

Scilab - podstawy

Scilab jest środowiskiem numerycznym, programistycznym i numerycznym dostępnym za darmo z INRIA (*Institut Nationale de Recherche en Informatique et Automatique*). Jest programem podobnym do MATLABa oraz jego darmowego 'klonu' OCTAVE'a.

Scilab jest samodzielnym programem zawierającym wiele wbudowanych funkcji numerycznych oraz graficznych. Jest wyposażony w język programowania.

Wersje instalacyjne programu Scilab mogą zostać pobrane ze strony <http://www.scilab.org>. Również na tej stronie znajdują się linki do dokumentacji. Wpisując w wyszukiwarce na przykład słowa "Scilab tutorial" można znaleźć linki do różnego rodzaju podręczników i wykładów wprowadzających do Scilaba.

Zmienne:

W programie operuje się na zmiennych. Nadawanie im wartości odbywa się poprzez *podstawienie*:

```
a=2.5
b=6.3
c=a+b
a+b
a*b
a/b
a^b
```

Umieszczenie średnika na końcu polecenia sprawia, że wyliczona wartość (lub wartości) **nie jest wyświetlana**.

Stałe specjalne:

`%i` - wartość urojona równa $\sqrt{-1}$

`%pi` - π

`%e` - podstawa logarytmu naturalnego

`%eps` - największa wartość, dla której $1 + \%eps = 1$

`%inf` - nieskończoność (komputerowa)

`%t`, `%f` - zmienne logiczne o wartościach prawda (true) i fałsz (f) (Co wyświetli się po wypisaniu wyrażenia logicznego $2 > 5$?)

Help - uzyskiwanie pomocy

- `help` polecenie, np. `help sin`
- `apropos` polecenie wyświetla informacje związane z danym poleceniem
- Na stronie <http://www.scilab.org/product/man>

Zmienne- kontynuacja

```
a=2
b=3
save ('val.dat',a) // zapisanie zmiennej a w pliku 'val.dat'
clear a // skasowanie zmiennej a
a // widać, że zmienna a została zniszczona
b
load ('val.dat','a') // ponowne załadowanie zmiennej a
```

```
a
exp(a)+exp(b)
sin(a*pi/b)
```

Wektory

Program Scilab jest programem wektorowym, większość operacji jest wykonywanych w odniesieniu do wektorów. Istotne jest rozróżnienie wektora wierszowego i kolumnowego.

Wektor wierszowy - elementy oddzielone spacją lub przecinkiem:

```
v=[1 2 3 4]
albo
v=[1, 2, 3, 4]
```

Wektor kolumnowy - elementy oddzielone średnikiem lub pisane od nowego wiersza:

```
r=[1; 2; 3; 4]
r=[1
2
3
4]
```

Uwaga: W przypadku dużych zestawów wartości należy umieszczać średnik na końcu polecenia.

Transpozycja - czyli zamiana wektora wierszowego na kolumnowy, i odwrotnie:

```
vt = v'
rt = r'
vt'
y=[1 2 3]'
```

Wektor można zdefiniować zadając wartość pierwszego elementu, krok oraz wartość ostatniego elementu, oddzielając je dwukropkiem:

```
x=[-10:0.5:10]
```

Zwróćmy uwagę na różnicę:

```
x=[-10:0.5:10]'
```

Z kolei polecenie:

```
z=linspace(-10, 10, 101);
```

wygeneruje wektor o 101 elementach, przyjmujących wartości z przedziału [-10, 10].

Można zrobić tak:

```
x=[-1:0.1: 1]'
```

```
y=sin(x)
```

A następnie naszkicować wykres:

```
plot2d(x, y)
```

Na wektorach o zgodnych wymiarach (co to oznacza?) **można wykonywać różne operacje:**

```
x+y
x-y
x*y
x*y'
x'*y
```

Proszę porównać różnice w trzech ostatnich poleceniach

Można odwoływać się do *elementów* wektorów, np:

```
x(1)
x(10)
```

```
y(2)+x(3)
```

Wektory można konstruować w oparciu o zmienne:

```
a=1
b=2
c=3
x=[a, b, c]
```

Macierze:

```
A=[1,2,3; 4,5,6; 7,8,9]
B=[1,2,3
1, 2, 3
1, 2, 3]
u=[-1, -2, -3]
v=[1, 1, 1]
w=[-1, 0, 1]
C=[u; v; w] // należy zwrócić uwagę na te trzy polecenia
r=[u v w] // i dzielące je różnice
D=[u' v' w']
```

