Tryb interpretera

```
≫ 2 * 3

ans =

     6

≫ 2 * 3;
≫ pi

ans =

    3.1416

≫ sin(pi / 2)

ans =

     1

≫ [1 4; 6 8]

ans =

     1     4
     6     8

≫ 6: 2: 20

ans =

     6    8   10   12   14   16   18   20
```

```
≫ b

? Undefined function or variable 'b'

≫ A = [1 4; 6 8]

A =

   1     4
   6     8

≫ A - 1

ans =

     0     3
     5     7

≫ A * A

ans =

    25    36
    54    88

≫ sin(A)

ans =

    0.8415   -0.7568
   -0.2794    0.9894
```

Programowanie

Zadanie 1. Napisać funkcję znajdującą pierwiastek równania liniowego.

```
function x = rown1(a, b)
if a == 0
   if b ~= 0
      x = [];
   else
      x = NaN;
   end;
else
   x = -b / a;
end;
```

Uwaga: Tę funkcję zapisuje się w pliku rown1.m !

Efekt działania:

```
≫ rown1(1, 2)

ans =
    -2
```

Zadanie 2. Napisać funkcję rozwiązującą równania liniowe i kwadratowe.

```
function x = rown21(a, b, c)
if nargin == 2
   x = rown1(a, b);
elseif a == 0
   x = rown1(b, c);
else
   delta = b * b - 4 * a * c;
   if delta >= 0
      if b > 0
         x = (-b-sqrt(delta))/(2*a);
      else
         x = (-b+sqrt(delta))/(2*a);
      end;
      if x == 0
         x = [x 0];
      else
         x = [x (c/a)/x];
      end;
   end;
end;
```

Wskazana jest również diagnostyka błedów:

```
if nargin < 2
   error('Za malo parametrow');
end;
```

Pętla `for` i wektoryzacja

pętla `for`	wektoryzacja

`for i = 1: 100`
`  y(i)=sin(i);`
`end;`
	`y = sin(1: 100);`

Co wybierać? Zawsze wektoryzację!

Zadanie. Napisać funkcję obliczającą sumę *p*-tych potęg *n* pierwszych liczb naturalnych.

Wersja „klasyczna”:

```
function s = sump(n, p)
s = 0;
for i = 1: n
   s = s + i^p;
end;
```

Wersja zwektoryzowana:

```
function s = sumpv(n, p)
s = sum((1: n).^p);
```

Wektoryzacja jest możliwa, jeżeli operacje w pętli nie zależą od porządku, w ktorym są wykonywane.

Specyfika pętli `for`

Zadanie. Co będzie rezultatem poniższych instrukcji:

①
```
for i=12:-2:1; disp(i), end;
```

②
```
for i=12:-2:1; disp(i), ...
  i = 0; end;
```

③
```
for i=12:-2:1; disp(i); ...
  if i == 8, break; end; end;
```

Zadanie. Napisać funkcję wyznaczającą *n*-ty element ciągu Fibonacciego.

Wersja prostsza:

```
function f = fibs(n)
if n == 1 | n == 2
   f = 1;
else
   a = 1; b = 1;
   for i = 3: n
      c = a + b;
      a = b;
      b = c;
   end;
   f = c;
end;
```

Wersja bardziej efektywna:

```
function f = fib(n)
persistent rez
if length(rez) < 2
   rez = [1 1];
end;
```

```
if length(rez) < max(n)
   for i = length(rez)+1: max(n)
      rez(i)=rez(i-1)+rez(i-2);
   end;
end;
f = rez(n);
```

Efekt działania:

```
≫ fib([3, 9, 4])

ans =
     2    34     3
```

Zadanie. Napisać funkcję obliczającą *n* pierwszych linii trójkąta Pascala.

```
function c = trojkat(n)
c = zeros(n, n);
c(:,1) = 1;
for i = 2:n
   for j = 2:i
      c(i,j) = c(i-1,j-1)+c(i-1,j);
   end;
end;
```

Efekt działania:

```
≫ trojkat(6)

ans =
     1     0     0     0     0     0
     1     1     0     0     0     0
     1     2     1     0     0     0
     1     3     3     1     0     0
     1     4     6     4     1     0
     1     5    10    10     5     1
```

Sposoby wektoryzacji

Prześledźmy poniższą sesję:

```
≫ a = 1: 16;
≫ a = reshape(a, 4, 4)

a =
     1     5     9    13
     2     6    10    14
     3     7    11    15
     4     8    12    16
```

```
>> a1 = a(:, 4: -1: 1)

a1 =
    13     9     5     1
    14    10     6     2
    15    11     7     3
    16    12     8     4

>> a2 = a(4: -1: 1, 4: -1: 1)

a2 =
    16    12     8     4
    15    11     7     3
    14    10     6     2
    13     9     5     1

>> a3 = a'

a3 =
     1     2     3     4
     5     6     7     8
     9    10    11    12
    13    14    15    16
```

To samo wykonuje zresztą predefiniowana funkcja `nchoosek`.

Zadanie. Utworzyć wektor, w którym każdemu indeksowi parzystemu *p* będzie odpowiadać kwadrat, a indeksowi nieparzystemu – sześcian *p*-tej liczby naturalnej.

```
v(1: 2: n) = (1: 2: n).^3;
v(2: 2: n) = (2: 2: n).^2;
```

Zadanie. Mając dany wektor `x`, utworzyć tablicę `a` o *n* wierszach będących kopiami `x`.

```
≫ x = 1: 6; n = 5;
≫ a = x(ones(1, n), :)

a =
     1     2     3     4     5     6
     1     2     3     4     5     6
     1     2     3     4     5     6
     1     2     3     4     5     6
     1     2     3     4     5     6
```

Zadanie. Dla tablicy

```
A =
     3     0     3    -4    -4     1
    -1     3     2     1    -1     2
    -1     0     0     3     0     5
     1    -3    -4     2    -2     3
     1     2     2     5     1     2
     2     5    -1     5    -5    -1
```

wygenerować tablicę złożoną z elementów leżących w wierszach od trzeciego do piątego i kolumnach 6, 1 i 3.

```
≫ B = A(3: 5, [6 1 3])

B =
     5    -1     0
     3     1    -4
     2     1     2
```

Jak w tej tablicy wyzerować wiersze nieparzyste, a elementy większe od 3 zamienić na 1.

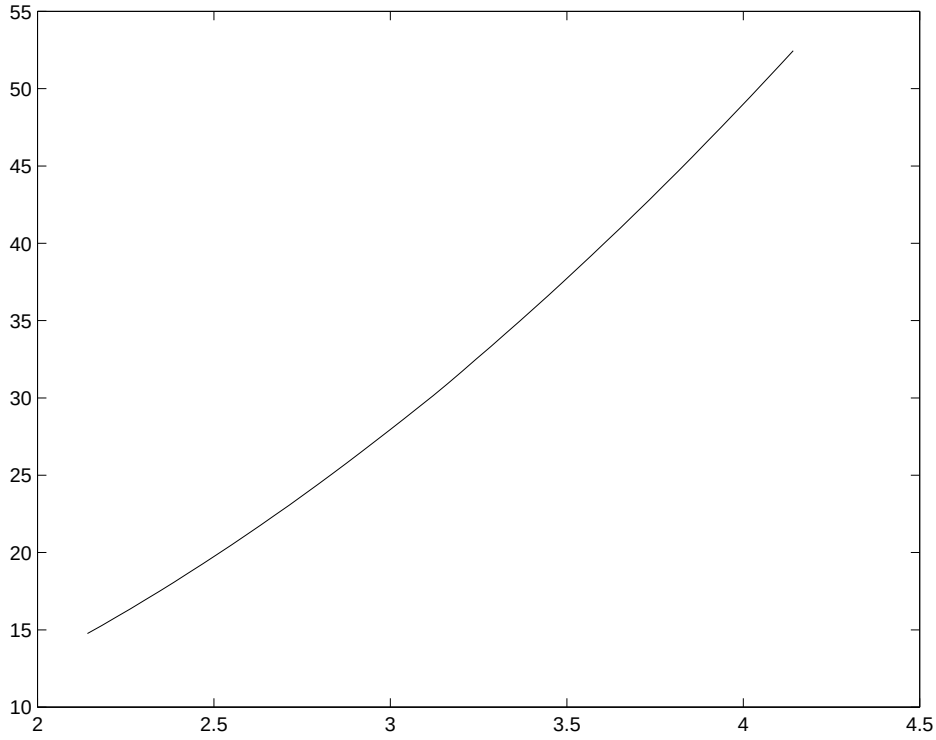
```
≫ A(1: 2: 6, :) = 0;
≫ A(A > 3) = 1;
```

Zadanie. Narysować wykres funkcji

$$y = 3x^2 + \frac{\ln(x-\pi)^2}{\pi^4} + 1$$

w przedziale $[\pi - 1, \pi + 1]$. Dlaczego nie widać nieciągłości w punkcie $x = \pi$?

```
x = linspace(pi-1, pi+1, 50);
y = 3*x.^2+log((x-pi).^2)/pi^4+1;
plot(x, y)
```



