Wykład 1: Wprowadzenie do Matlaba

1 Informacje wstępne

Na mojej stronie internetowej

`http://www.ii.uni.wroc.pl/~asz/`

znajdą Państwo materiały pomocnicze do wykładu oraz informacje organizacyjne związane z kolokwiami, egzaminem,...

2 Informacje bardzo podstawowe

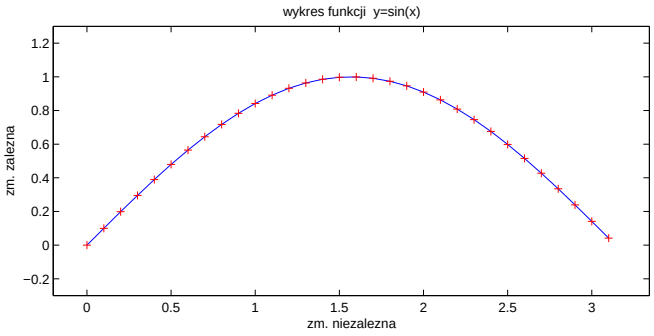
1. **Matlab używa zmiennych o wartościach rzeczywistych, zespolonych, posiada stałe...**

   » **pi+1** $\Longrightarrow$ $ans = 4.1416$ (ans od słowa ANSWER, jest stała π),
   » **realmax** $\Longrightarrow$ $ans = 1.7977e+308$,
   » **sqrt(-1)** $\Longrightarrow$ $ans = 0 + 1.0000i$ (liczba zespolona),
   » **a=i+5** $\Longrightarrow$ $a = 5.0000 + 1.0000i$ (i jest jedn. urojoną, instrukcja podstawienia spowodowała zadeklarowanie zmiennej a i nadanie jej wartości zespolonej),
   » **eps** $\Longrightarrow$ $ans = 2.2204e-016$ (też stała, o niej później).

   Wykonując powyższe instrukcje Matlab zadeklarował zmienne: ***ans*** oraz ***a***.

2. **Podstawową strukturą danych Matlaba jest macierz** - (nazwa Matlaba pochodzi od słów MATrix LABoratory). **Prostą i wygodną jest grafika:**

```
>> u=0:.1:pi;
>> v=sin(u);
>> plot(u,v,'b',u,v,'r+')
>> xlabel('zm. niezalezna')
>> ylabel('zm. zalezna')
>> title('wykres funkcji  y=sin(x)')
>> axis equal
>> axis([-.2 pi+0.2 -0.3 1.3])
>> set (gcf, 'Color', 'w')
```



   Dodatkowo zostały zadeklarowane zmienne (macierze) ***u*** oraz ***v***.

3. **Popularną jest wymiana plików między użytkownikami**. W internecie, pod adresem:

   `http://www.mathworks.com/matlabcentral/`

   jest ***MATLAB Central; An open exchange for the MATLAB and Simulink user community***, w którym można znaleźć sporo użytecznych funkcji zamieszczonych przez użytkowników Matlaba i udostępnionych innym:

3 Przegląd podstawowych komend Matlaba

matlab – uruchamia Matlaba ***w aktualnej kartotece***.

exit, quit – kończy pracę Matlaba.

demo – otwiera okno w którym można oglądać prezentacje możliwości Matlaba.

edit – uruchamia wygodny edytor matlabowski. Wygodnie jest zapamiętywać planowane do wykonania sekwencje komend Matlaba w postaci *skryptu* tak, aby np. łatwiej coś w nim poprawić, a sam przydatny skrypt zachować...

help – drukowany jest długi plik z nazwami *topics* – są to nazwy kartotek Matlaba, jego funkcyj, itd...
Można zajrzeć suwakiem wyżej w okienku, ale wygodnie jest wcześniej wydać komendę:

more on – po zapełnieniu okna Matlab czeka na naciśnięcie *s*pacji lub innego klawisza, by drukować dalej. Odwołanie: **more off**.

help help – informuje dokładniej o działaniu komendy *help*, a

type help – drukuje zawartość komentarza w pliku *help.m*.

lookfor – pozwala na uzyskanie informacji o interesujących nas zbiorach matlabowskich w sytuacji, gdy niezbyt dokładnie znamy ich nazwę, np. komenda » *lookfor real* wyświetla:

- REALMAX Largest positive floating point number.
- REALMIN Smallest positive floating point number.
- ISREAL True for real array.
- REAL Complex real part.
- ...