Operacje na macierzach (**Uwaga**, bardzo istotna jest zgodność wymiarów macierzy)

```
A+B
C-D
A*B - mnożenie macierzowe
B*A
A.*B - mnożenie 'element po elemencie'
C*u
inv(A)
cond(B)
det(C)
A*inv(A)
```

Operacje 'element po elemencie':

```
.* ./ .^
```

Rozwiązywanie układów równań liniowych

```
A=[1, 3, 2; 2,13, 8; 0, 2 3]
b=[1, 2, -4]
x=A\b
```

Jest to najszybszy sposób rozwiązania układu równań.

A jak inaczej rozwiązać układ równań?

Szybkie tworzenie specjalnych macierzy i wektorów

```
c=ones(5,3)
d=zeros(10,1)
dd=zeros(10)
I=eye(5,5)
D=diag([2 1 0 -1 -2])
L=diag([1,2,3,4], -1)
U=rand(3,3)
R=rand(3,3,'normal')
rv=rand(10,1)
rv=rand(1,10)
```

Zapisanie sesji do pliku:

```
diary ('sesja.txt')
.....
.....
```

diary (0)

W pliku `sesja.txt` znajduje się zapis wszystkich wykonanych poleceń oraz wyników wypisywanych na monitor.

Skrypty

Polecenia programu Scilab też można umieścić w pliku (skrypcie) i wielokrotnie wykonywać. W plikach tych oprócz poleceń umieszcza się komentarze; są to linie rozpoczynające się `//`. Zwyczajowe rozszerzenie takich plików to `.sce`. Uruchomienie poleceń z pliku następuje po wprowadzeniu polecenia:

```
exec plik.sce
```

Funkcje:

```
sin, cos, log, exp, abs, sqrt  
sum, max, min  
sort
```

Grafika - rysowanie prostych wykresów

```
x=(-5 : 0.1 : 5)';  
y = x .* abs(x) ./ (1 + x.^2);  
xbas(); // usunięcie poprzedniego wykresu  
plot2d (x,y)
```

UWAGA: Funkcja `plot2d` wymaga, aby wektory były wektorami kolumnowymi. Jeśli tak nie jest, można wykorzystać operator transpozycji `'`, np. `plot2d(x',z')`

```
y2=x .* abs(x) ./ (5 + x.^2);  
plot2d (x,y2,style=-1)
```

style równe wartości ujemnej oznacza wykreślanie **tylko markerów punktów**; różnym wartościom odpowiadają różne markery.

```
y3= x .* abs(x) ./ (1/5 + x.^2);  
plot2d (x,y3,style=5)
```

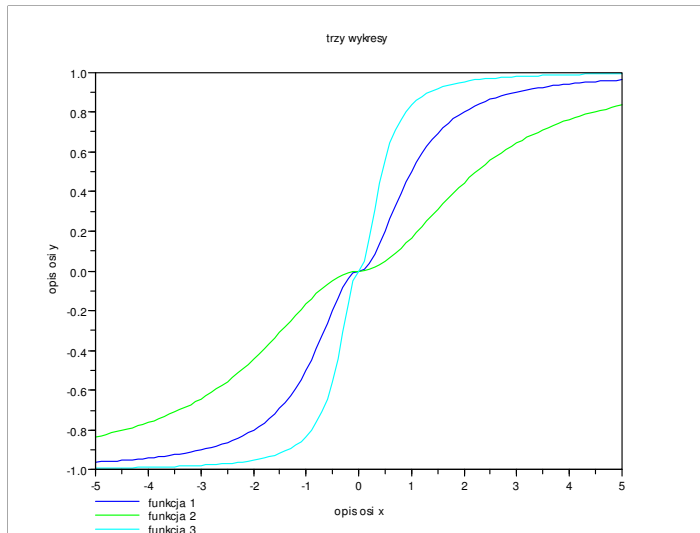
style **dotądnie** definiują kolory wykresu.