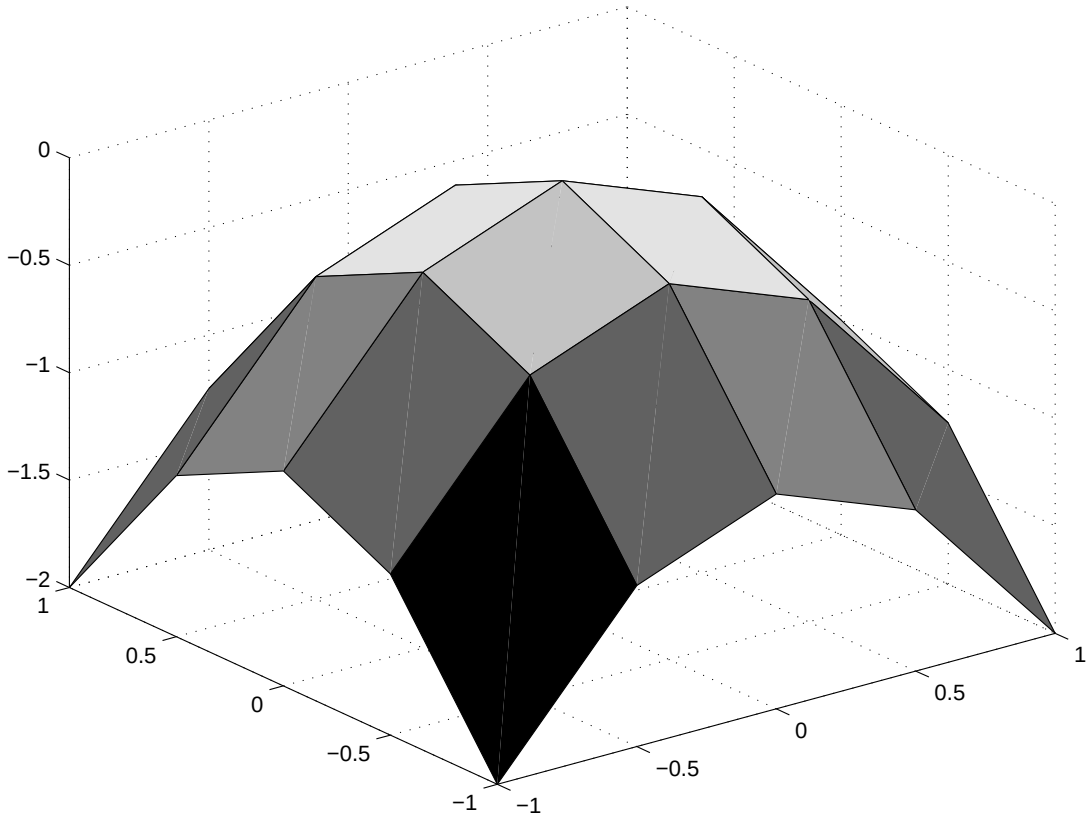
Zadanie. Narysować powierzchnię

$$z = -x^2 - y^2$$

dla $-1 \leq x,\ y \leq 1$.

```
≫ x = linspace(-1, 1, 5)

x =

   -1.00   -0.50    0    0.50   1.00

≫ y = x;
≫ [X, Y] = meshgrid(x, y)

X =

   -1.00 -0.50  0  0.50  1.00
   -1.00 -0.50  0  0.50  1.00
   -1.00 -0.50  0  0.50  1.00
   -1.00 -0.50  0  0.50  1.00
   -1.00 -0.50  0  0.50  1.00

Y =

   -1.00 -1.00 -1.00 -1.00 -1.00
   -0.50 -0.50 -0.50 -0.50 -0.50
       0     0     0     0     0
    0.50  0.50  0.50  0.50  0.50
    1.00  1.00  1.00  1.00  1.00
```

```
≫ Z = -X.^2 - Y.^2

Z =

   -2.00 -1.25 -1.00 -1.25 -2.00
   -1.25 -0.50 -0.25 -0.50 -1.25
   -1.00 -0.25      0 -0.25 -1.00
   -1.25 -0.50 -0.25 -0.50 -1.25
   -2.00 -1.25 -1.00 -1.25 -2.00

≫ surf(x, y, Z)
≫ colormap(gray)
```



Zadanie. Napisać funkcję `v = vnd(x)`, która na bazie wektora `x` o składowych $(x_0, \ldots, x_n)$ utworzy następującą macierz (tzw. *macierz Vandermonde'a*):

$$\begin{pmatrix} x_0^n & \cdot & \cdot & \cdot & x_0^2 & x_0 & 1 \\ x_1^n & \cdot & \cdot & \cdot & x_1^2 & x_1 & 1 \\ \vdots & \cdot & \cdot & \cdot & \vdots & \vdots & \vdots \\ x_n^n & \cdot & \cdot & \cdot & x_n^2 & x_n & 1 \end{pmatrix}$$

```
function v = vnd(x)
x = x(:); % x powinno byc kolumna
v = ones(size(x));
for i = 1: length(x) - 1
   v = [v(:, 1).*x v];
end;
```

```
≫ vnd([2 3 4 5])

ans =
     8     4     2     1
    27     9     3     1
    64    16     4     1
   125    25     5     1
```

Pętla *while* i rekurencja

Pętla nieskończona z instrukcjami `break`:

```
while 1
   ...
   if condition_1
      break;
   end;
   ...
   if condition_2
      break;
   end
end;
```

Zadanie. Napisać funkcję szukającą w wektorze `t` ostatniego wystąpienia zadanego elementu.

```
function i = szuk1(t, x)
i = 0;
for j = length(t): -1: 1
   if t(j) == x
      i = j;
```

```
      break;
   end
end;

function i = szuk2(t, x)
i = find(x == t);
if isempty(i)
   i = 0;
else
   i = i(end);
end;

function i = szuk3(t, x)
i = length(t);
while 1
   if i == 0
      break;
   end;
   if t(i) == x
      break;
   end;
   i = i - 1;
end;
```

```
≫ t = randperm(6000);
≫ tic; szuk1(t, 3000); toc
elapsed_time = 0.0400
≫ tic; szuk2(t, 3000); toc
elapsed_time = 0.0110
≫ tic; szuk3(t, 3000); toc
elapsed_time = 0.1000
```

Zadanie. Znaleźć pierwiastki równania

$$ax^3 + bx^2 + cx + d = 0.$$

Zastosować algorytm:

1. Znaleźć pierwiastek rzeczywisty x_1 metodą Newtona.
2. Podzielić trójmian przez $x - x_1$. (Łatwo pokazać, że odpowiedni iloraz ma postać $ax^2 + \beta x + \gamma$, gdzie: $\beta = ax_1 + b$, $\gamma = \beta x_1 + c$.)
3. Rozwiązać otrzymane równanie kwadratowe otrzymując pozostałe dwa pierwiastki.

```
function x1 = newton1(a, b, c, d)
epsi = 1.0e-5;
x1 = 0;
while 1
  fprim = (3*a*x1+2*b)*x1+c;
  if fprim == 0
    x1 = -b/a;
  else
    dx = -(((a*x1+b)*x1+c)*x1+d)...
         /fprim;
    x1 = x1 + dx;
    if abs(dx) < epsi
      break;
      end
  end
end;

function x = rown3(a, b, c, d)
if a == 0
  x = rown21(b, c, d);
else
  x = newton1(a, b, c, d);
  beta = b + a * x;
```

```
  gamma = c + beta * x;
  x = [x rown21(a, beta, gamma)];
end;
```

```
≫ rown3(2, -12, 22, -12)

ans =
    1.0000    3.0000    2.0000
```

Uwaga! W praktyce stosuje się funkcję `roots`:

```
≫ roots([2, -12, 22, -12])

ans =
    3.0000
    2.0000
    1.0000
```

Ogólny schemat rekurencji:

```
function y = p(n)
if n == 1
   y = wyrazenie_explicite;
else
   y = zwiazek_z( p(n - 1) );
end;
```

Zadanie. Napisać rekurencyjną wersję funkcji wyznaczającej wartość n^p, $(n, p) \in \mathbb{N}^2$.

```
function y = potega(n, p)
if p == 0
   y = 1;
elseif rem(p, 2) == 1
   y = n * potega(n*n, floor(p/2));
else
   y = potega(n*n, floor(p/2));
end;
```

Zadanie. Napisać rekurencyjną wersję funkcji szukającej w wektorze `t` ostatniego wystąpienia zadanego elementu.

```
function i = szuk4(t, x)
if isempty(t)
   i = 0;
elseif t(end) == x
   i = length(t);
else
   i = szuk4(t(1: end-1), x);
end;
```

Zadanie. Napisać funkcję obliczającą przybliżoną wartość całek adaptacyjną metodą Simpsona. Wykorzystać ją do obliczenia całki

$$\int_0^2 x^4 \ln(x + \sqrt{x^2+1})\,\mathrm{d}x$$

Rozwiązanie: Wzór Simpsona jest postaci

$$\int_\alpha^\beta f(x)\,\mathrm{d}x \approx I_S(f,\alpha,\beta)$$
$$= \left(f(\alpha) + 4f\left(\frac{\alpha+\beta}{2}\right) + f(\beta)\right)\frac{\beta-\alpha}{6}$$

Aby obliczyć całkę z dokładnością $\epsilon|b-a|$, dzielimy przedział $[a,b]$ na podprzedziały $[x_i, x_{i+1}]$ w taki sposób, aby na każdym z nich osiągnąć dokładność $\epsilon|x_{i+1}-x_i|$.

$\int_{x_i}^{x_{i+1}} f(x)\,\mathrm{d}x$ można obliczyć na dwa sposoby:

$$S_i = I_S(f, x_i, x_{i+1})$$
$$\bar{S}_i = I_S(f, x_i, x_{i+1/2}) + I_S(f, x_{i+1/2}, x_{i+1})$$

gdzie: $x_{i+1/2}$ – środek $[x_i, x_{i+1}]$.

Można udowodnić, że

$$I_i - \bar{S}_i \approx \frac{1}{15}(S_i - \bar{S}_i)$$

gdzie: I_i – dokładna wartość odpowiedniej całki.

Wynika stąd algorytm wyznaczania $I_{SA}(f, x_i, x_{i+1})$:
Oblicza się wartości

$$s = I_S(f, x_i, x_{i+1})$$
$$s_1 = I_S(f, x_i, x_{i+1/2})$$
$$s_2 = I_S(f, x_{i+1/2}, x_{i+1})$$

oraz „błąd” $|s_1 + s_2 - s|$. Jeżeli „błąd” jest większy niż $15\epsilon|x_{i+1} - x_i|$, dokonuje się podziału $[x_i, x_{i+1}]$ na podprzedziały $[x_i, x_{i+1/2}]$ i $[x_{i+1/2}, x_{i+1}]$, a następnie powtarza powyższe obliczenia na każdym z nich z osobna.

