bmatrix} \cdot \begin{bmatrix} 8 \\ 6 \\ -3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

Ogólnym algebraicznym sposobem rozwiązania tak sformułowanych układów równań jest tzw. metoda Cramera, stanowiąca uogólnienie metody wyznaczników. W metodzie tej każda niewiadoma opisana jest wzorem

$$x_i = \frac{\det(A_i)}{\det(A)} \quad , \quad i = 1, 2, ..., n$$

det(A) to wyznacznik macierzy głównej (musi być nieosobliwa, by istniało jedno rozwiązanie), a $\det(A_i)$ to wyznacznik macierzy powstałej z macierzy głównej A po zastąpieniu jej i-tej kolumny wektorem wyrazów wolnych b. Np. pierwsza niewiadoma wcześniej analizowanego układu równań wg wzorów Cramera obliczana jest następująco

$$x_1 = \frac{\det(A_1)}{\det(A)} = \frac{\begin{vmatrix} 8 & 3 & 0 \\ 6 & 1 & 1 \\ -3 & -3 & 1 \end{vmatrix}}{5} = \frac{5}{5} = 1$$

W Matlabie nie ma osobnej funkcji rozwiązującej układy równań metodą Cramera (jest to sposób niewiele lepszy, jeżeli chodzi o czas obliczeń i liczbę operacji, w porównaniu do metody wynikającej wprost z definicji z uwagi na obecność wyznaczników). Ale można wzory na wszystkie niewiadome wpisać ręcznie, z wykorzystaniem funkcji "det". Najpierw stworzymy trzy macierze A_1, A_2 oraz A_3, różniące się od oryginalnej macierzy A w taki sposób, iż odpowiednio 1, 2 i 3 kolumna będą zastąpione wektorem wyrazów wolnych

```
>> A1 = A;
>> A1(:,1) = b
>> A2 = A;
>> A2(:,2) = b
>> A3 = A;
>> A3(:,3) = b
```

W następnej kolejności wpisujemy niewiele już różniące się wzory na kolejne trzy niewiadome

```
>> x1 = det(A1)/det(A)
>> x2 = det(A2)/det(A)
>> x3 = det(A3)/det(A)
```

które następnie można zebrać do jednego wektora rozwiązań "x"

```
>> x = [x1; x2; x3]
```

Jeżeli wyznacznik macierzy głównej układu będzie równy 0, żadna z metod nie znajdzie rozwiązania. Z matematyki wiemy, że układ taki może mieć nieskończenie wiele rozwiązań (układ nieokreślony) lub może nie mieć w ogóle rozwiązania (układ sprzeczny). O tym jednak decyduje już postać wektora prawej strony b. Np. przywróćmy lub stwórzmy w Matlabie osobliwą macierz D

```
>> D = [1 2 3; -1 0 1; 2 4 6]
>> det(D)
```

Każda z prób rozwiązania układu równań z macierzą D i wektorem b skończy się niepowodzeniem

```
>> D^-1*b
>> inv(D)*b
>> D\b
```

W algebrze istnieją także układy równań o różnej liczbie niewiadomych (*n*) i równań (*m*)

$$\begin{cases} a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,n}x_n = b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,n}x_n = b_2 \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ a_{m,1}x_1 + a_{m,2}x_2 + \cdots + a_{m,n}x_n = b_m \end{cases}$$

- tzw. układy niedookreślone (lub pod-określone - gdy równań *m* jest mniej niż niewiadomych *n*) i nad-określone (gdy równań *m* jest więcej niż niewiadomych *n*). Ogólnym sposobem analizy (nie rozwiązania, lecz przekonania się, czy taki układ posiada rozwiązanie, a jeżeli tak, to czy jest ono jednoznaczne) jest wykorzystanie twierdzenia Kroneckera-Capellego, które wiąże się z pojęciem tzw. rzędu macierzy. Rząd macierzy jest to wymiar największego niezerowego podwyznacznika tej macierzy. Rozumienie tego pojęcia zacznijmy od macierzy kwadratowych. Jeżeli wyznacznik macierzy kwadratowej o rozmiarze *n* jest niezerowy, to rząd takiej macierzy wynosi *n*. Jeżeli jest zerowy, to należy sprawdzić, czy któryś z podwyznaczników tej macierzy o wymiarze $(n-1)$ jest