Dodamy tytuł:

```
xtitle ('trzy wykresy')
```

Jeszcze raz; dodatkowo pojawi się legenda oraz opis osi:

```
xbas()  
plot2d (x, [y y2 y3], leg='funkcja 1@funkcja 2@funkcja 3', style=[2 3 4])  
xtitle('trzy wykresy', 'opis osi x','opis osi y') // tytuł wykresu oraz  
opisy osi
```

**Przykład:**

```
x=linspace(0, 2*pi, 40)'; // 40 wartości równomiernie rozłożonych w
                           // przedziale [0, 2*pi]. Uwaga na transpozycję.
y=sin(x);
yp=sin(x) + 0.1* rand(x,'normal'); // zaburzenie gaussowskie
xbasc()
plot2d (x, [yp, y], style=[-2, 2], leg='y=sin(x) + zaburzenie@y=sin(x)')
```

Zadanie: Wykorzystać funkcję `plot2d` do pokazania, że

$$\lim_{x \rightarrow 0} \frac{\sin(x)}{x} = 1$$

Funkcje pisane przez użytkowników - odrobina programowania

```
function [y]=f(x)
y=x*abs(x)/(1+x^2);
endfunction
```

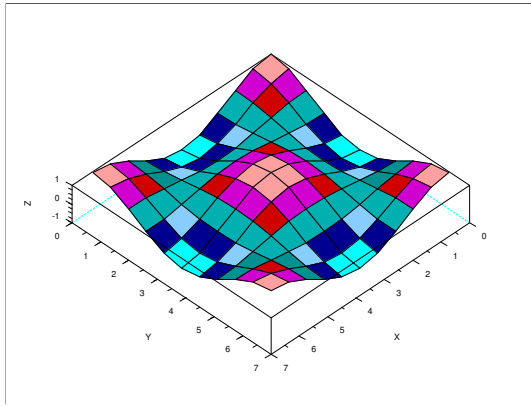
```
fplot2d (x,f)
```

Wykresy w 3D

```
x=linspace (0, 2*pi, 11); // wektor wierszowy
y=x;
z=cos(x)' * cos(x); // wynikiem takiego mnożenia jest macierz!
plot3d(x,y,z);
```

albo:

```
plot3d1(x,y,z);
```



Większy przykład: Rozwiązywanie równań różniczkowych zwyczajnych

Wykorzystamy funkcję `ode` (Ordinary Differential Equation).

Należy podać cztery **parametry funkcji**:

- warunek początkowy
- chwilę początkową t_0
- wektor kolejnych punktów czasowych, w których jest wyznaczane rozwiązanie
- nazwa funkcji, w której zapisana jest prawa strona równania.

Funkcja ta w ogólnym przypadku służy do rozwiązywania **układów** równań różniczkowych cząstkowych.

Na początek rozwiążemy równanie logistyczne:

$$\frac{dN(t)}{dt} = r N(t) \left(1 - \frac{N}{K}\right),$$

gdzie: $N(t)$ - zagęszczenie osobników w chwili t ;

$r > 0$ – współczynnik rozrodczości gatunku;

$K > 0$ – pojemność środowiska,

dla:

$K=150$;

$r=1$;

// TU: zapisana jest **prawa strona równania**

```
function [pochodne]=prawa_strona(t,N)
```

```
pochodne(1)=r*N*(1-N/K);
```

```
endfunction
```

```
N0=10;
```

```
t0=0;
```

```
t=0:0.1:10;
```

```
y=ode(N0,t0,t,prawa_strona);
```

```
xbasc();
```

```
plot2d(t,y);
```

Dla **układu** równań sprawa wygląda podobnie, aczkolwiek bardziej skomplikowanie:

Model Lotki-Volterra

Niech: $H(t)$ – zagęszczenie ofiar w chwili t ,

$P(t)$ – zagęszczenie drapieżników w chwili t .

Wtedy populacja dwóch gatunków jest opisana układem równań różniczkowych zwyczajnych:

$$\begin{cases} \frac{dH(t)}{dt} = r H(t) - a H(t) P(t) \\ \frac{dP(t)}{dt} = b H(t) P(t) - m P(t) \end{cases}$$

gdzie: r – współczynnik rozrodczości gatunku ofiar

a – współczynnik skuteczności polowań

b – współczynnik rozrodczości drapieżników (na jednostkę upolowanej ofiary)