Implementacja: Funkcję podcałkową można zdefiniować dwoma sposobami:

- w pliku `fun.m`

```
function y = fun(x)
y = x.^4.*log(x+sqrt(x.*x+1));
```

- lub w linii poleceń:

```
≫ fun = inline(
'x.^4.*log(x+sqrt(x.*x+1))');
```

Moduł inicjujący rekurencję, zapisany w pliku simps.m, ma postać

```
function int = simps(f, przedz, epsi)
int = [];
a = przedz(1);
b = przedz(2);
if nargin < 3
   epsi = 0.5*sqrt(eps)/(b-a);
else
   epsi = 0.5*epsi/(b-a);
end;
przyb_pocz = (feval(f,a) ...
           +4*feval(f,0.5*(a+b)) ...
           +  feval(f,b))*(b-a)/6.0;
int = simps_rek(f, przedz, ...
                   przyb_pocz, epsi);
```

W tym samym pliku zapisuje się funkcję wewnętrzną `simps_rek` realizującą rekurencję.

```
% funkcja wewnetrzna
function intr = simps_rek(f, ...
            przedz, stare_przyb, epsi)
a = przedz(1);
b = przedz(2);
srod = (b+a)*0.5;
h = (srod-a)/6.0;
fsrod = feval(f,srod);
s1 =    (feval(f,a) ...
   + 4.0*feval(f,(a+srod)*0.5) ...
   +     fsrod)*h;
s2 =    (fsrod ...
   + 4.0*feval(f,(srod+b)*0.5) ...
   +     feval(f,b))*h;
err = abs(s1+s2-stare_przyb);
if err>=15.0*epsi*(b-a)
   s1 = simps_rek(f, [a, srod], ...
                           s1, epsi);
   s2 = simps_rek(f, [srod, b], ...
                           s2, epsi);
end;
intr = s1 + s2;
```

Zauważmy, że wywołanie `simps` zależy od sposobu zdefiniowania funkcji podcałkowej:

- z zastosowaniem pliku:

```
≫ simps('fun', [0 2], 1.0e-4)
ans =
    8.1534
```

- za zastosowaniem `inline`:

```
≫ simps(fun, [0 2], 1.0e-4)
ans =
    8.1534
```

Ciągi znaków i pliki tekstowe

Zadanie. Wyświetlić wszystkie znaki ASCII o kodach od 32 do 255, oprócz znaku o kodzie 127. Dlaczego nie wymaga się wyświetlenia pozostałych znaków?

```
≫ i = char(32:255);
≫ i(127) = [];
≫ i
```

Zadanie. Zinterpretować poniższe rezultaty:

```
≫ abs('zorro')

ans =
   122   111   114   114   111

≫ sin('zorro')

ans =
0.4987 -0.8646 0.7850 0.7850 -0.8646

≫ sin zorro

ans =
0.4987 -0.8646 0.7850 0.7850 -0.8646
```

Zadanie. Napisać funkcję określającą liczbę wystąpień danego znaku.

```
function lwyst = ile(s, x)
lwyst = sum(s == x);
```

Zadanie. Napisać funkcję określającą czy dane słowo jest palindromem.

```
function r = palindrom(s)
r = strcmp(s, s(end: -1: 1));
```

Zadanie. Napisać funkcję zamieniającą pierwsze litery słów danego ciągu znaków na duże litery.

```
function s1 = duze_lit(s)
s = [' ' s];
ind = findstr(s, ' ');
s = [s ' '];
s(ind+1) = upper(s(ind+1));
s1 = s(2: end-1);
```

```
≫ duze_lit('to jest  ciag')

ans = To Jest  Ciag
```

Zadanie. Napisać funkcję odwracającą kolejność słów w zdaniu.

```
function s1 = odwroc(s)
[pslowo reszta] = strtok(s);
if isempty(pslowo)
  s1 = s;
else
  s1 = [odwroc(reszta), ' '* ...
 ones(1,(length(reszta)>0)), pslowo];
end;
```

```
≫ odwroc('to jest jedno zdanie')

ans =
zdanie jedno jest to
```

Zadanie. Napisać funkcję zapisującą w pliku ciąg znaków oraz odpowiednią funkcję czytającą ten ciąg.

```
function zapisz(s, nazwa)
f = fopen(nazwa,'w');
fwrite(f, s);
fclose(f);

function s = czytaj(nazwa)
f = fopen(nazwa,'r');
s = fread(f);
fclose(f);
s = char(s');
```

Formatowane wyjście:

```
x = cumprod(1.0e-4*ones(5, 1));
a = [x, exp(x), exp(-x), ...
        (exp(x)- exp(-x))./(2*x)];
fprintf('%15s %15s %15s %15s\n', ...
  'x', 'exp(x)', 'exp(-x)', 'poch.');
fprintf( ...
'%15.4e %15.4e %15.4e %15.4e\n', a');
```

Rezultat:

```
              x          exp(x)
   1.0000e-004     1.0001e+000
   1.0000e-008     1.0000e+000
   1.0000e-012     1.0000e+000
   1.0000e-016     1.0000e+000
   1.0000e-020     1.0000e+000

      exp(-x)           poch.
   9.9990e-001     1.0000e+000
   1.0000e+000     1.0000e+000
   1.0000e+000     1.0000e+000
   1.0000e+000     5.5511e-001
   1.0000e+000     0.0000e+000
```

Błędy zaokrągleń

Zadanie. Wyjaśnić rezultat wykonania poniższego skryptu:

```
x = 1.0e+29;
y = 1.0e-9;
z = ((y + x) - x) / y % => 0
z = (y + (x - x)) / y % => 1
```

Co to jest `eps`?

```
≫ eps

   ans = 2.2204e-016

≫2^(-52)

   ans = 2.2204e-016

≫1 + eps == 1

   ans =  0

≫1 + eps/2 == 1

   ans =  1
```

Zadanie. Napisać program obliczający wartości e^x dla $x = -30$, -20, -10, 0, 10, 20, 30.
Wykorzystać w tym celu definicję

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

i porównać rezultaty z generowanymi przez `exp`.

```
function s = expo( x )
 t = 1.0; n = 1; s = 0.0;
 while 1
   if ( abs(t) <=  1.0e-14 )
     break;
   end;
   s = s + t;
   t = t * (x/n);
   n = n + 1;
 end
```

Skrypt:

```
for i = -3:3
 fprintf('%4d %10.3e %10.3e \n', ...
       10*i, exp(10*i), expo(10*i));
end
```

```
-30   9.358e-014   6.103e-006
-20   2.061e-009   6.148e-009
-10   4.540e-005   4.540e-005
  0   1.000e+000   1.000e+000
 10   2.203e+004   2.203e+004
 20   4.852e+008   4.852e+008
 30   1.069e+013   1.069e+013
```

Pytanie: Jak ograniczyć błędy zaokrągleń?

Odpowiedź: Ograniczyć obliczenia do przedziału $[0, 1]$:

$$e^x = \begin{cases} 1/e^{-x}, & x < 0 \\ e^{\mathrm{fix}(x)} e^{x-\mathrm{fix}(x)}, & x > 0 \end{cases}$$

Zadanie. Porównać wartości funkcji $f(x) = (1-x)^6$ w przedziale $[0.9999, 1.0001]$ z wartościami tej samej funkcji, ale liczonymi wg formuły

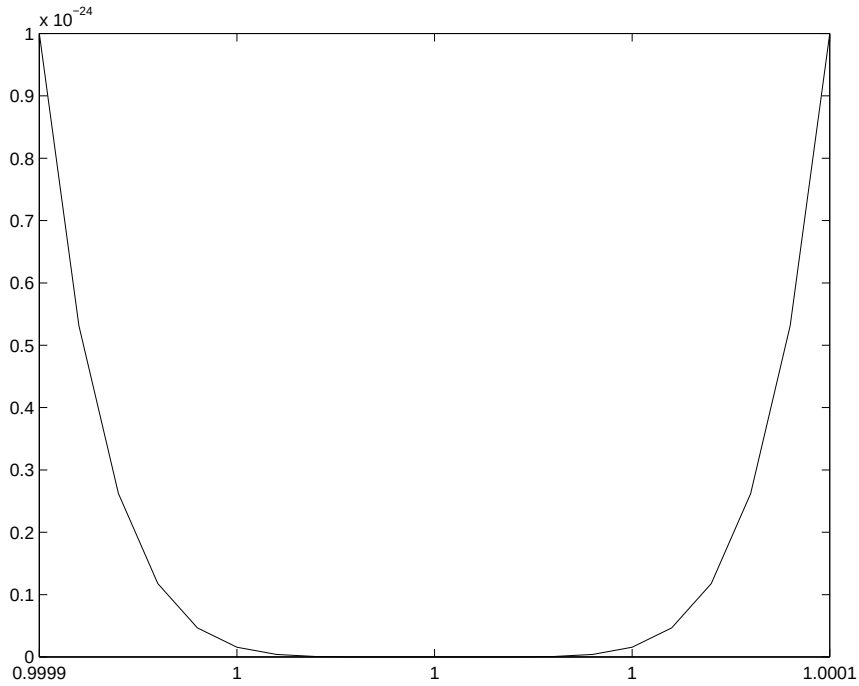
$f(x) = 1 - 6x + 15x^2 - 20x^3 + 15x^4 - 6x^5 + x^6$

```
function e = bin1(x)
x = 1-x;
e = x.*x.*x.*x.*x.*x;

function e = bin2(x)
x2 = x.*x;
x4 = x2.*x2;
e  = 1 - 6.*x + 15.*x2 ...
     - 20.*x2.*x + 15.*x4 ...
     - 6.*x4.*x + x4.*x2;
```

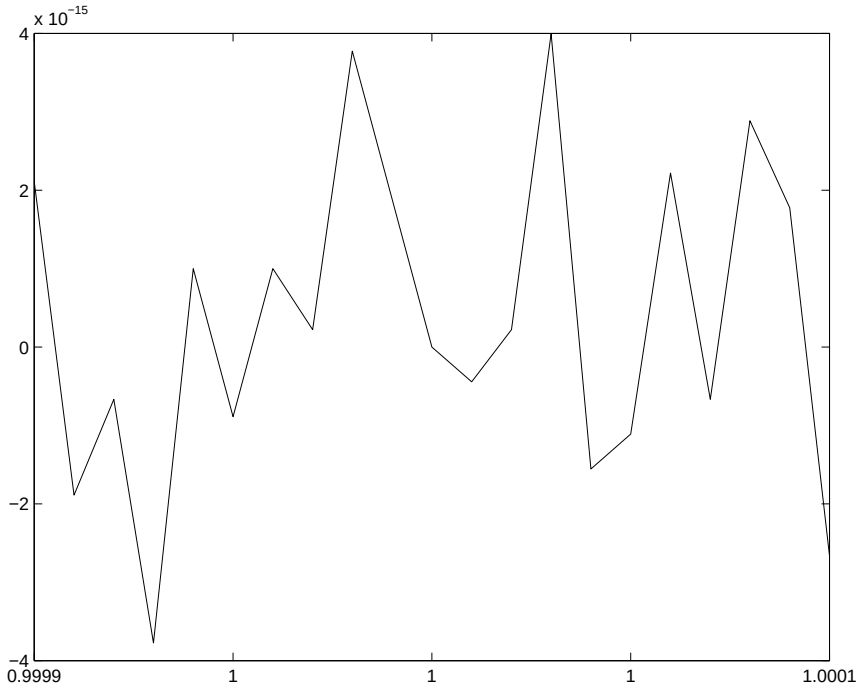
Skrypt:

```
poc = 99990;
kon = 100010;
for i=(poc: kon)/100000
fprintf('%15.5f %10.3e %10.3e\n', ...
          i, bin1(i), bin2(i));
end;
i=(poc: kon)/100000;
plot(i, bin1(i));
pause;
plot(i, bin2(i));
```

bin1.m:


x 10^-24
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0.9999
1
1
1
1.0001

bin2.m:


x 10^-15
4
2
0
−2
−4
0.9999
1
1
1
1.0001

Różniczkowanie numeryczne

Dla pierwszych pochodnych stosuje się

$$f'(a) \approx f[a, a+h] = \frac{f(a+h) - f(a)}{h}$$

$$E(f) = \frac{1}{2} h f''(\eta)$$

lub

$$f'(a) \approx f[a-h, a+h] = \frac{f(a+h) - f(a-h)}{2h}$$

$$E(f) = -\frac{1}{6} h^2 f'''(\eta)$$

Dla drugich pochodnych mamy:

$$f''(a) \approx \frac{f(a+h) - 2f(a) + f(a-h)}{h^2}$$

$$E(f) = -\frac{1}{12} h^2 f^{(\mathrm{iv})}(\eta)$$

```
for x = cumprod(0.1 * ones(1, 10))
fprintf('%2.0e %12.10f %14.10f \n', ...
  x, (exp(x) - exp(-x))/(2 * x), ...
  (exp(x) - 2 + exp(-x))/x^2); end;
```