Dłuższe wyświetlanie odpowiedzi, a także pracę Matlaba można przerwać wciskając **Ctrl^C**.

who – informuje o zadeklarowanych i istniejących zmiennych; otrzymujemy:

Your variables are: A X a ans u v x

(Widać, że Matlab rozróżnia litery duże i małe.)

whos – to samo, ale dokładniej – otrzymujemy:

Name	Size	Bytes	Class
A	1x1	8	double array
X	1x1	8	double array
a	1x1	16	double array (complex)
ans	1x1	8	double array
u	1x32	256	double array
v	1x32	256	double array
x	1x1	8	double array

Grand total is 69 elements using 560 bytes

what – Matlab drukuje jedynie nazwy matlabowskich zbiorów (*.m* lub *.mat*) w aktualnej kartotece.

exist <nazwa> – Matlab sprawdza czy zmienna lub funkcja o podanej nazwie jest dostępna.
Sprawdzić *helpem* jak wyglądają odpowiedzi.

which <nazwa> – Matlab znajduje szukany zbiór matlabowski <*nazwa*> i podaje ścieżkę do niego, np. ***w*hich dist**.

format –

Obejrzyjmy:

» **X=realmax** $\Longrightarrow$ `X = 1.7977e+308`
» **x=realmin** $\Longrightarrow$ `x = 2.2251e-308`
» **A=bitmax** $\Longrightarrow$ `A = 9.0072e+015` (help: BITMAX Maximum floating point integer.)

Widać, że liczby są drukowane jedynie z 5-ma ***cyframi znaczącymi*** (więcej o tym – później).

- **help format** informuje m.in:

  FORMAT Set output format...

  FORMAT SHORT – Scaled fixed point format with 5 digits.
  FORMAT LONG – Scaled fixed point format with 15 digits for double and 7 digits for single.
  FORMAT LONG E – Floating point format with 15 digits for double and 7 digits for single.
  FORMAT HEX – Hexadecimal format.

Obejrzyjmy kilka wartości w układzie szesnastkowym:

» **format hex**		
» **1**	$\Longrightarrow$	ans = `3ff0000000000000`
» **2**	$\Longrightarrow$	ans = `4000000000000000`
» **3**	$\Longrightarrow$	ans = `4008000000000000`
» **4**	$\Longrightarrow$	ans = `4010000000000000`
» **bitmax**	$\Longrightarrow$	ans = `433fffffffffffff`
» **1024*bitmax**	$\Longrightarrow$	ans = `43dfffffffffffff`
» **realmin**	$\Longrightarrow$	ans = `0010000000000000`
» **realmin*2**	$\Longrightarrow$	ans = `0020000000000000`
» **realmax**	$\Longrightarrow$	ans = `7fefffffffffffff`
» **realmax*2**	$\Longrightarrow$	ans = `7ff0000000000000`
» **format long e**		
» **realmax**	$\Longrightarrow$	ans = `1.797693134862316e+308`
» **realmax+100000**	$\Longrightarrow$	ans = `1.797693134862316e+308`
» **realmax*1.1**	$\Longrightarrow$	ans = Inf
» **1/0**	$\Longrightarrow$	ans = Inf (Warning: Divide by zero.)
» **0/0**	$\Longrightarrow$	ans = NaN (Warning: Divide by zero.)

date, clock – data, data i czas. Czasem warto takie informacje umieścić w wynikach.
Obejrzeć *helpem*. Wypróbować: **clock**, **fix(clock)**.

