

## „Notatki” z sesji Scilaba

Istnieje możliwość dokładnego zapisu przebiegu aktualnej sesji pracy ze Scilabem: polecenie **diary('nazwa\_pliku.txt')** powoduje zapis do podanego pliku tekstowego wszystkich wpisywanych przez nas poleceń oraz wszystkich odpowiedzi Scilaba. **Zakończenie: diary(0)**. Polecenie to „koniecznie” chce założyć nowy plik tekstowy o podanej nazwie — zatem po przerwaniu zapisu trzeba zacząć zapisywanie do pliku o innej nazwie (we wcześniejszych wersjach polecenie to zachowywało się na tyle kulturalnie, że nie kasowało wcześniejszego pliku o podanej nazwie tylko informowało, że taki plik już istnieje, w wersji 3.0 kasuje utworzony wcześniej plik nie „zawracając głowy” użytkownikowi).

## Wykresy

**Scilab umożliwia tworzenie różnych typów wykresów 2D lub 3D oraz ich dostosowywanie. Scilab umożliwia także eksport stworzonych wykresów do różnego typu plików graficznych.**

Tworzenie wykresu składa się najczęściej z następujących etapów:

- utworzenie ciągu wartości 'x-ów'
- utworzenie ciągu wartości 'y-ków'
- rysowanie
- zapisanie rysunku do pliku graficznego (opcjonalnie).

Uwaga: Robienie wykresu na komputerze wykorzystuje dyskretne przedstawienie funkcji w postaci ciągu wartości 'punktowych', zawierających współrzędne punktów, w oparciu o które tworzony jest wykres. Im więcej jest takich punktów, tym dokładniejszy jest wykres.

Wartości 'x-ów':

Wartości 'x-ów' można w Scilabie zdefiniować na kilka sposobów, wpisując odpowiednie polecenie na konsoli.

```
x=[0,1,2,3,4,5,5.5,10,20]; - ciąg wartości
x=(-10:0.1:10);
    (wartość początkowa : krok : wartość końcowa)
x=linspace(0, 3.141592, 20);
    (wartość początkowa, wartość końcowa, ile wartości)
```

**Uwagi:**

- ; (średnik) na końcu sprawia, że wyznaczone wartości nie są wypisywane na konsoli;
- określenie 'x' jest symboliczne, równie dobrze tworzony ciąg wartości może mieć nadaną inną nazwę, np.: `z=(0:1:100)`, albo `a1a=(-5:0.1:5)`;
- jako wartość  $\pi$  można wpisać `%pi`, czyli: `x=linspace(0,%pi,20)`.
- Przy okazji zwracamy uwagę na fakt, że część ułamkowa jest oddzielona od części całkowitej kropką.

Wartości 'y-ków':

Uwaga: określenie 'y-ki' jest symboliczne; tworzony obiekt może mieć dowolną nazwę.

Wartości 'y-ków' można tworzyć na wiele sposobów, na razie wykorzystamy najprostsze. W poniższych przykładach wartości są budowane w oparciu o wcześniej utworzony ciąg 'x-ów'.

```

y=x;
y1=2*x;
z=2*x-1;
fun=sin(x)+cos(2*x);
y2=x^3
g=tan(x)^2;

```

Ale wartości 'y-ków' mogą również być zdefiniowane jako ciąg wartości.

```
y=[1,3,5,15,-1]
```

Rysowanie – funkcja plot

**plot(x,y)**

Uwaga przed pierwszym przykładem. Zaleca się, aby każdy program rozpoczynać poleceniem:

**clear();**

Polecenie to powoduje 'wyczyszczenie' wszystkich zmiennych – upewniamy się w ten sposób, że podczas obliczeń nie pojawią się żadne 'stare' wartości.

Pierwszy przykład:

**clear();**

**x=linspace(0,%pi,50);**

**y=sin(x);**

**plot(x,y);**

**y1=cos(2\*x);**

**plot(x,y1);**

**xgrid();**

Zapisanie rysunku do pliku:

W oknie graficznym (interakcyjnie):

Plik / eksportuj do / ... wybrać typ pliku (PNG, GIF, JPG,...), podać nazwę pliku. Poprzez wpisanie w oknie konsoli odpowiedniego polecenia, np:

**xs2png(numer\_okna\_graficznego,'nazwa\_pliku.png')**

Uwagi:

numer\_okna\_graficznego – jest wyświetlony w pasku tytułowym okna. Standardowo pierwsze utworzone okno ma numer 0.