Efekty są intrygujące:

```
1e-001 1.0016675002   1.0008336112
1e-002 1.0000166667   1.0000083334
1e-003 1.0000001667   1.0000000834
1e-004 1.0000000017   1.0000000050
1e-005 1.0000000000   0.9999989725
1e-006 1.0000000000   0.9999778783
1e-007 0.9999999995   0.9992007222
1e-008 0.9999999939   0.0000000000
1e-009 1.0000000272 111.0223024625
1e-010 1.0000000827   0.0000000000
```

Wyjaśnienie:

$$f'_{\mathrm{obl}}(a) = \frac{f(a+h) + E^{+} - f(a-h) - E^{-}}{2h} - \frac{1}{6}h^2 f'''(\eta)$$

$$= \frac{f(a+h) - f(a-h)}{2h} + \frac{E^{+} - E^{-}}{2h} - \frac{1}{6}h^2 f'''(\eta)$$

Całkowanie numeryczne

Zadanie. Z zastosowaniem funkcji `quad` lub `quad8` obliczyć całkę

$$\int_0^2 x^4 \ln(x + \sqrt{x^2+1})\,\mathrm{d}x$$

```
≫ f1 = inline(...
'x.^4.*log(x+sqrt(x.*x+1))');
≫ quad8(f1, 0, 2)

ans =
    8.1534
```

Zadanie. Z zastosowaniem funkcji `dblquad` obliczyć całkę

$$\int_0^5 \int_0^5 e^{-x^2-y^2}\,\mathrm{d}x\,\mathrm{d}y$$

```
function v = f2(x, y)
   v = exp(-x.*x-y.*y);
```

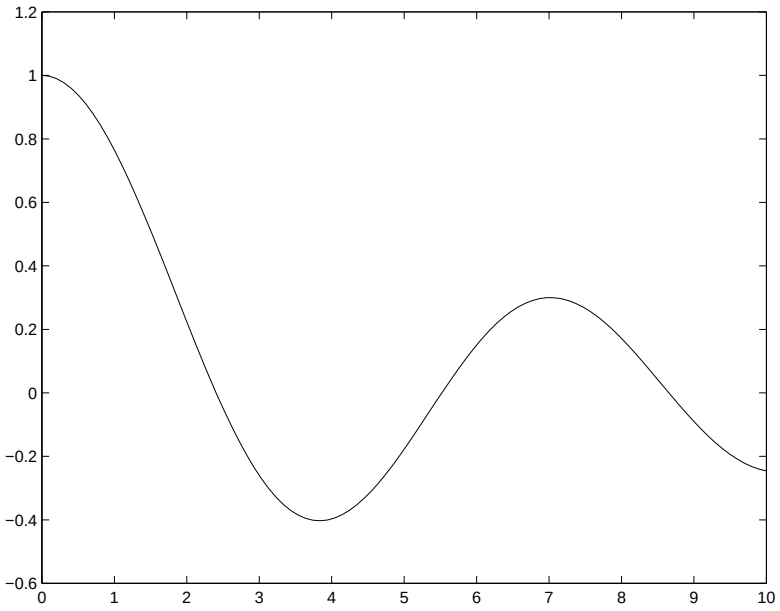
```
≫dblquad('f2', 0, 5, 0, 5, ...
[], [], 'quad8')
ans =
    0.7854
```

Zadanie. Narysować wykres funkcji

$$x \mapsto \frac{1}{\pi} \int_0^{\pi} \cos(x \sin u)\, \mathrm{d}u$$

w przedziale $[0, 10]$.

```
f = inline('cos(x.*sin (u))./pi', ...
                          'u', 'x');
x = linspace(0, 10, 100);
for i = 1: 100
   y(i) = quad(f, 0, pi, ...
                       [], [], x(i));
end;
plot(x, y);
```



Metoda Monte-Carlo

$\{X_p\}$ – ciąg niezależnych zmiennych losowych o jednakowym rozkładzie, $\mathrm{E}(|X|) < \infty$, $\Longrightarrow$

$$\frac{1}{N}\sum_{p=1}^{N} X_p \xrightarrow[N\to\infty]{\text{p.w.}} \mathrm{E}(X)$$

$X_p \sim$ jednostajny na hiperkostce $[0,1]^m$ $\Longrightarrow$

$$\frac{1}{N}\sum_{p=1}^{N} f(X_p) \xrightarrow[N\to\infty]{\text{p.w.}}$$

$$\int_0^1 \cdots \int_0^1 f(u_1,\ldots,u_m)\,\mathrm{d}u_1 \ldots \mathrm{d}u_m$$

$X_p \sim$ jednostajny na hiperkostce $\prod_{i=1}^{m} [a_i, b_i]$ o hiperobjętości $V \Longrightarrow$

$$\frac{1}{N} \sum_{p=1}^{N} f(X_p) \xrightarrow[N \to \infty]{\text{p.w.}}$$

$$\frac{1}{V} \int_{a_1}^{b_1} \cdots \int_{a_m}^{b_m} f(u_1, \ldots, u_m) \, \mathrm{d}u_1 \ldots \mathrm{d}u_m$$

Błąd można oszacować za pomocą odchylenia standardowego z próby $\{f(x_1), \ldots, f(x_N)\}$ (zob. `std`) pomnożonego przez $V/\sqrt{N}$.

Zadanie. Napisać funkcję generującą $m \times n$ macierz, której każdy wiersz zawiera próbę z rozkładu jednostajnego na danym przedziale.

```
function x = random(gran, n)
x = rand(size(gran,1), n);
dlug = gran(:, 2) - gran(:, 1);
x = gran(:, ones(1, n)) ...
    + x.*dlug(:, ones(1, n));
```

Zadanie. Napisać funkcję przybliżającą całkę wielokrotną po zadanej hiperkostce.

```
function [int, st] = mtc( ...
                     fun, gran, npkt);
if nargin <= 2
  npkt = 10000;
end;
obj = prod(gran(:,2)-gran(:,1));
z = feval(fun, random(gran, npkt));
if nargout == 2
  st = obj * std(z) / sqrt(npkt);
end;
int = obj * mean(z);
```

Zadanie. Obliczyć objętość przecięcia kul o środkach w $(0,0,0)$ i $(2,0,0)$ oraz promieniach odpowiednio 3 i 2.

```
function y = ff(x)
y = ((x(1,:)).^2   + x(2,:).^2 ...
      + x(3,:).^2 < 9)...
  & ((x(1,:)-2).^2 + x(2,:).^2 ...
      + x(3,:).^2 < 4);
```

```
≫[int, st] = mtc('ff', ...
[0 3; -2 2; -2 2], 100000)

int =
   24.6931
st =
    0.0759
```

Równania różniczkowe

$$\frac{\mathrm{d}y}{\mathrm{d}x} = f(x, y), \quad y(x_0) = y_0$$

gdzie: $x \in \mathbb{R}$, $y = (y_1, \ldots, y_n)$, $f : \mathbb{R}^{n+1} \to \mathbb{R}^n$

Metoda Eulera

$$y_{m+1} = y_m + hf(x_m, y_m) + O(h^2)$$

gdzie: $x_{m+1} = x_m + h$

Metoda RK2

$$k_1 = hf(x_m, y_m)$$
$$k_2 = hf(x_m + \frac{1}{2}h, y_m + \frac{1}{2}k_1)$$
$$y_{m+1} = y_m + k_2 + O(h^3)$$

Metoda RK4

$$k_1 = hf(x_m, y_m)$$
$$k_2 = hf(x_m + \frac{1}{2}h, y_m + \frac{1}{2}k_1)$$
$$k_3 = hf(x_m + \frac{1}{2}h, y_m + \frac{1}{2}k_2)$$
$$k_4 = hf(x_m + h, y_m + k_3)$$
$$y_{m+1} = y_m + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) + O(h^5)$$

Zadanie. Zaimplementować wymienione metody.

```
function yh = euler(f, x, y, h)
yh = y + h * feval(f, x, y);
```

```
function yh = rk2(f, x, y, h)
k1 = h * feval(f, x, y);
k2 = h * feval(f, x + h/2, y + k1/2);
yh = y + k2;
```

```
function yh = rk4(f, x, y, h)
k1 = h * feval(f, x, y);
k2 = h * feval(f, x + h/2, y + k1/2);
k3 = h * feval(f, x + h/2, y + k2/2);
k4 = h * feval(f, x + h, y +  k3  );
yh = y + (k1 + 2*k2 + 2*k3 + k4)/6.0;
```