tic, toc – obie funkcje służą do mierzenia czasu potrzebnego na wykonanie określonych obliczeń:
tic - zapamiętuje aktualny czas
obliczenia ...
toc - mierzy i drukuje czas od ostatniego ***tic***u, np. *Elapsed time is 5.407450 seconds.*
Można też zapamiętać zużyty czas instrukcją: $t = toc$ Otrzymujemy np. $t = 51.6711$

clear – możemy zwalniać pamięć (*workspace*) zajmowaną przez zmienne komendami:

clear a, **clear**. Więcej – proszę sprawdzić *helpem*...

clc – czyścimy okno komend.

diary – przebieg sesji z Matlabem możemy kopiować do pliku (zapamiętać) korzystając z komend:

diary <nazwa> – otwieramy plik <nazwa> do zapisywania przebiegu sesji,
diary off – przerywamy zapis zamykając zbiór ze sprawozdaniem,
diary on – wznawiamy zapis.

echo on – po uruchomieniu skryptu Matlab drukuje w okienku komend kolejne wiersze skryptu (komentarze również) poczynając od pierwszej instrukcji w skrypcie (tzn. pomija komentarz *helpa*). Wygodne przy *debugowaniu programu*. Porównać z **diary**.
Odwołanie: **echo off**.

dir, ls – możemy wydrukować zawartość kartoteki w której Matlab pracuje.

cd <nazwa> – możemy zmienić aktualną kartotekę. (Można wykorzystać ***cd ..*** aby przejść wyżej.)

delete <nazwa> – usuwamy zbiór z kartoteki.

save – zapamiętuje wartości wszystkich istniejących zmiennych w pliku *matlab.mat* tak, aby później można było ruszyć z obliczeniami dalej. Można też zapamiętać wartości jedynie wybranych zmiennych, oraz można skorzystać z pliku o wybranej nazwie, np. komenda ***save*** *a A x a* zapamiętuje wartości zmiennych: *A, x, a* w pliku *a.mat*.

load – na odwrót – wprowadza zmienne i ich wartości zapisane w pliku *matlab.mat* .
(I analogicznie, komenda ***load*** *a* wprowadzi wartości zmiennych: *A, x, a*).

4 Zadania na Ćwiczenia i na Pracownię

1. Uruchomić **demo** – obejrzeć możliwości Matlaba.

2. Sprawdzić *helpem* jaką wartość ma stała **eps**, a potem sprawdzić komendami Matlaba – czy rzeczywiście.

3. W komputerach obowiązuje reprezentacja dwójkowa pamiętanych liczb. Czy pamiętają Państwo jeszcze algorytm zamiany postaci liczb z układu dziesiętnego na dwójkowy?
A z układu dwójkowego na dziesiętny?

4. Zapisać liczbę 123.456 w systemie dwójkowym z trzema cyframi po kropce
(ostatnia cyfra powinna być prawidłowo zaokrągloną).

5. ***(zad. teoretyczne)*** Wyobraźmy sobie arytmetykę o podstawie 10, z dwiema cyframi cechy i czterema cyframi mantysy (cecha i mantysa ze znakiem) – omówimy na wykładzie:

$$\pm\,.\square\square\square\square_{\,10}\pm\square\square\ ?$$

Proszę zastanowić się, jak na osi liczbowej rozkładają sie liczby reprezentowane dokładnie w takiej postaci. Jaka liczba jest największą, najmniejszą dodatnią, jak wyglądają różnice pomiędzy kolejnymi liczbami?

Ile wynosi największa odległość pomiędzy dwiema kolejnymi liczbami pamiętanymi w komputerze dokładnie? A najmniejsza odległość?

6. Oglądając **lookfor** widzieliśmy, że Matlab korzysta z arytmetyki ***floating point***, a oglądając **whos** – że liczby są pamiętane w typie ***double***. Proszę znaleźć więc takie liczby *a, b, c*, żeby $(a+b)+c \neq a+(b+c)$.
Zadanie trudne: *znaleźć takie a, b, c aby otrzymać największą różnicę pomiędzy lewą i prawą stroną powyższej nierówności...*

7. W Matlabie można używać: ***NaN, Inf*** (zob. tabelkę na str. 2). Sprawdzić, czy inne sekwencje dużych i małych liter dla tych symboli również są dopuszczalne.