niezerowy - jeżeli chociaż jeden okaże się być różny od zera, to rząd macierzy wynosi $(n-1)$. Jeżeli wszystkie będą zerowe, należy sprawdzać podwyznaczniki rzędu $(n-2)$ aż do czasu, kiedy dotrzemy do wyznaczników z pojedynczych liczb. Poniżej kilka przykładów obliczeń w Matlabie rzędu macierzy kwadratowych z wykorzystaniem funkcji "rank"

```
>> rank(A)
>> rank(D)
>> rank([4 0 0; 0 0 0; 0 0 0])
>> rank(zeros(50,50))
```

Macierz A o rozmiarze 3 jest nieosobliwa (wyznacznik rzędu 3 różny od zera), więc jej rząd wynosi 3. Macierz D jest osobliwa, więc rząd nie może wynosić 3, ale posiada ona podwyznaczniki rzędu 2 - niezerowe, np. poniższy powstały po skreśleniu pierwszej kolumny i trzeciego wiersza

$$D_{3,1} = \begin{vmatrix} 2 & 3 \\ 0 & 1 \end{vmatrix} = 2 \neq 0$$

Dlatego rząd D wynosi 2 (nie dlatego, że podwyznacznik jest równy "2", tylko dlatego, że jego rząd wynosi "2" !).
Rząd trzeciej macierzy o wymiarze 3, składającej się z samych zer, i jednej czwórki wynosi 1, bo tylko jeden podwyznacznik rzędu 1 jest niezerowy (chodzi właśnie o tę czwórkę i podwyznacznik z niej równy 4). Rząd czwartej macierzy jest równy zero, bo mimo dużego rozmiaru (50), składa się ona z samych zer i nawet podwyznacznik rzędu 1-szego jest zerowy.
Z kolei rzędy macierzy prostokątnych wymagają wycinania z nich podwyznaczników w wyniku skreślania różnej liczby wierszy i kolumn. Przywróćmy macierz C o rozmiarach 4x2

```
>> C = [3 4; 5 6; -1 2; 8 9]
```

Macierz jest niekwadratowa, więc nie można obliczyć z niej wyznacznika, ale można wycinać z niej podwyznaczniki rzędów 3, np. poniższy powstały z wykreślenia trzeciego i czwartego wiersza

$$C_{3..4,0} = \begin{vmatrix} 3 & 4 \\ 5 & 6 \end{vmatrix} = 18 - 20 = -2 \neq 0$$

Jest on niezerowy, dlatego też rząd macierzy powinien być równy 2. Sprawdźmy

```
>> rank(C)
```

Twierdzenie Kroneckera-Capellego mówi rzecz następującą. Niech: $r_A = rz(A)$ - rząd macierzy głównej układu oraz $r_R = rz(Ab)$ - rząd macierzy rozszerzonej (macierzy głównej A z dołączoną po prawej stronie kolumną wyrazów wolnych). Jeżeli układ ma rozwiązanie (wtedy i tylko wtedy gdy $r_A = r_R$), to o liczbie jego rozwiązań można wnioskować według następujących reguł:

(i) dla $r_A = r_R = n$ (n - liczba niewiadomych) układ ma dokładnie jedno rozwiązanie, przy czym dla układu jednorodnego (wektor b jest wektorem zerowym) tym rozwiązaniem jest $x_1 = x_2 = ... = x_n = 0$,
(ii) dla $r_A = r_R < n$ zbiór rozwiązań zależy $n - r$ parametrów i tworzy rozmaitość liniową wymiaru $n - r$. Jeśli przy tym układ jest jednorodny, to zbiór tych rozwiązań tworzy $n - r$ wymiarową podprzestrzeń przestrzeni n - wymiarowej,
(iii) przypadek $r = s > n$ jest niemożliwy.