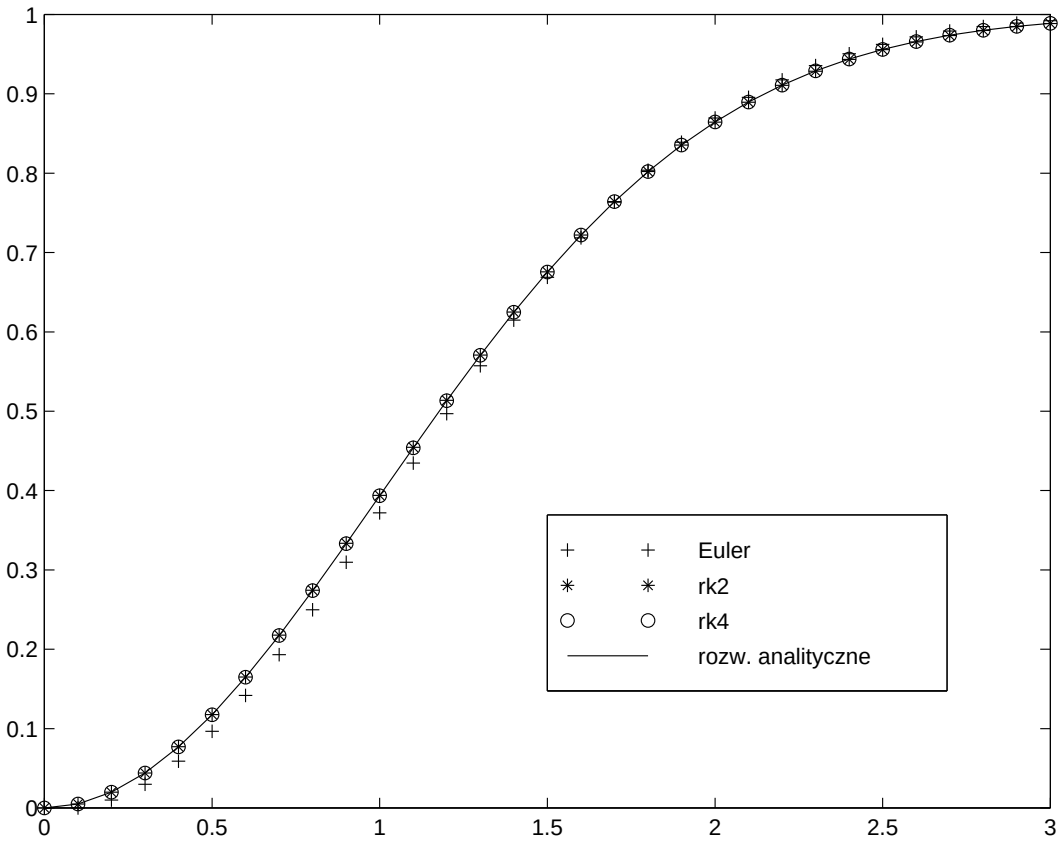
Zadanie. Porównać przybliżone rozwiązania zagadnienia

$$\frac{\mathrm{d}y}{\mathrm{d}x}(x) + xy(x) = x, \quad y(0) = 0$$

z rozwiązaniem dokładnym $y(x) = 1 + e^{-x^2/2}$ dla `x = 0: 0.1: 3`.

```
f = inline('x.*(1-y)', 'x', 'y');
h = 0.1;
x = 0: h: 3;
ye = 0;
```

```
for i = x(1: end-1)
   ye = [ye euler(f, i, ye(end), h)];
end;
y2 = 0;
for i = x(1: end-1)
   y2 = [y2 rk2(f, i, y2(end),h)];
end;
y4 = 0;
for i = x(1: end-1)
   y4 = [y4 rk4(f, i, y4(end), h)];
end;
delete(gcf)
plot(x, ye, 'r+')
hold on
plot(x, y2, 'b*')
plot(x, y4, 'mo')
sol = inline('1-exp(-x.*x/2)');
plot(x, sol(x), 'y-');
legend('Euler', 'rk2', 'rk4', ...
       'rozw. analityczne', 0);
```



Zadanie. Powtórzyć poprzednie zadanie dla problemu

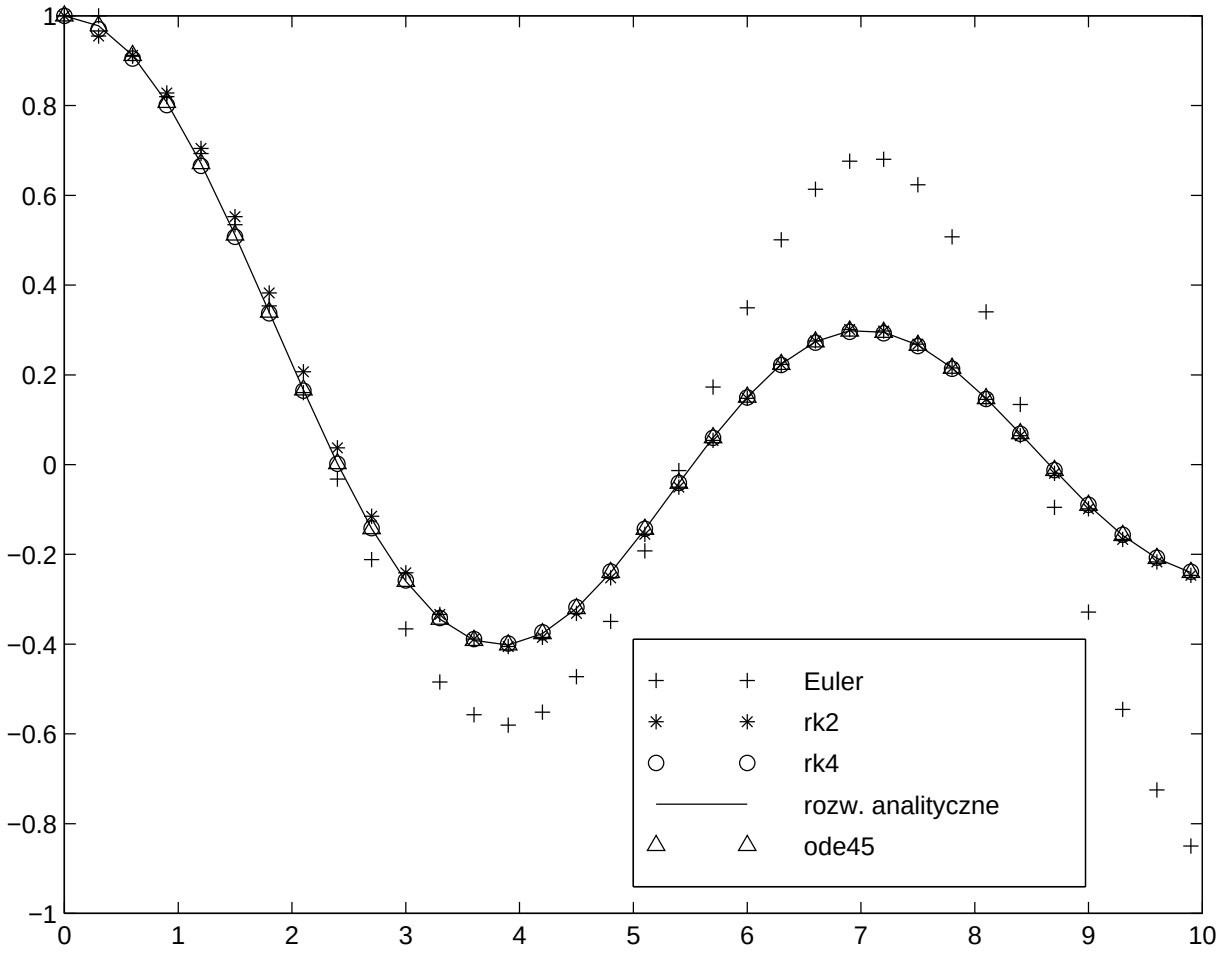
$$x\frac{\mathrm{d}^2y}{\mathrm{d}x^2}(x) + \frac{\mathrm{d}y}{\mathrm{d}x}(x) + xy(x) = 0,$$

$$y(0) = 1, \quad \frac{\mathrm{d}y}{\mathrm{d}x}(0) = 0$$

oraz `x = 0: 0.3: 10`.

```
f = inline(...
   '[-y(1)/(x+eps)-y(2); y(1)]', ...
        'x', 'y');
h = 0.3;
x = 0: h: 10;
ye = [0; 1];
for i = x(1: end-1)
  ye = [ye euler(f, i, ye(:,end), h)];
end;
y2 = [0; 1];
for i = x(1: end-1)
  y2 = [y2 rk2(f, i, y2(:, end), h)];
end;
y4 = [0; 1];
for i = x(1: end-1)
  y4 = [y4 rk4(f, i, y4(:, end), h)];
end;
delete(gcf)
plot(x, ye(2, :), 'r+')
hold on
plot(x, y2(2, :), 'b*')
plot(x, y4(2, :), 'mo')
plot(x, besselj(0, x), 'y-')
```

```
y45 = [0; 1];
[x y45]=ode45(f, x, y45);
plot(x, y45(:, 2), 'g^')
legend('Euler', 'rk2', 'rk4',...
    'rozw. analityczne', 'ode45', 0);
```



Zadanie. W zbiorniku umieszczono p_0 moli Cl_2 i 1 mol benzenu. Oznaczmy odpowiednio przez p, q, r, s, i t ilości Cl_2, benzenu, chlorobenzenu, dwuchlorobenzenu i trójchlorobenzenu w chwili θ. Dla objętości v produktu obowiązują równania

$$-k_1pq = v\frac{\mathrm{d}p}{\mathrm{d}\theta}, \quad k_1pq - k_3pr = v\frac{\mathrm{d}q}{\mathrm{d}\theta},$$
$$k_2pr - k_3ps = v\frac{\mathrm{d}s}{\mathrm{d}\theta}, \qquad k_3ps = v\frac{\mathrm{d}t}{\mathrm{d}\theta}$$

gdzie: $k_1 = 8k_2$, $k_2 = 30k_3$. Można przyjąć $v = 1$ i $k_3 = 1$.

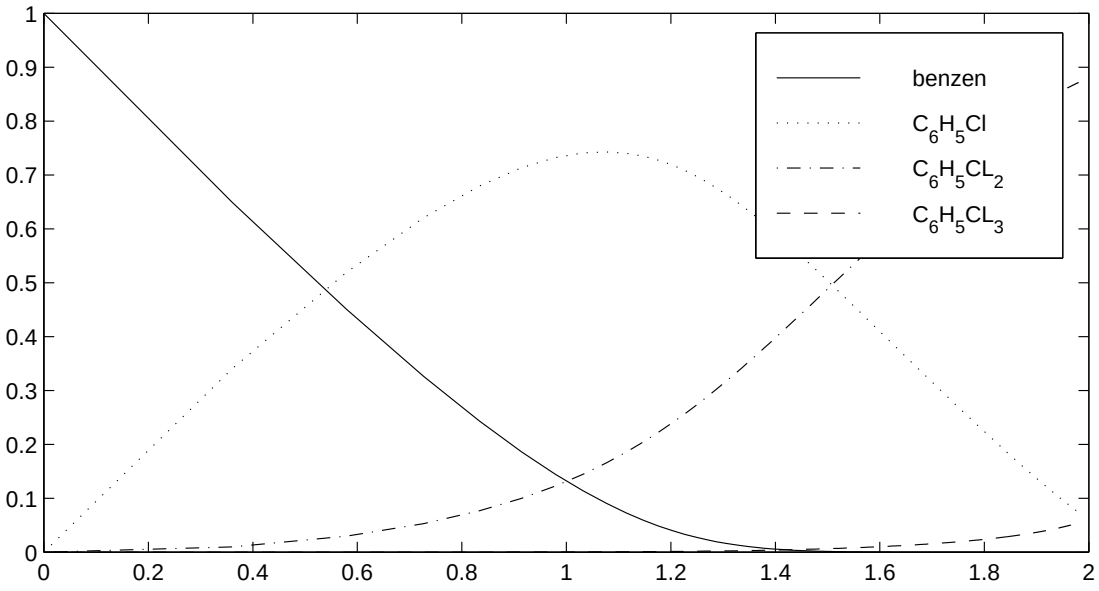
Zwróćmy uwagę, że ostatnie równanie różniczkowe nie jest istotne, bo zachodzi $q + r + s + t = 1$. Zużywana ilość Cl_2 wynosi $z = r + 2s + 3t$.