Scilab wyróżnia katalog bieżący (Plik/ Wyświetl katalog bieżący). O ile nazwa pliku nie zostanie poprzedzona ścieżką dostępu, plik zostanie zapisany w katalogu bieżącym. Zmiana katalogu bieżącego: Plik/ Zmiana bieżącego katalogu...

Inne formaty plików graficznych, to (między innymi):

**eps – funkcja xs2eps,**

**postscript – xs2ps,**

**pdf – xs2pdf,**

**gif – xs2gif,**

**jpg – xs2jpg.**

Więcej informacji o tworzeniu wykresów:

Kolejne polecenia 'plot' powodują dodanie ('dorysowanie') kolejnego wykresu w bieżącym oknie.

Operacje na oknach:

**clf()** – wyczyszczenie bieżącego okna.

**clf(1)** – wyczyszczenie okna nr 1.

**scf(1)** – utworzenie okna o numerze 1.

**xdel()** – usunięcie bieżącego okna.

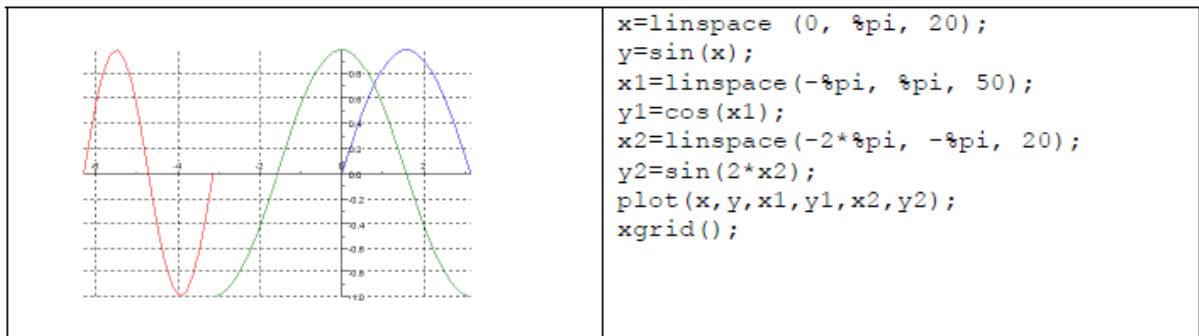
### **xdel(1) – usunięcie okna o numerze 1.**

Kilka wykresów (różnych) funkcji na jednym rysunku:

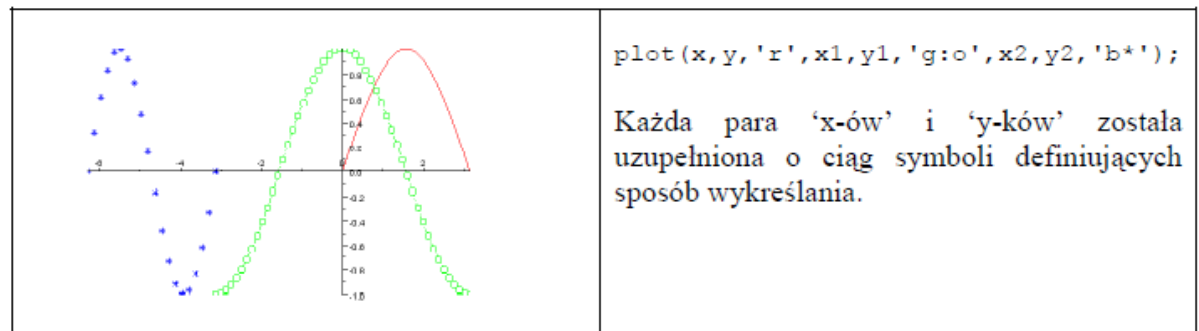
Polecenie:

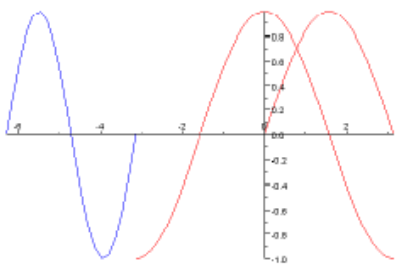
#### **plot(x,y,x,y1);**

spowoduje narysowanie wykresu funkcji przechowywanej w wektorze y, oraz drugiej funkcji przechowywanej w wektorze y1. Obydwa wykresy wykorzystują ten sam zestaw x-ów. Ale również można na jednym rysunku umieszczać wykresy zdefiniowane dla różnych zestawów argumentów (zakresów lub 'gęstości' x-ów).