Cóż to wszystko oznacza? Przede wszystkim o istnieniu jednoznacznego rozwiązania układu prostokątnego $(n \neq m)$ decyduje równość dwóch rzędów macierzy: głównej i rozszerzonej głównej o wyrazy wolne. Wyjaśnimy to na przykładzie

$$\begin{cases} x_1 + 2x_2 = 5 \\ -3x_1 + 4x_2 = 5 \\ -2x_1 - 4x_2 = -10 \end{cases}$$

Aby dokonać analizy tego układu, wprowadźmy w Matlabie macierz A i wektor b

```
>> A = [1 2; -3 4; -2 -4]
>> b = [5;5;-10]
```

Następnie obliczmy rzędy macierzy A i macierz rozszerzonej Ab

```
>> rank(A)
>> rank([A b])
```

Obydwa rzędy są równe 2, to oznacza, iż rozwiązanie istnieje, a dalej, wg powyższego twierdzenia interesuje nas przypadek (i): jest tylko jedno rozwiązanie i nie jest to rozwiązanie zerowe, gdyż wektor prawej strony nie jest zerowy (układ nie jest jednorodny). Dlatego też mimo pozornej nad-określoności (trzy równania, dwie niewiadome) układ ma jedno rozwiązanie. I rzeczywiście - równanie trzecie to równanie pierwsze przemnożone przez "-2". Jest to więc pozornie nad-określony układ równań, gdyż - w interpretacji graficznej - prosta pierwsza i trzecia pokrywałyby się ze sobą. Jak zatem to rozwiązanie znaleźć? Najłatwiej jest oczywiście jedno z równań (pierwsze lub trzecie) odrzucić, ale oczywiście trzeba wiedzieć, które z nich są zależne. Twierdzenie o tym nie mówi, a metoda "na oko" jest dobra w przypadku małych układów równań. Spróbujmy więc zastosować wzór wynikający z definicji

```
>> x = A^-1*b
>> x = inv(A)*b
```

Obydwa powyższe zapisy zawodzą, gdyż nie da się odwrócić macierzy niekwadratowej. Wykorzystajmy więc trzeci zapis, który również był wcześniej stosowany

```
>> x = A\b
```

Udało się! Stąd jasny wniosek, iż niewinny z pozoru znak lewostronnego dzielenia ('\') robi nieco więcej, niż tylko mnoży odwrotność A przez b. I faktycznie: w przypadku układów prostokątnych układ taki przechodzi proces "ukwadratowienia", odpowiedniej

specjalistycznej procedury, która sprowadza układ prostokątny do kwadratowego o wymiarach $n \times n$. Do tego działanie dobiera odpowiednią optymalną metodę eliminacji do dalszego rozwiązania tego układu, wynikającą z postaci macierzy współczynników (symetrycznej, pasmowej, rzadkiej, diagonalnej, trójkątnej itp.).
Sprawdźmy, czy otrzymane rozwiązanie jest rozwiązaniem właściwym dla wyjściowego układu równań

```
>> A*x
>> b
```

Równość powyższych wyrażeń potwierdza przypuszczenia (na podstawie równości rzędów) o pozornej nad-określoności tego układu.
Rozważmy podobny przykład

$$\begin{cases} x_1 + 2x_2 = 5 \\ -3x_1 + 4x_2 = 5 \\ -2x_1 - 4x_2 = -12 \end{cases}$$

ale różniący się ostatnim wyrazem wolnym. Skonstruujmy nowy wektor b

```
>> b = [5;5;-12]
```

i dokonajmy analizy rzędów dwóch macierzy

```
>> rank(A)
>> rank([A b])
```

Rząd macierzy A (jak i sama macierz) nie zmienił się i wynosi 2. Natomiast rząd macierzy rozszerzonej wynosi teraz 3 (co oznacza, iż wyznacznik macierzy jest niezerowy). W myśl twierdzenia nie ma rozwiązania - układ jest sprzeczny. Sprawdźmy zatem, co obliczy Matlab

```
>> x = A\b
```