- Narysować ewolucję q, r, s i t w funkcji z dla $0 \leq z \leq 2$.
- Dla jakiej wartości q otrzymuje się maksimum chlorobenzenu?
- Jaką wartość Cl_2 na mol benzenu należy wprowadzić do zbiornika, aby osiągnąć to maksimum?

```
%obliczenia
opcje=odeset('abstol', 1.0e-4, ...
        'reltol', 1.0e-4);
[t y]=ode45('tricl', ...
  [0:0.001:0.7], [1 0 0 ], opcje);
q = y(:,1);
r = y(:,2);
s = y(:,3);
t = 1-y(:,1)-y(:,2)-y(:,3);
p = r+2*s+3*t;
[m i] = max(r);

%prezentacja
delete(gcf);
figure('name', ...
   'Trojchlorowanie benzenu', ...
   'number', 'off', 'Menu', 'none');
subplot('Position', ...
   [0.1 0.4 0.8 0.55]);
plot(p, q, 'y-',p, r, 'r:',p, ...
           s, 'b-.', p, t, 'g--')
legend('benzen', 'C_6H_5Cl', ...
       'C_6H_5CL_2', 'C_6H_5CL_3');
```

```
subplot('Position',...
   [0.1 0.01 0.8 0.35]);
axis off
st{1} = sprintf(['Maksimum ', ...
     'C_6H_5Cl wynosi %6.4f', ...
     ' i odpowiada'], m);
st{2} = sprintf(...
    ['koncentracji C_6H_6', ...
      ' rownej %6.4f.'], q(i));
st{3} = sprintf(['Ilosc ', ...
    'chloru na mol C_6H_6', ...
    ' potrzebna do jego', ...
    ' otrzymania']);
st{4} = sprintf(...
       'wynosi %6.4f.', p(i));
text(0, 0.55, st, 'fontsize', 14)
```

```
function yp = tricl(theta,y)
k = [ 240 30 1]; p0 = 2; v = 1;
p = p0 -y(2)-2*y(3)...
    -3*(1-y(1)-y(2)-y(3));
q = y(1); r = y(2); s = y(3);
yp = [ -k(1)*p*q;
       (k(1)*q - k(2)*r)*p;
       (k(2)*r -k(3)*s)*p;
     ]/v;
```



Maksimum C_6H_5Cl wynosi 0.7430 i odpowiada koncentracji C_6H_6 rownej 0.0907.
Ilosc chloru na mol C_6H_6 potrzebna do jego otrzymania wynosi 1.0762.

Wielomiany, aproksymacja i interpolacja

```
w1 = poly(1:20);
r1 = roots(w1);
w2 = w1;
w2(5) = w2(5)+1;
r2 = roots(w2);
```

`r1` ma postać:

```
    1.0000    2.0000    3.0000
    4.0000    5.0000    6.0000
    6.9999    8.0003    8.9988
   10.0028   10.9966   11.9980
   13.0182   13.9612   15.0535
   15.9512   17.0290   17.9878
   19.0029   19.9997
```

`r2` ma postać:

```
   1.0000     2.0000     3.0000
   4.0000     5.0050     5.8427
   6.5881-0.8857i   6.5881+0.8857i
   7.6787-2.2021i   7.6787+2.2021i
   9.1896-3.8170i   9.1896+3.8170i
```

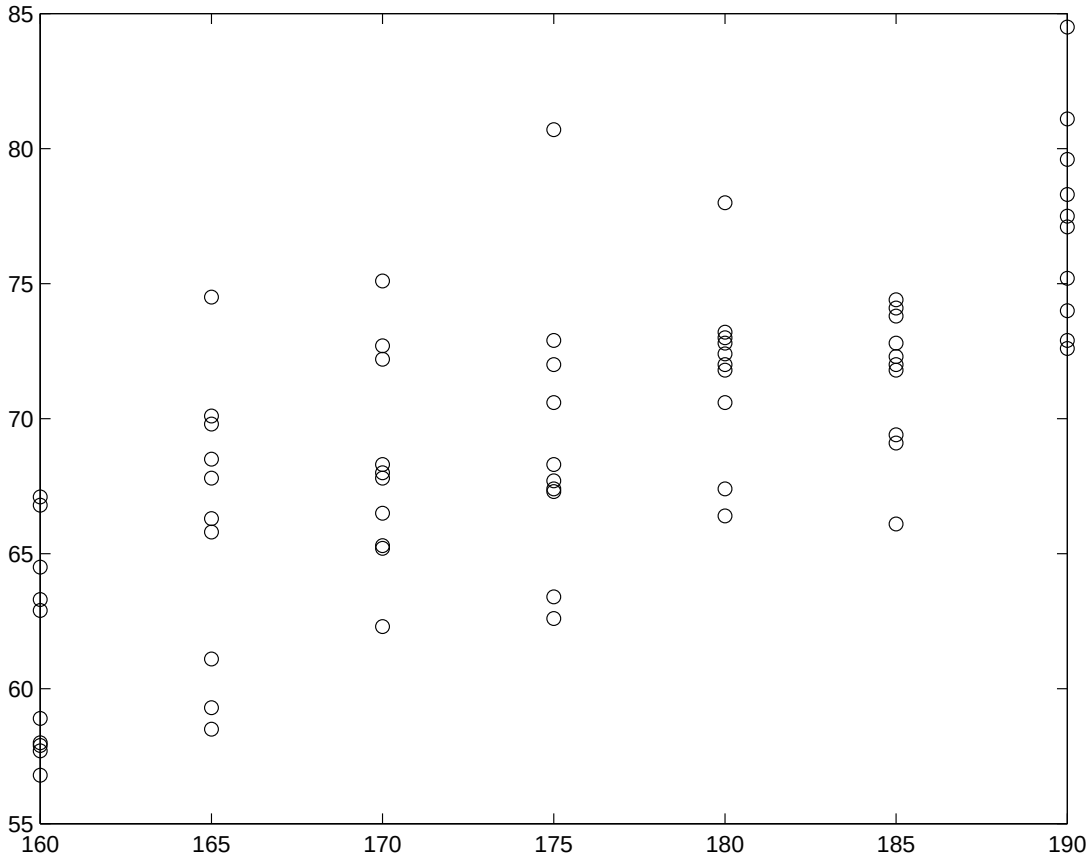
```
  11.5097-5.6414i  11.5097+5.6414i
  15.1822-7.0961i  15.1822+7.0961i
  20.1572-6.6537i  20.1572+6.6537i
  24.2707-2.8471i  24.2707+2.8471i
```

Stopień wielomianu:

```
function [n, p] = degree(p, tol)

if nargin == 1
   tol = 0.0;
end;

m = max(abs(p));
if m == 0.0
   n = -inf;
else
   v = find(abs(p)>tol*m);
   if isempty(v)
      n =  -inf;
   else
      n = length(p)- min(v);
   end;
end;
```

```
if nargout == 2
   if isinf(n)
      p = [];
   else
      p = p(length(p)-n:length(p));
   end;
end;
```

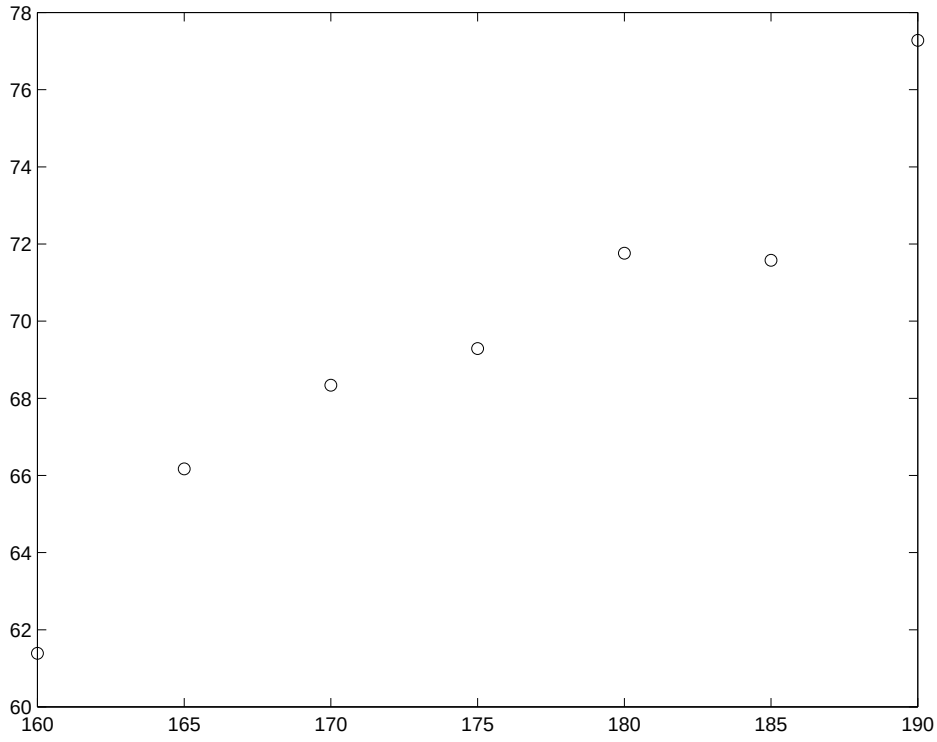
Zadanie. Dopasować prostą do zestawu par (wzrost, waga) dla danych 70 osób.



```
[wzrosty, i] = sort(wzrosty);
wagi = wagi(i);
plot(wzrosty, wagi, 'o')
```