8. Obejrzeliśmy trzy formaty. Proszę wypróbować inne...

9. Jaką maksymalną długość mogą mieć nazwy rozróżnialnych zmiennych w Matlabie? Wypróbować doświadczalnie różne (wygodne) nazwy zmiennych.

10. Dowiedzieć się *helpem* o własnościach komend **diary**, **save**, **load**, a następnie wypróbować to w praktyce.

5 Zmienne, operatory, instrukcje

1. Podstawową strukturą danych jest zmienna tablicowa (macierz) - pamiętamy: **MAT**rix **LAB**oratory
skalar jest macierzą rozmiaru 1×1.
Wartości zmiennym można nadawać explicite ***instrukcją podstawienia***, np:

```
» x=1;
» A=[ 1 2 3
      4 5 6];   albo   A=[ 1 2 3; 4 5 6];   – średnik wewnątrz oznacza koniec wiersza macierzy,
» a=[1 0 2]';                               – transpozycja, a staje się kolumną,
» b=A*a;                                    – macierz 2×3 można przemnożyć przez wektor 3×1,
» x1=1:10^2;                                – domyślnie - krok = 1, x1 zmienia się do 100,
» x2=1:3.6:10;                              – teraz krok = 3.6, jaki jest rozmiar zmiennej x2 ?
» x3=100:-4:11;                             – krok = −4.
```

1a. Nazwa zmiennej w Matlabie składa się z liter, cyfr lub znaku podkreślenia i **musi zaczynać się od litery**.

1b. Rozmiar zmiennej (macierzy) wyznaczymy funkcją **size**:
» **size(A)** $\Longrightarrow ans = 2 \quad 3$.

2a. Operatory arytmetyczne (obejrzeć *help ops*, *help arith*):

^ | *, /, \ | +, -	–	wymienione grupami od najsilniejszych do najsłabszych; jest *lewe dzielenie*, a przy potęgowaniu co najmniej jeden z argumentów musi być skalarem. Nazwy operatorów: *matrix power*, *matrix multiply*...
.^ | .*, ./, .\	–	dopisanie kropki powoduje, że operacja jest wykonywana *componentwise* tzn. na poszczególnych elementach macierzy. Nazwy operatorów: *array power*, *array multiply*...

Ponieważ operacje te można wykonywać ogólnie na macierzach wielowymiarowych, więc odpowiednie rozmiary argumentów operacji ***muszą być zgodne***.

2b. Pierwszeństwo operatorów możemy zmieniać stosując nawiasy okrągłe.

3. Transpozycja realizuje sprzężenie dla argumentu zespolonego (była użyta w p. 1)

» **a=[1 2+i]** $\Longrightarrow \quad a = [\,1.0000 \quad 2.0000 + 1.0000i\,]$,

» **a'** $\Longrightarrow \quad ans = \begin{bmatrix} 1.0000 \\ 2.0000 - 1.0000i \end{bmatrix}$, *(complex conjugate transpose)*

» **a.'** $\Longrightarrow \quad ans = \begin{bmatrix} 1.0000 \\ 2.0000 + 1.0000i \end{bmatrix}$, *(non-conjugate transpose)*

» **(a.')' = (a').' = a'.' = a."** $\Longrightarrow \quad ans = [\,1.0000 \quad 2.0000 - 1.0000i\,]$.

4. Instrukcje (komendy) mają postać:
zmienna = ***wyrażenie*** (instrukcja podstawienia) lub
wyrażenie (np. wywołanie funkcji, lub obliczanie wartości wyrażenia jak kalkulator).

4a. Jeżeli instrukcja jest zbyt długa i nie mieści się w wierszu,
to możemy jej fragment przenieść do linii następnej wykorzystując trzy kropki, np:

```
» b=A*...
   a;
```

4b. Instrukcje pisane w jednym wierszu oddzielamy od siebie średnikiem lub przecinkiem.