Cóż to są za liczby? Czyżby program działał wbrew udowodnionemu twierdzeniu algebraicznemu? Może najpierw sprawdźmy poprawność rozwiązania wstawiając je do układu równań i porównując z wektorem b

```
>> A*x
>> b
```

Czyli nie jest to właściwe rozwiązanie, skoro otrzymaliśmy nieco inny wektor. Graficznie oznaczałoby to, że punkt o współrzędnych zawartych w wektorze "x" leży idealnie na prostej drugiej, ale nie leży na pierwszej i trzeciej. To, co obliczył Matlab, nazywa się pseudo-rozwiązaniem i ma charakter kompromisu dla trzech sprzecznych sobie warunków - stanowi współrzędne punktu leżącego możliwe blisko każdej z prostych, nie wyróżniając przy tym żadnej z nich. W ogólnym przypadku punkt może nie leżeć na żadnej z prostych - w rozważanym przypadku leży na prostej drugiej, na co pozwoliły warunki zadania. Nie jest jednak rozwiązaniem w sensie algebraicznym (stąd nazwa - pseudo-rozwiązanie), bo nie spełnia równania "w całości". Jednakże zastosowanie w technice (np. w obróbce danych pomiarowych) tego rodzaju uogólnienia rozwiązania układu jest spore - często

spośród dużej liczby informacji, często wzajemnie wykluczających się, należy wybrać jedno - nie idealne, ale optymalne.
Przy układach nad-określonych i zastosowaniu dzielenia "\" Matlab wykonuje kilka operacji w celu uzyskania postaci kwadratowej. Te operacje sprowadzają się do budowy nowego układu równań wg następujących wzorów

$$\underset{[n\times n]}{C}\,\underset{[n\times 1]}{x} = \underset{[n\times 1]}{d} \quad , \quad \begin{array}{l} \underset{[n\times n]}{C} = \underset{[n\times m]}{A}^{T}\,\underset{[m\times n]}{A} \\ \underset{[n\times 1]}{d} = \underset{[n\times m]}{A}^{T}\,\underset{[m\times 1]}{b} \end{array}$$

Oczywiście wzory te mają swoje uzasadnienie, ale na razie nie będzie ono przytaczane. Proszę zwrócić uwagę na fakt, iż w wyniku tych "sprytnych" mnożeń wymiar równaniowy "m" jest kasowany - "nie pamięta" go zarówno macierz C, jak i wektor d nowego układu równań. Sprawdźmy, czy tak faktycznie jest, budując macierz C i wektor d na podstawie ostatniego rozważanego układu prostokątnego i rozwiązując go bez użycia operatora "\"

```
>> C = A'*A
>> d = A'*b
>> x = inv(C)*d
```

Wynik jest identyczny z tym, otrzymanym wcześniej w wyniku bezpośredniego "dzielenia" (czyli rozwiązania układu)

```
>> x = A\b
```

Analiza układów pod-określonych jest nieco trudniejsza. Rozważmy następujący przykład takiego układu o dwóch równaniach i trzech niewiadomych

$$\begin{cases} x_1 + x_2 + x_3 = 3 \\ x_1 - 2x_2 + 3x_3 = 2 \end{cases}$$

Wprowadźmy dane do Matlaba

```
>> A = [1 1 1; 1 -2 3]
>> b = [3;2]
```

Dokonajmy analizy rzędów macierzy

```
>> rank(A)
>> rank([A b])
```

Obydwa rzędy są równe 2 - rozwiązanie istnieje. Jednakże liczba niewiadomych równa 3 kieruje nas do wariantu (ii) twierdzenia Kroneckera-Capellego. Zatem nie istnieje rozwiązanie jednoznaczne (jedna jedyna trójka liczb x_1, x_2, x_3), ale nieskończenie wiele rozwiązań - rodzina rozwiązań zależna od jednego (3-2=1) parametru. Nazwijmy go C. Wtedy zastępując jedną z niewiadomych (np. x_1) tym parametrem, możemy zapisać