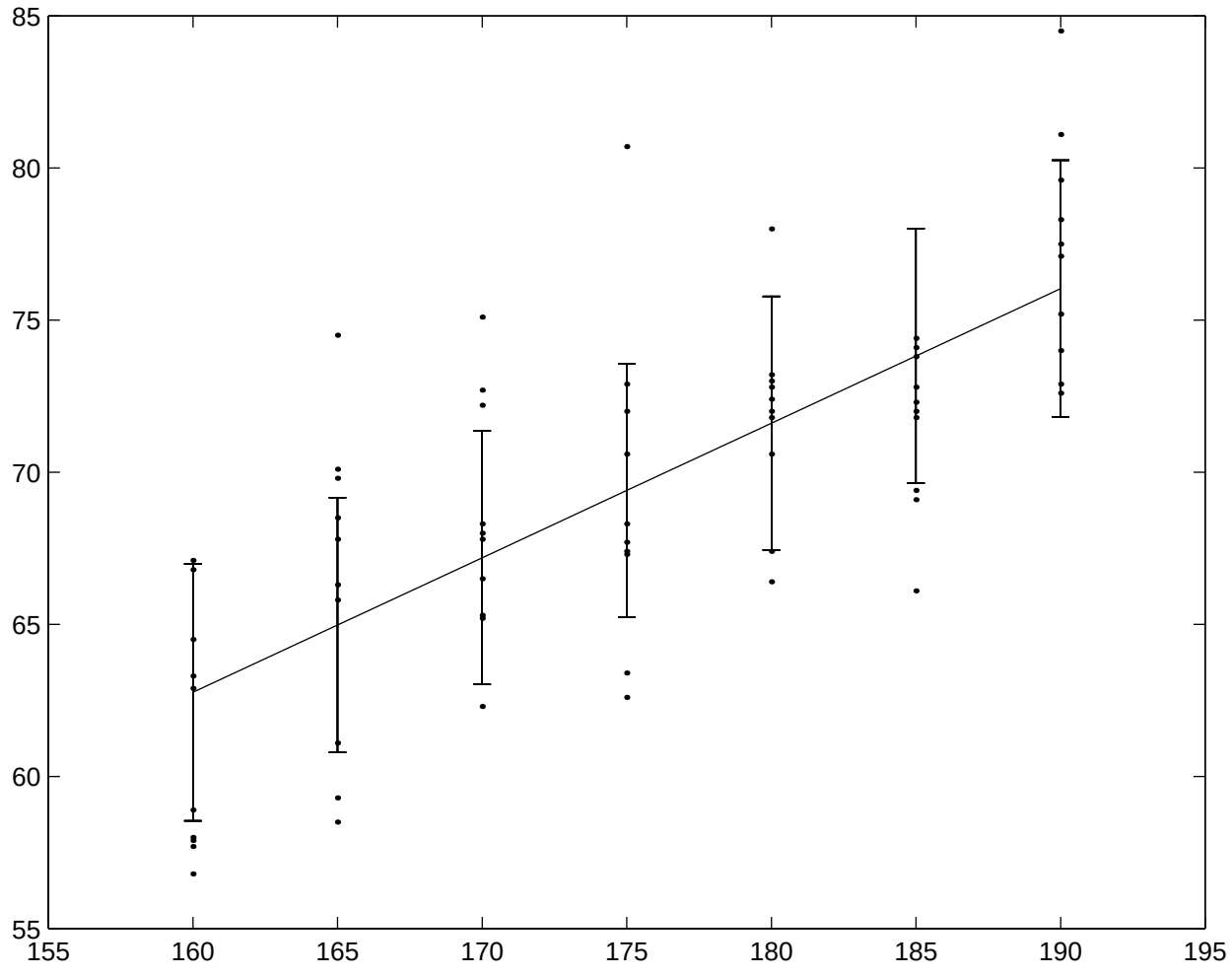
Dla każdego wzrostu można najpierw wyznaczyć średnią wagę:

```
rwzrosty = wzrosty(...
      logical([1 diff(wzrosty)~=0]));
for i = 1: length(rwzrosty)
   swagi(i) = mean( ...
      wagi(wzrosty == rwzrosty(i)));
end;
plot(rwzrosty, swagi, 'o')
```



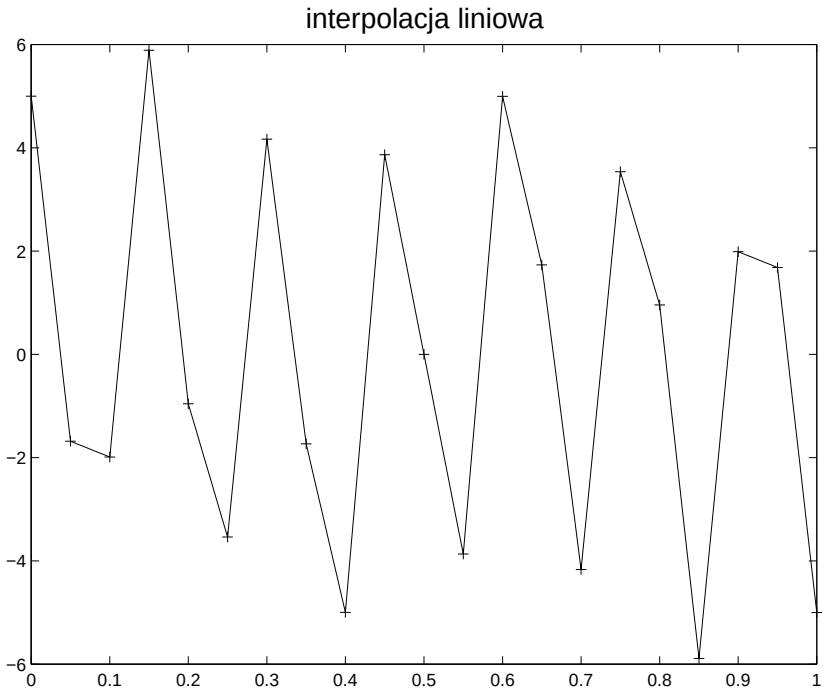
Metoda najmniejszych kwadratów:

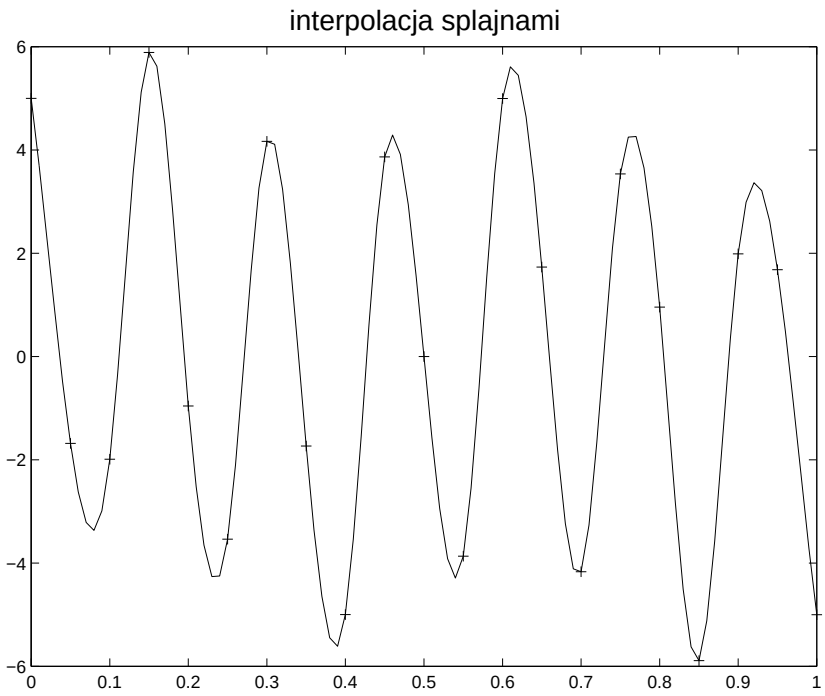
```
[p s] = polyfit(wzrosty, wagi, 1);
[awagi delta] = polyval(...
                    p, wzrosty, s);
plot(wzrosty, wagi, '.')
hold on
errorbar(wzrosty, awagi, delta)
```



Interpolacja:

```
x = 0: 0.05: 1;
y = sin(4.0.*x.*pi) ...
  + 5.0.*cos(13.0.*x.*pi);
plot(x,y, '+');
xi = 0: 0.01: 1;
yi = interp1(x,y,xi,'linear');
plot(x,y, '+');
hold on
plot(xi,yi);
title('interpolacja liniowa', ...
                  'fontsize', 16);
print -deps interp11.eps
hold off
yi = interp1(x,y,xi,'spline');
plot(x,y, '+');
hold on
plot(xi,yi);
title('interpolacja splajnami', ...
                  'fontsize', 16);
print -deps interp12.eps
```





Inne funkcje: `polyvalm`, `polyder`, `residue`, `conv`, `deconv`, `interpft`.

Układy równań liniowych

Jak rozwiązać układ $Ax = b$?

```
≫A = rand(3, 3)

A =

    0.2190    0.6793    0.5194
    0.0470    0.9347    0.8310
    0.6789    0.3835    0.0346

≫b = rand(3, 1)

b =

    0.0535
    0.5297
    0.6711

≫x = A \ b

x =

 -159.3380
  314.8625
 -344.5078
```

Jak sprawdzić poprawność rozwiązania?

```
≫A * x - b

ans =
   1.0e-13 *
   -0.2602
   -0.1732
   -0.0322

≫norm(A * x - b)

ans =
  1.6435e-014
```

Rozkład LU

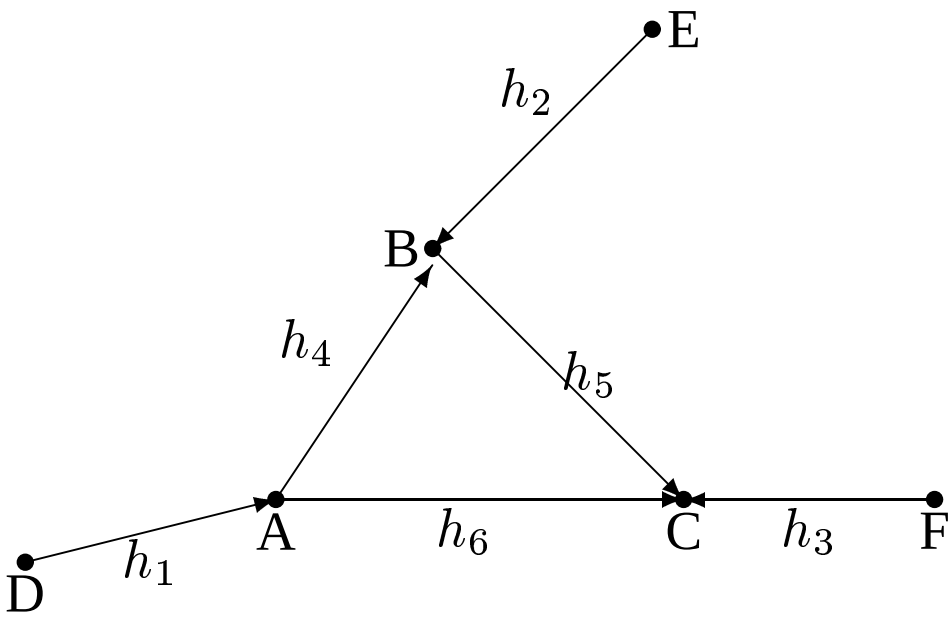
Funkcja `lu` dokonuje rozkładu macierzy $A \in \mathbb{R}^{n \times n}$ postaci $PA = LU$, gdzie: $L \in \mathbb{R}^{n \times n}$ – macierz trójkątna dolna, $U \in \mathbb{R}^{n \times n}$ – macierz trójkątna górna, $P \in \mathbb{R}^{n \times n}$ – macierz permutacji.

```
≫[L, U, P] = lu(A)

L =
    1.0000         0         0
    0.0692    1.0000         0
    0.3226    0.6118    1.0000
```

```
U =
    0.6789    0.3835    0.0346
         0    0.9082    0.8286
         0         0    0.0013

P =
     0     0     1
     0     1     0
     1     0     0

≫x = U \ (L \ (P' * b))

x =
 -161.5276
  319.2097
 -349.2705
```

Układy liniowe nadokreślone

Zadanie. Aby oszacować wysokość trzech punktów A, B i C nad poziomem morza, zmierzono różnice wysokości zgodnie z poniższym rysunkiem. Punkty D, E i F leżą na poziomie morza.