m – współczynnik śmiertelności drapieżników

Wyznamy populacje gatunków dla... (wartości są zapisane w skrypcie, plik lotka_volterra.sce):

```
// oznaczenia: y(1) <--> H
//              y(2) <--> P
// pochodne(1) - prawa strona 1 rownania
// pochodne(2) - prawa strona 2 rownania
function [pochodne]=prawa_strona(t,y)
pochodne(1) = r*y(1)-a*y(1)*y(2)
pochodne(2) = b*y(1)*y(2)-m*y(2)
endfunction

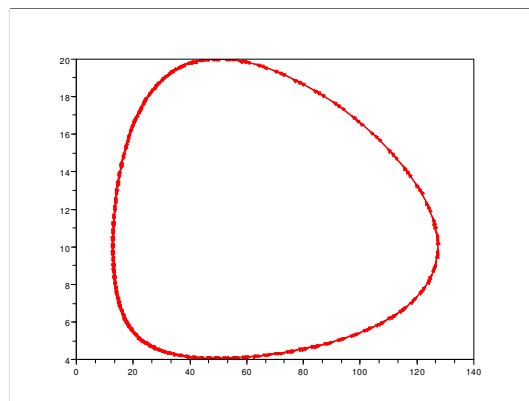
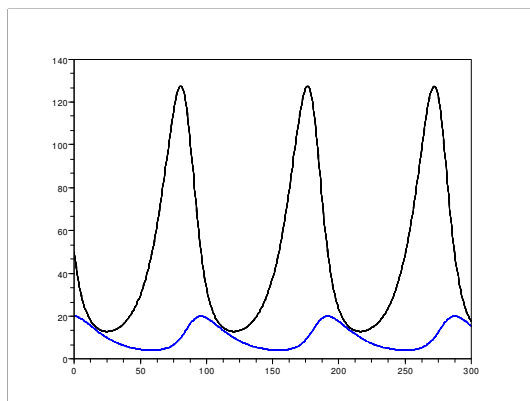
t0=0;
t=0:1:300;

H0=50;
P0=20;
r=0.1;
a=0.01;
b=0.001;
m=0.05;

HiP=ode([H0; P0],t0,t,prawa_strona);
// rozwiazanie H znajduje sie w pierwszym wierszu macierzy HiP (Hi(1,:))

xbascc();
xset ('window',1);
plot2d(t, [HiP(1,:) ' HiP(2,:)'], style=[1,2]);

xset ('window',2);
xbascc();
plot2d4(HiP(1,:) ', HiP(2,:) ', style=[5]);
```



Co trzeba zmienić w powyższym kodzie, aby rozwiązać układ równań opisujących następujący model?

- **Mutualizm (symbioza)**

Model uwzględniający ograniczenia zysków wynikających z mutualizmu, w którym w wyniku pozytywnego oddziaływania drugiego gatunku zwiększa się pojemność środowiska gatunku pierwszego, można opisać w postaci następującego układu równań (May, 1981):

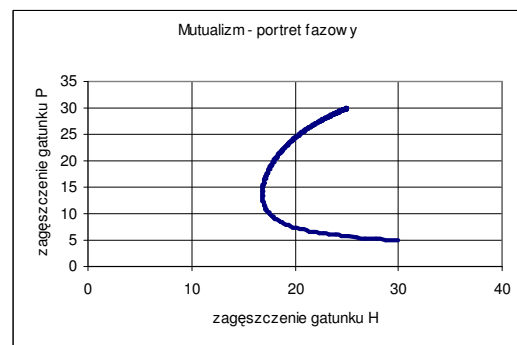
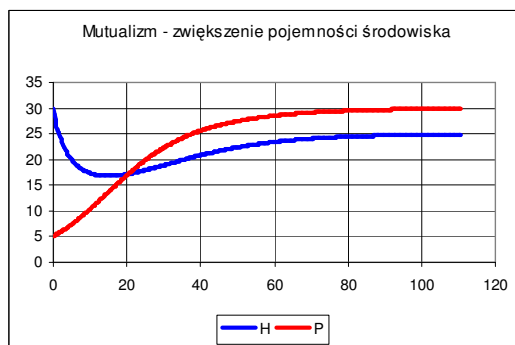
$$\begin{cases} \frac{dH}{dt} = r_1 H \left(1 - \frac{H}{K_1 + \alpha P}\right) \\ \frac{dP}{dt} = r_2 P \left(1 - \frac{P}{K_2 + \beta H}\right) \end{cases}$$

gdzie: r_1, r_2 – współczynniki rozrodczości gatunków

K_1, K_2 – pojemności środowiska dla obu gatunków

α, β – intensywność mutualistycznych oddziaływań między gatunkami.

W obliczeniach przyjąć parametry z rysunku poniżej.



Wyniki uzyskano dla następujących wartości parametrów:

$r_1=0,1$ $r_2=0,1$, $K_1=10$, $K_2=20$, $\alpha=0,5$, $\beta=0,4$,
 $H_0=30$, $P_0=5$.