$$\begin{cases} x_2 + x_3 = 3 - c \\ -2x_2 + 3x_3 = 2 - c \end{cases} \rightarrow \quad Cx = d - c\begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad , \quad C = \begin{bmatrix} 1 & 1 \\ -2 & 3 \end{bmatrix} \quad d = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

Rozwiązując w sposób tradycyjny układ równań $Cx = d$

```
>> x = inv([1 1;-2 3])*[3;2]
```

otrzymamy rozwiązanie $x = \begin{bmatrix} 1.4 \\ 1.6 \end{bmatrix}$. Należy go uzupełnić o dodatkowy "parametryczny" fragment $-c \cdot C^{-1}\begin{bmatrix} 1 \\ 1 \end{bmatrix}$. Pomocniczego mnożenia macierzy odwrotnej do C i wektora [1;1] możemy dokonać w Matlabie

```
>> inv(C)*[1;1]
```

co daje skorygowany o parametr wektor rozwiązania $x(c) = \begin{bmatrix} 1.4 \\ 1.6 \end{bmatrix} - c\begin{bmatrix} 2.5 \\ -1 \end{bmatrix}$, zależny od tego parametru. Przyjmując zatem za parametr c (odpowiadający niewiadomej x_1) dowolną liczbę rzeczywistą obliczymy dwie pozostałe niewiadome x_2, x_3. Zobaczmy jeszcze, jakie rozwiązanie wyprodukuje Matlab przy zapisie z dzieleniem lewostronnym

```
>> x = A\b
```

Jest to jedno z możliwych rozwiązań rozważanego układu równań, dla parametru c równego 0.

Kolejnym zagadnieniem, które będzie poruszone, jest problem obliczenia długości wektora. Długość wektora $a = [a_1 \quad a_2 \quad ... \quad a_n]$ to liczba nieujemna będąca pierwiastkiem kwadratowym sumy kwadratów jego współrzędnych

$$|a| = \sqrt{\sum_{i=1}^{n} a_i^2}$$

Uwaga! W powyższym wzorze zapis $|a|$ nie oznacza już wyznacznika (jak poprzednio), ale długość wektora a (lub bardziej ogólnie: normę kwadratową). Istnieje kilka możliwości obliczenia tej wielkości w Matlabie, z wykorzystaniem poznanych już wcześniej operacji, a także kilku nowych. Zdefiniujmy w Matlabie wektor wierszowy

```
>> a = [1 2 -3]
```

Obliczenie długości wektora za pomocą iloczynu skalarnego

```
>> sqrt(a*a')
>> sqrt(dot(a,a))
```

W powyższych zapisach korzystamy z faktu, iż iloczyn skalarny wektora przez samego siebie to nic innego, jak suma iloczynu współrzędnych wektora - wynik ten wystarczy pierwiastkować (funkcja "sqrt").

Obliczenie długości wektora za pomocą działania tablicowego

```
>> sqrt(sum(a.*a))
>> sqrt(sum(a.^2))
```

W powyższym zapisie a.*a (oraz a.^2) oznacza wektor złożony z kwadratów elementów wektora a. Należy je dodatkowo zsumować ze sobą (funkcja "sum") oraz pierwiastkować.