Każdy pomiar daje związek liniowy dla wysokości x_A, x_B i x_C punktów A, B i C:

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \\ -1 & 0 & 1 \end{bmatrix}}_{F} \underbrace{\begin{bmatrix} x_A \\ x_B \\ x_C \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} 1 \\ 2 \\ 3 \\ 1 \\ 2 \\ 1 \end{bmatrix}}_{p}$$

Jak „rozwiązać” powyższy układ równan?

```
function [x, odl] = mnk(A, b)
[m, n] = size(A);
if (m <= n)
   error(['Uklad nie jest', ...
          ' nadokreslony'])
end
if (rank(A) < n)
   error(['Macierz musi byc', ...
          ' pelnego rzedu'])
end
H = chol(A' * A);
x = H \ (H' \ (A' * b));
r = b - A * x;
odl = norm(r);
```

```
≫ [x, r] = mnk(F, p)

x =

    1.2500
    1.7500
    3.0000

r =

    1.2247
```

```
≫x = F \ p

x = 1.2500
    1.7500
    3.0000
```

Macierze rzadkie

Macierz $A \in \mathbb{R}^{5000 \times 5000}$ wymaga alokacji pamięci dla 25 milionów liczb `double`, nawet wtedy gdy tylko 50 000 z nich jest niezerowych. Te same 50 000 niezerowych niezerowych elementów można jednak zapamiętać z zastosowaniem 50,000 liczb `double` i 50,000 par indeksów całkowitych, czyli mniej niż 0.5% pamięci ($<$ 1 MB zamiast 200 MB). Podobnie, rozwiązanie układu $Ax = b$ zajęłoby większość dnia, a w zastosowaniem techniki `sparse` zajmuje mniej niż poł minuty!

```
≫A = [0 0 1;1 0 2;0 -3 0]

A =
     0     0     1
     1     0     2
     0    -3     0
```

```
≫S = sparse(A)

S =
   (2,1)        1
   (3,2)       -3
   (1,3)        1
   (2,3)        2

≫whos
Name  Size  Bytes  Class

A     3x3      72  double array
S     3x3      64  sparse array

Grand total is 13 elements
             using 136 bytes
```

Praktyczniejszy sposób definiowania:

```
≫A = sparse(3,2)

A =
   All zero sparse: 3-by-2

≫A(1,2)=1;
≫A(3,1)=4;
```

```
≫ A(3,2)=-1;
≫ A

A =
    (3,1)         4
    (1,2)         1
    (3,2)        -1
```

Inna wersja:

```
S = sparse(I,J,S,m,n,maxnz).
```

Zadanie. Jak efektywniej zapamiętać poniższą macierz?

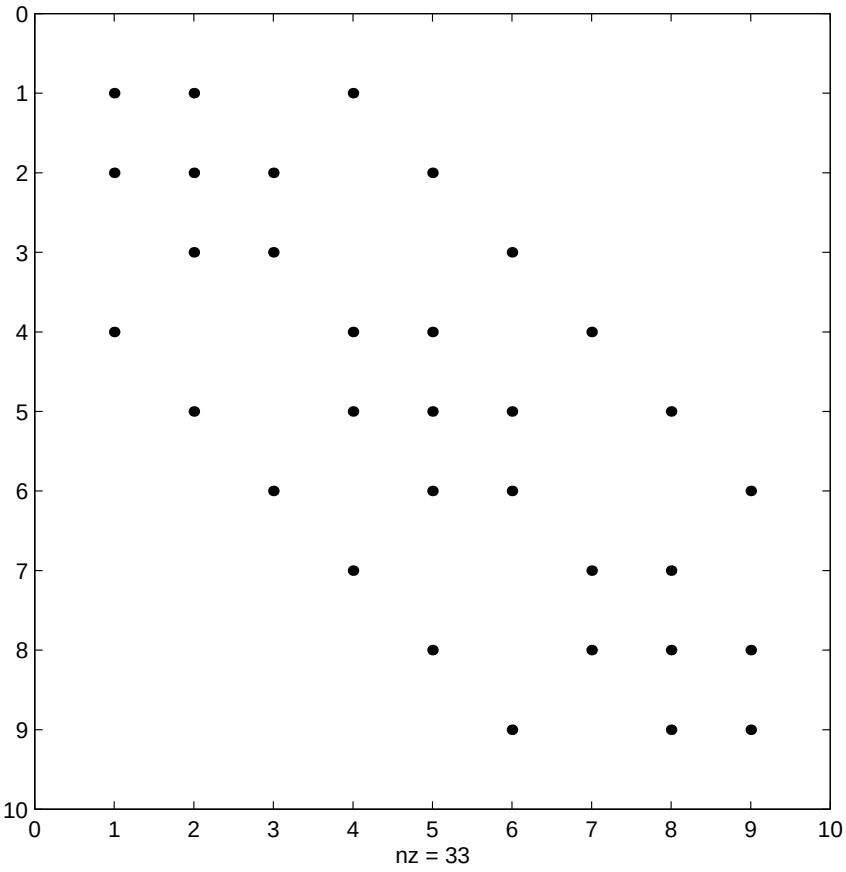
```
≫ A

A =
 64 -16   0 -16   0   0   0   0   0
-16  64 -16   0 -16   0   0   0   0
  0 -16  64   0   0 -16   0   0   0
-16   0   0  64 -16   0 -16   0   0
  0 -16   0 -16  64 -16   0 -16   0
  0   0 -16   0 -16  64   0   0 -16
  0   0   0 -16   0   0  64 -16   0
  0   0   0   0 -16   0 -16  64 -16
  0   0   0   0   0 -16   0 -16  64
```

Rozwiązanie:

```
≫ B = [
    -16   -16    64     0     0
    -16   -16    64   -16     0
    -16     0    64   -16     0
    -16   -16    64     0   -16
    -16   -16    64   -16   -16
    -16     0    64   -16   -16
      0   -16    64     0   -16
      0   -16    64   -16   -16
      0     0    64   -16   -16
];
≫ d = [-3,-1,0,1,3];
≫ S = spdiags(B,d,9,9);
```

W celu sprawdzenia, można wywołać funkcję `spy`:

```
≫ spy(A)
```



Macierz jednostkowa:

```
≫ I = speye(n)
```

Struktury

Rozważmy funkcję $f(x) = (x_1 - 1)^2 + x_1 x_2$.

```
function fx = f(x)
fx.wartosc = (x(1)-1)^2+x(1)*x(2);
fx.gradient = [2*(x(1)-1)+x(2);x(1)];
fx.hesjan = [2 1;1 0];
```

```
>> x = [2; 1]

x =

     2
     1

>> fx = f(x)

fx =

     wartosc: 3
    gradient: [2x1 double]
      hesjan: [2x2 double]

>> whos

Name  Size  Bytes  Class

fx     1x1    428  struct array
x      2x1     16  double array

Grand total is 12 elements
             using 444 bytes
```

Tablice wielowymiarowe:

```
>> gx.wartosc = 12;
>> gx.gradient = [2; 1];
```

```
≫ A(1,1) = fx;
≫ A(2,1) = gx;
??? Subscripted assignment between
                dissimilar structures.
≫ fieldnames(fx)
ans =
    'wartosc'
    'gradient'
    'hesjan'
≫ fieldnames(gx)
ans =
    'wartosc'
    'gradient'
≫ help struct
```

Tablice komórkowe

Zadanie. Chcemy pamiętać imię i nazwisko osoby oraz jej numer telefonu.

Dwa sposoby definicji:

❶
```
≫A(1,1) = {'Jan Kowalski'};
≫A(1,2) = {[1 2 3 4 5 6 7 8
9]};
```

❷
```
≫B{1,1} = 'Jan Kowalski';
≫B{1,2} = [1 2 3 4 5 6 7 8 9];
```

```
≫A

A =
    'Jan Kowalski'    [1x9 double]

≫celldisp(A)

A{1} =
Jan Kowalski


A{2} =
     1  2  3  4  5  6  7  8  9

≫B{1,1}

ans =
Jan Kowalski
```

Jak usunąć element tablicy?

```
≫ B(1) = []

B =
     [1x9 double]

≫ C = {A B}

C =
    {1x2 cell}    {1x1 cell}

≫ celldisp(C)

C{1}{1} =
Jan Kowalski

C{1}{2} =
     1  2  3  4  5  6  7  8  9


C{2}{1} =
     1  2  3  4  5  6  7  8  9

≫ help cell
```

Pytanie: Jak usunąć `C(2,1)`?

Zadanie. Napisać funkcję generującą liczby losowe o rozkładzie $\mathcal{N}(m, \sigma)$.

```
function x = normal(m, sig, varargin)
if nargin <= 2
   x = randn * sig + m;
else
   x = m + randn([varargin{:}])*sig;
end;
```

Możliwe wywołania: `normal(1,2,3)`, `normal(1,2,[3,5])`, `normal(1,2,3,5)`.

Konstrukcja `switch-case`

```
x = ceil(10*rand);
switch x
   case {1,2}
      disp('Prawdopodob. = 20%');
   case {3,4,5}
      disp('Prawdopodob. = 30%');
   otherwise
      disp('Prawdopodob. = 50%');
end
```