4c. Jeżeli instrukcja nie jest zakończona średnikiem, to Matlab wypisuje otrzymany wynik instrukcji w okienku komend.

5. Do dyspozycji jest wiele **standardowych funkcji arytmetycznych** m.in.: *sqrt*, *exp*, *log*, *log10*, *log2*, *sin*, *cos*, *tan*, *abs*, *sqrt*, *sign*, *fix*, *floor*, *ceil*, *round*, *mod*, *rem*... ***help elfun*** wyświetla m. in:

Rounding and remainder.

fix	-	Round towards zero.
floor	-	Round towards minus infinity.
ceil	-	Round towards plus infinity.
round	-	Round towards nearest integer.
mod	-	Modulus (signed remainder after division).
rem	-	Remainder after division.
sign	-	Signum.

Proponuję obejrzeć wyjaśnienia Matlaba na temat definicji wymienionych funkcji... Np:

```
>> help rem
 REM    Remainder after division.
    REM(x,y) is x - n.*y where n = fix(x./y) if y ~= 0.  If y is not an
    integer and the quotient x./y is within roundoff error of an integer,
    then n is that integer. The inputs x and y must be real arrays of the
    same size, or real scalars.

    By convention:
       REM(x,0) is NaN.
       REM(x,x), for x~=0, is 0.
       REM(x,y), for x~=y and y~=0, has the same sign as x.

    Note: MOD(x,y), for x~=y and y~=0, has the same sign as y.
```

6. Często, pisząc instrukcję podstawienia możemy ***zasłonić*** istniejącą już nazwę, np:

» sin=5; Przywrócić funkcję *sin* możemy jedynie usuwając utworzoną zmienną *sin* komendą:
» clear sin;
Nazwa *end* nie da się zasłonić:
» end=5; $\Longrightarrow$ *??? end = 5 ... Error: Missing operator, comma, or semicolon.*

6 Generowanie macierzy

1. **explicite:** piszemy A = [1 2,3; 4 5 6];
spacja, przecinek oddzielają kolumny, a średnik lub *nowy_ wiersz* oddzielają wiersze.

2. **generowanie wektorów:**

$x = x1 : x2$;	–	tworzy wektor o składowych zmieniających się co 1,	$(x1 \leq x2)$
$x = x1 : dx : x2$;	–	składowe zmieniają się co dx,	
$x = linspace(x1, x2, n)$;	–	liniowo rozłożone składowe,	(domyślnie 100)
$x = logspace(x1, x2, n)$;	–	logarytmicznie rozłożone składowe od 10^{x1} do 10^{x2},	(domyślnie 50)

3. **funkcje standardowe, tworzące macierze**, oto niektóre z nich:

a=ones(m,n);	–	macierz jedynek rozmiaru $(m \times n)$,	ones(n)
a=zeros(m,n);	–	macierz zer rozmiaru $(m \times n)$,	zeros(n)
a=eye(n);	–	macierz jednostkowa rozmiaru $(n \times n)$,	eye(2,3)
a=diag([1 2 3],k);	–	macierz o zadanej k-tej diagonalnej,	diag([1 2 3])
A=rand(m,n);	–	macierz o wartościach losowych z $(0, 1)$,	rand(n), zob. ***help rand***
A=hilb(n);	–	macierz Hilberta rozmiaru $(n \times n)$,	
A=invhilb(n);	–	macierz odwrotna do m. Hilberta,	
A=magic(n);	–	kwadrat magiczny, n = 1, 3, ...	

4. **m–files:** Wygodnie jest wprowadzać większe dane ze zbiorów matlabowskich. Np. uruchomienie skryptu

   » amac

   czyli istniejącego zbioru ***amac.m*** o treści

```
A = [ 1 2 3
      4 5 6];
```

   powoduje zdefiniowanie zmiennej A, o wartości w postaci macierzy rozmiaru (2×3).