Obliczenie długości wektora za pomocą funkcji "norm"

```
>> norm(a)
```

Najprostsza w działaniu funkcja pozwala na wiele więcej czynności. Krótki spis jej możliwości uzyskamy wpisując polecenie

```
>> help norm
```

Wyświetla się opis działania funkcji dla macierzy i dla wektorów. Przyjrzyjmy się składni dla wektorów ("For vectors…"). Wywołanie

```
>> norm(a)
```

jest równoważne wywołaniu

```
>> norm(a,2)
```

co, wg opisu (w składni Matlab), oznacza dla dowolnego drugiego argumentu "p" normę postaci

$$\|a\|_p = \left(\sum_{i=1}^{n} |a_i|^p \right)^{\frac{1}{p}}$$

Powyższy wzór (podwójne kreski $\|a\|$ oznaczają normę, a wewnątrz wzoru $|a_i|$ oznacza wartość bezwzględną) stanowi uogólnienie normy kwadratowej (długość wektora) na normy 0 potędze p. Obliczenie tej normy np. dla p = 3

```
>> norm(a,3)
```

powinno dać liczbę zbliżoną dla norm(a), ale jednak nieco różną. Skąd takie bogactwo normowe? Mianowicie każdą formułę można nazwać normą, o ile spełnia trzy warunki: jest nieujemna, jest równa zeru tylko wtedy, gdy argument jest zerem oraz spełnia tzw. warunek trójkąta ($\|a+b\| \le \|a\| + \|b\|$).

Inne wywołania normy wymagają podania jako drugiego argumentu "inf" lub "-inf". Dla "inf" otrzymamy

$$\|a\|_{\max} = \max_i |a_i|$$

największy z modułów wyrazów wektora a, a dla "-inf"

$$\|a\|_{\min} = \min_i |a_i|$$

najmniejszy z modułów wyrazów wektora a.

```
>> norm(a,inf)
>> norm(a,-inf)
```

Powyższych formuł nie należy mylić z funkcjami szukającymi największej i najmniejszej wartości spośród wyrazów wektora, gdyż powyższe odszukają największe i najmniejsze wartości bezwzględne. Z kolei funkcje "max" oraz "min" odszukają ekstremalne wartości bez modułów

```
>> max(a)
>> min(a)
```

Z kolei średnią elementów wektora możemy prosto obliczyć dzieląc sumę elementów tego wektora przez ich liczbę

```
>> sum(a)/3
```

Funkcje "max", "min", "prod" (funkcja obliczająca iloczyn elementów wektora) oraz "sum" dla argumentu macierzowego zwracają wektor wierszowy złożony odpowiednio z elementu największego, najmniejszego, iloczynu oraz sumy wyrazów w poszczególnych kolumnach. Jeżeli interesuje nas np. element maksymalny z całej macierzy, należy dwukrotnie wykonać dane polecenie

```
>> E = round(rand(4,4)*10)
>> max(max(E))
```

Sortowanie elementów wektora w kolejności rosnącej załatwia funkcja "sort"

```
>> b = round(rand(20,1)*100)
>> sort(b)
```

Dla wymuszenia kolejności malejącej należy dodać drugi argument w wywołaniu funkcji "sort"

```
>> sort(b,'descend')
```

Sortowanie elementów macierzy względem wybranej kolumny realizuje polecenie "sortrows". Jako argumenty podajemy nazwę macierzy oraz numer kolumny, względem której funkcja ma posortować macierz. Domyślnie elementy będą segregowane rosnąco, dla porządku malejącego należy numer kolumny poprzedzić znakiem "-". Przykład posortowania macierzy względem kolumny drugiej rosnąco i malejąco

```
>> A = round(rand(20,2)*100)
```

```
>> A = sortrows(A,2)
>> A = sortrows(A,-2)
```

Cała siła tego polecenia polega na tym, iż w ślad za elementami kolumny drugiej podążają elementy kolumny pierwszej, co przypomina sortowanie dostępne np. w arkuszach kalkulacyjnych typu Excel.
Informacji o rozmiarach macierzy bądź wektora dostarcza funkcja "size"

```
>> size(A)
>> size(b)
>> E = [0 1 2 3; -1 0 1 2; 0 2 3 4; 2 -1 2 0]
>> size(E)
```