Generowanie ciągów liczb (pesudo) losowych – przykład

Przykłady wywołania funkcji `rand`.

`rand(1, 20)` – wektor wierszowy o 20 elementach o rozkładzie jednostajnym (0,1)

`rand(1, 20)'` – wektor kolumnowy (bo zrobiono transpozycję)

`rand(20, 1)` – wektor kolumnowy (= macierzy o 20 wierszach i jednej kolumnie)

`rand(20, 1, 'normal')` – wektor kolumnowy o 20 elementach o rozkładzie normalnym (0,1).

`rand(3, 4)`

`rand(3, 5, 'normal')`

Wektory i macierze losowe można konstruować w oparciu o istniejące wektory lub macierze. Powstały wektor/macierz będzie miał taką samą liczbę wierszy i kolumn. Operacja `rand(x)` nie wpływa na zawartość wektora x .

```
x=linspace(-100,100,5001);
```

```
y=rand(x);
```

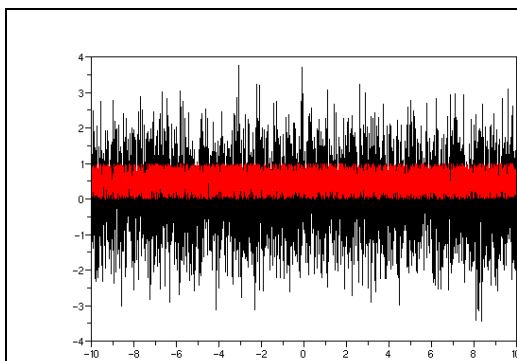
```
z=rand(x, 'normal');
```

Przykładowy większy skrypt:

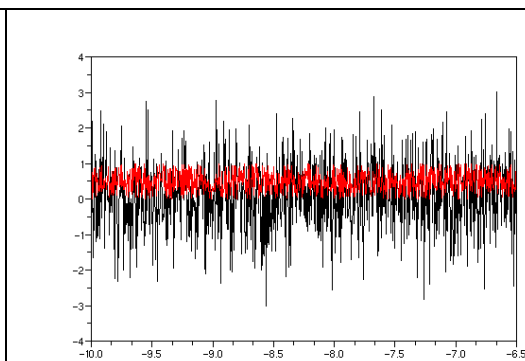
```

x=linspace(-10,10,5001);
y=rand(x); // rozkład jednostajny
z=rand(x,'normal'); // rozkład normalny
xset ('window',0);
xbasc()
histplot(20,z); // HISTOGRAM dla rozkładu normalnego, wartości podzielono
na 20 grup.
xtitle ('rozkład normalny');
xset ('window',1);
xbasc()
histplot(20,y); // HISTOGRAM dla rozkładu jednostajnego
xtitle ('rozkład jednostajny');
xset ('window',2);
plot2d(x,z);
plot2d(x,y,style=5);

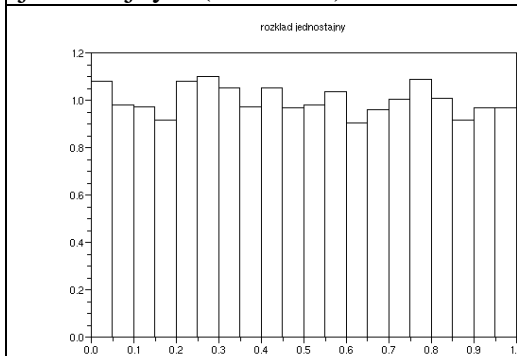
```



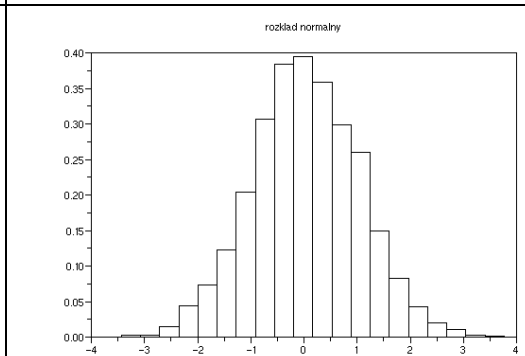
Na wykresie znajdują się wartości o rozkładzie normalnym (czarne) i jednostajnym (czerwone).



Powiększenie rysunku obok.



Histogram dla rozkładu jednostajnego



Histogram dla rozkładu normalnego