5. **czytanie z dysku:** Można również wprowadzić większe dane, np. ze zbioru ***amac.txt*** o treści

```
1 2 3
4 5 6
```

   wydając komendę

   » load amac.txt

   tym razem zostaje zdefiniowana zmienna *amac*, a jej wartością zostaje wczytana macierz rozmiaru (2×3).

7 Operacje na macierzach (array manipulation)

1. **standardowe funkcje operujące na macierzach:**

a=triu(A,k);	–	podmacierz trójkątna górna poczynając od k-tej przekątnej,	triu(A)
a=tril(A,k);	–	... dolna ...	tril(A)
a=diag(A,k);	–	k-ta diagonalna utworzona jako kolumna,	diag(A)
a=rot90(A);	–	obrót macierzy o 90^o p.d.r.w.z,	rot90(A,k)
a=fliplr(A);	–	obrót macierzy wokół pionowej osi symetrii,	
a=flipud(A);	–	obrót macierzy wokół poziomej osi symetrii,	
a=reshape(A,m,n);	–	zamiana macierzy A na postać $(m \times n)$,	A(:) - kolumna
y=det(A);	–	wyznacznik,	
a=inv(A);	–	macierz odwrotna,	
[m, n]=size(A);	–	m – liczba wierszy, n – liczba kolumn,	
n=length(A);	–	maksimum(l.wierszy, l.kolumn),	

2. **do operacji tablicowych należą wszystkie standardowe funkcje skalarne:**
   sin, cos, tan, cot, asin, acos, atan, acot, sinh, cosh, tanh, coth, asinh, acosh, atanh, acoth, exp, log, log10, abs, angle, real, imag, conj, sqrt, sign, rem, round, floor, fix, ceil, gcd, lcm.

3. a=sqrtm(A); ≡(?) a=A^0.5;

4. **funkcje wektorów:**
   max, min, sum, prod, sort, length, median, mean, std, any, all, cumprod, cumsum.

5. **działania na macierzach:** Niech A=[1 5 6; 4 2 8; 7 3 9]; b=[1 0 -3];. Wówczas:

» [A b']	–	dopisana kolumna,
» [A; b]	–	dopisany wiersz,
» [A b'; b 1]	–	dopisane: wiersz i kolumna,
» u=1:2; v=[1 3]; A(u, v)	–	wynik: [1 6; 4 8],
» A(:, 2)	–	druga kolumna,
» A(2, :)	–	drugi wiersz,
» A(:,[3:-1:1])	–	przestawione kolumny,
» b(b<0) = -b(b<0)	–	wynik: b=[1 0 3],
» A(1:2, 2:3)=ones(2,2)	–	wstawiona podmacierz jedynek,

8 Instrukcje warunkowe

(1) **if** <warunek> <instr1>; <instr2>; ...<instr$n$>; **end**
(zamiast średników mogą być przecinki — co to zmieni?)

(2) **if** <warunek>
<instrukcjeA>
else
<instrukcjeB>
end

(3) **if** <warunek1>
<instrukcjeA>
elseif <warunek2>
<instrukcjeB>
elseif <warunek3>
<instrukcjeC>
...
else
<instrukcjeE>
end

9 Wyrażenia logiczne

W instrukcji warunkowej, w miejscu <warunek> można umieścić

(1) ***wyrażenie arytmetyczne*** — <warunek> otrzymuje wartość:
true – jeżeli wartość wyrażenia jest różna od zera,
false – w przeciwnym razie.

(2) ***wyrażenie logiczne*** — może ono być zbudowane za pomocą następujących operatorów logicznych (podane zostały w kolejności pierwszeństwa):
<, <=, >, >=, ==, ~= – *relacje*,
~ – *negacja*,
& – *koniunkcja*,
| – *alternatywa*.
Przykład z równania kwadratowego:
if b^2-4*a*c==0&a~=0 x=-b/(2*a), **end**
(Widać, że warto używać nawiasów i rozstrzelić tekst w kilku wierszach.)