Zwraca ona wierszowy wektor dwuelementowy, zawierający liczbę wierszy (jako pierwszy element) oraz liczbę kolumn (jako drugi element).
Z kolei funkcja "length" wbrew nazwie nie zwraca długości wektora (to robi omawiana funkcja "norm"), a największy z rozmiarów (liczbę wierszy bądź liczbę kolumn, w zależności od tego, która jest większa), np.

```
>> length(A)
>> length(b)
>> length(E)
```

Wszystkie funkcje matematyczne i zaokrąglające omawiane w pierwszych rozdziałach (sqrt, sin, cos, abs, log, round itd.) są tak skonstruowane, iż jeżeli ich argumentem jest macierz lub wektor, to w rezultacie dostajemy macierz lub wektor wartości danej funkcji

```
>> x = (1:10)'
>> sqrt(x)
>> abs(x)
>> sin(x)
>> exp(x)
>> log(x)
>> round(sqrt(x))
```

Należy jednak uważać przy zamianie tych funkcji na ich algebraiczne odpowiedniki. Np. funkcja "sqrt" dla argumentu macierzowego lub wektorowego działa inaczej niż podniesienie do potęgi 0.5. Funkcja "sqrt" dla takiego argumentu oblicza zestaw pierwiastków kwadratowych wszystkich elementów tablicy (czyli działa na wzór operacji tablicowej - "z kropką"), a operacja "^0.5" dokonuje macierzowego potęgowania (czyli w przypadku wektora "x" działanie zakończy się błędem). Aby uzyskać w obydwu przypadkach to samo, należałoby napisać ".^0.5"

```
>> sqrt(x)
>> x^0.5
>> x.^0.5
```

13. Ćwiczenia do samodzielnego wykonania

1. *Które z poniższych działań da się wykonać? Jeżeli nie, napisz, dlaczego, jeżeli tak, wykonaj je. Obliczenia ręczne skontroluj wykonując odpowiednie polecenia w Matlabie*

$$\boldsymbol{A}=\begin{bmatrix}3 & 1\\ -1 & 2\end{bmatrix},\quad \boldsymbol{B}=\begin{bmatrix}3 & 1\\ -1 & 2\\ 0 & -2\end{bmatrix},\quad \boldsymbol{C}=\begin{bmatrix}-1 & -2\\ 1 & 2\end{bmatrix},\quad \boldsymbol{D}=\begin{bmatrix}3 & -2 & -1\\ 1 & 0 & 1\end{bmatrix}$$

$$\boldsymbol{a}=\begin{bmatrix}2 & 1\end{bmatrix},\quad \boldsymbol{b}=\begin{bmatrix}1\\ 2\\ 3\end{bmatrix},\quad \boldsymbol{c}=\begin{bmatrix}-1 & 3\end{bmatrix}$$

$\boldsymbol{A}+\boldsymbol{B}$

$\boldsymbol{A}+\boldsymbol{C}$

$\boldsymbol{B}+\boldsymbol{D}^{T}$

$-2\cdot\boldsymbol{B}$

$\boldsymbol{A}\cdot\boldsymbol{D}$

$\boldsymbol{a}\cdot\boldsymbol{A}$

$\boldsymbol{b}^{T}$

$\boldsymbol{B}\cdot\boldsymbol{D}$

$\boldsymbol{c}^{2}$

$\boldsymbol{A}^{2}$

$\boldsymbol{C}^{-1}$

$\boldsymbol{B}^{-1}$

$(\boldsymbol{B}\cdot\boldsymbol{D})^{-1}$

2. *Oblicz ręcznie poniższe wyznaczniki. Wynik skontroluj w programie Matlab*

$$\left|-2\right|,\quad \begin{vmatrix}-1 & 1\\ 2 & -3\end{vmatrix},\quad \begin{vmatrix}-2 & -1 & 3\\ 0 & 1 & 2\\ 3 & -1 & -2\end{vmatrix},\quad \begin{vmatrix}1 & -1 & 0 & 1\\ -1 & 1 & 1 & 0\\ 0 & 1 & 1 & -1\\ 1 & 0 & -1 & 1\end{vmatrix}$$