10 Pętle, instrukcje break i continue

1. **pętla *for*** **for** <zmienna> = <lista> definiująca kolejne wartości <zmiennej>
<instrukcje> wykonywane dla każdej wartości <zmiennej>
end

przykłady:
```
y=pi;  s=[];
for x=0:y, s=[s x];  end   % tworzenie wektora  s
for x=s,   h=x                % co bedzie wydrukowane?
end
a=[s; s' s'+1 s' [4 3 2 1]'];  % obejrzec macierz  a
for x=a,   h=x                % co bedzie wydrukowane?
end
```

2. **pętla *while*** **while** <wyrażenie>
<instrukcje>
end
<instrukcje> są wykonywane dopóty, dopóki <wyrażenie> ma wartość prawdy logicznej.

przykład:
```
n=0;  N=55.444;
while n^2 < N , n=n+1; end
n                                % co bedzie wydrukowane?
```

3. **instrukcja *break*** — powoduje przerwanie pętli.

4. **instrukcja *continue*** — powoduje pominięcie dalszych instrukcji wewnątrz pętli i przejście do następnego obrotu pętli.

11 Zadania na ćwiczenia

1. Niech » x = 30*pi/180; y = 1 - sin(x)ˆ 2 - cos(x)ˆ 2 Czy wyświetlony wynik będzie równy zeru? dlaczego?

2. Indeksy macierzy z założenia są wektorami. Łatwo można odwoływać się do podmacierzy. Jaki będzie skutek wykonania poniższych komend dla istniejącej macierzy A 5-go stopnia:

- A(2:4,[2 4])=5*ones(3,2);
- x=1:2:4; A([x],[x])=3*ones(2);
- x=1:2:4; A([x],[x])=3*ones(2); A([x+1], [x+1])=5*ones(2);
- x=A(:,3);
- x=A(:);
- A(:)=1:25;
- A(:)=(1:25)';
- A(:)=1:26;
- a=zeros(3,4); a(:)=1:12; b=reshape(a,2,6), c=reshape(a',2,6)

3. **Dana jest macierz** A**.**

(a) Napisać jedną instrukcję postaci $B = \ldots$, która wybiera (wycina) z macierzy A:
   i. trzeci element drugiego wiersza,
   ii. ostatni element drugiego wiersza,
   iii. ostatni element drugiej kolumny,
   iv. ostatnią kolumnę,
   v. wiersz trzeci od końca,
   vi. kolumny od 3 do 5 włącznie,
   vii. wiersze 2 i 4,
   viii. podmacierz o wierzchołkach $A(2,3)$ i $A(4,7)$,
   ix. podmacierz kwadratową stopnia 3, leżącą w prawym dolnym rogu macierzy A.

(b) **Przekształcić macierz** A**:**
   i. usunąć trzeci wiersz,
   ii. usunąć z macierzy wiersze o parzystych numerach,
   iii. zastąpić trzeci wiersz macierzy wierszem postaci: 1, 2, 3, ...,
   iv. dopisać na dole macierzy wiersz samych jedynek,
   v. zamienić miejscami wiersze 1 i 4,
   vi. zamienić miejscami kolumnę 2 z kolumną ostatnią.

4. Co będzie wydrukowane po wykonaniu programu:

```
clc, clear, close
x = [ ];
for a = [ 7 9 15 20 6 ]
    for b = a : 16
        x = [ x, b ];
        if  b < 10
            break
        end
    end
end
x
```

5. Zastąpić matlabowskie działania na macierzach: $C = A + B$, $C = A * B$, $C = D * (A + B)$ pętlami ***for***, a potem pętlami ***while***.

6. Proszę przypomnieć sobie na czym polega ***Paradoks Bertranda***.

12 Zadania na pracownię

1. Sprawdzić *helpem* jaką wartość ma stała **eps**, a potem sprawdzić komendami Matlaba – czy rzeczywiście.

2. Jakie wartości otrzymamy:

(a) $round(0.5) =$ **(b)** $round(-0.5) =$ **(c)** $fix(0.7) =$ **(d)** $fix(-0.7)' =$
(e) $ceil(0.2) =$ **(f)** $ceil(-0.2) =$ **(g)** $floor(0.7) =$ **(h)** $floor(-0.7)' =$
(i) $mod(20, 7) =$ **(j)** $mod(20, -7) =$ **(k)** $mod(-20, 7) =$ **(l)** $mod(-20, -7)' =$

3. Niech A i B będą macierzami kwadratowymi rozmiaru *size(A)* $\Longrightarrow$ *ans = 2 2* .
Sprawdzić jaka jest siła operatora transpozycji, np. kiedy nawiasy zmienią wynik działania?

(a) $A' + B' =$ **(b)** $(A + B)' =$ **(c)** $A' * B' =$ **(d)** $(A * B)' =$

4. Utworzyć macierze A=[1 2 4] , B = [4 5 6] , C =[7; 8; 9] , D=[10;11;12] .

5. Jaki jest efekt poleceń: E =[A B] , E =[C D] , E =[A' B'] , E =[C'; D'] ?

6. Utworzyć a =[1 2 3; 4 5 6] i sprawdzić, co to jest: a' .

7. (a) Utworzyć wierszowe wektory 1,3,5 7,9,11 13,15,17 i złożyć je w macierz 3×3 .
(b) Utworzyć kolumnowe wektory 1,7,13 3,9,15 5,11,17 i połączyć je w macierz 3×3 .

8. Obejrzeć: zeros(4), eye(3), ones(4), pascal(5), magic(3) i może jeszcze magic(2) .

9. Sprawdzić czy zadania z ćwiczeń zostały dobrze rozwiązane...

10. Sprawdzić działanie komend:

(a) rand , rand(2, 3) , rand(3, 2) , fix(rand) , fix(rand(3,2)) , fix(rand(2, 3)) , fix(10*rand) , i.t.d.
(b) Niech A = pascal(4) , B = ones(4) , C = A + B , C = A - B , C = A * B .
(c) Niech a = pascal(3) . Wyznaczyć 2 * a , a * 2 , a + 2 , 2 + a , a/2 , 2/a (czy jest tu błąd?).
(d) Niech A = pascal(4) , x=[1; 1; 1; 1] Wyznaczyć x' * A , A * x .
(e) A = [2 1; 1 2] , B = inv(A) , A * B .

11. Zaprogramować za pomocą pętli dodawanie oraz mnożenie macierzy.

12. A obliczanie wyznacznika macierzy kwadratowej?

13. (**zadanie dla chętnych, WARTO!**) Napisać program (skrypt), który na zadanym prostokącie: $XL <= x <= XP$, $YD <= y <= YG$

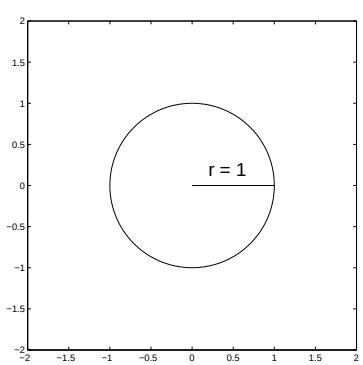
(a) będzie generować rozkładem jednostajnym dwa punkty,
(b) a potem połączy je odcinkiem wygenerowanego koloru...

Program powinien działać w nieskończonej pętli, a przerwać go można będzie klawiszami **Ctrl^C**.

14. (**zadanie dla chętnych, WARTO!**)
π

Aby wyznaczyć przybliżoną wartość π możemy skorzystać z teorii prawdopodobieństwa.
Prawdopodobieństwo trafienia w koło o promieniu 1 poprzez generowanie punktów losowych w kwadracie o boku 4 (jak na rysunku) jest równe $\frac{\pi}{16}$.
Zatem można przybliżać π wykorzystując liczbę trafień w koło dla coraz dłuższych sekwencji generowanych punktów:

$$\pi = 16 * \lim_{n\to\infty} \frac{l.trafień(n)}{n} .$$



* * *