3. *Dokonaj analizy liczby rozwiązań układów równań stosując odpowiednie twierdzenia. Jeżeli rozwiązania istnieją, znajdź je posługując się definicją lub metodą Cramera. Układy prostokątne mające rozwiązania doprowadź do układów kwadratowych i rozwiąż. Wyniki sprawdź posługując się programem Matlab.*

$$\begin{bmatrix}-1 & 1\\ 2 & -3\end{bmatrix}\begin{bmatrix}x\\ y\end{bmatrix}=\begin{bmatrix}1\\ -4\end{bmatrix},\qquad \begin{bmatrix}-2 & -1 & 3\\ 0 & 1 & 2\\ 3 & -1 & -2\end{bmatrix}\begin{bmatrix}x\\ y\\ z\end{bmatrix}=\begin{bmatrix}5\\ 8\\ -5\end{bmatrix}$$

$$\begin{bmatrix}-1 & 1\\ 2 & -3\\ -2 & 2\end{bmatrix}\begin{bmatrix}x\\ y\end{bmatrix}=\begin{bmatrix}1\\ -4\\ 2\end{bmatrix},\qquad \begin{bmatrix}-1 & 1 & 0\\ -2 & 2 & 1\end{bmatrix}\begin{bmatrix}x\\ y\\ z\end{bmatrix}=\begin{bmatrix}1\\ 5\end{bmatrix}$$

4. *Zadeklaruj w języku MATLAB macierze **A**, **B**, **C** i **D** oraz wektory **a**, **b** i **c** z zadania nr 1. Dodatkowo wygeneruj losowo macierz **E** o rozmiarze 20x30 złożoną z liczb całkowitych z zakresu od -2 do 7. Zapisz składnię poleceń realizujących automatycznie*

   - Iloczyn skalarny ***a*** i ***c***
   - Obliczenie długości wektora ***b***
   - Odwrócenie macierzy ***A***
   - Transpozycję macierzy ***C***
   - Obliczenie wyznacznika macierzy ***C***
   - Rozwiązanie układu równań $\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{c}^T$
   - Rozwiązanie układu równań $\boldsymbol{B} \cdot \boldsymbol{x} = \boldsymbol{b}$
   - Podniesienie współczynników macierzy ***A*** do potęg określonych poprzez odpowiadające im współczynniki macierzy ***C***
   - Podniesienie współczynników macierzy ***A*** do potęgi 3ciej
   - Dodanie odpowiadających sobie wyrazów macierzy ***A*** i ***C***
   - Mnożenie odpowiadających sobie wyrazów macierzy ***A*** i ***C***
   - Usunięcie z macierzy ***D*** trzeciej kolumny
   - Dodanie do macierzy ***B*** trzeciej kolumny w postaci wektora ***b***
   - Dodanie do pierwszego wiersza macierzy ***B*** dowolnej kombinacji pozostałych wierszy
   - Wybór z macierzy ***E*** wszystkich kolumn o numerach podzielnych przez 5
   - Wyzerowanie nieparzystych wierszy macierzy ***E***
   - Zamianę miejscami drugiego wiersza i pierwszej kolumny macierzy ***C***
   - Obliczenie wybranego podwyznacznika macierzy ***B***
   - Utworzenie macierzy dołączonej do macierzy ***C***
   - Obliczenie dopełnienia algebraicznego dowolnego elementu macierzy ***A***
   - Odjęcie od macierzy ***A*** macierzy jednostkowej
   - Obliczenie średniej wszystkich elementów macierzy ***E***
   - Obliczenie maksymalnych elementów w poszczególnych wierszach macierzy ***E***
   - Posortowanie elementów macierzy E względem trzeciego wiersza.