Kolory są ustalane automatycznie. Poprzez odpowiednie zdefiniowanie parametrów można sterować zarówno kolorami, jak i rodzajem linii oraz markerów; porównajmy z poprzednim wykresem:



symbol	kolor	<pre>plot(x,y,'r',x1,y1,'r',x2,y2);</pre> 
r	czerwony	
g	zielony	
b	niebieski	
c	cyjan	
m	magenta	
y	żółty	
k	czarny	
w	biały	

### **Style wykreślenia linii:**

plot(x,y,'-'); – linia ciągła (domyślnie)

plot(x,y,'- -'); – linia przerywana

plot(x,y,':'); – linia kropkowana

plot(x,y,'-.'); – linia kreskowo-kropkowa

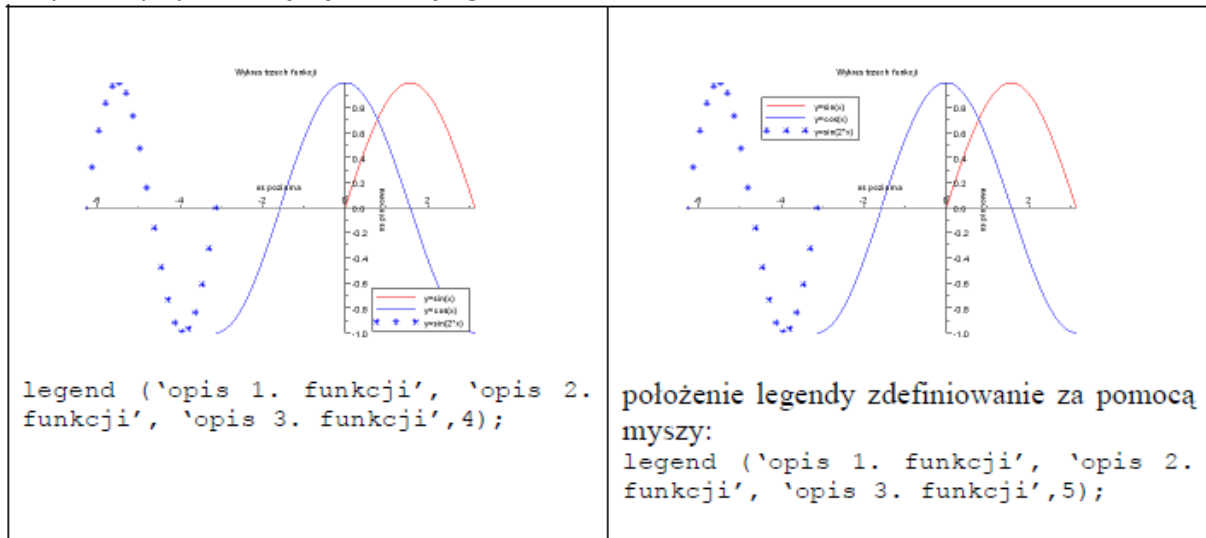
Opisywanie wykresów: tytuł, opisy osi, legenda

**title('Tytuł wykresu');**

**xtitle('Tytuł wykresu','opis osi x-ów','opis osi pionowej');**

**legend ('opis 1. funkcji', 'opis 2. funkcji', 'opis 3. funkcji');**

Wszystkie opisy odnoszą się do bieżącego okna.



Uwaga: Legendę można uzupełnić o informację o jej położeniu na rysunku poprzez podanie na końcu opcjonalnego parametru. Domyślnym położeniem legendy jest prawy górny róg.

Polecenie:

**legend ('opis 1. funkcji', 'opis 2. funkcji', 'opis 3. funkcji', 4);**

spowoduje umieszczenie legendy w lewym dolnym rogu.

Przykładowe inne możliwości – można podawać albo opis liczbowy, albo opis słowny (w apostrofach):

1 lub "in\_upper\_right" – prawy górny róg, przyjmowane domyślnie

2 lub "in\_upper\_left" – lewy górny róg

3 lub "in\_lower\_left" – lewy dolny róg

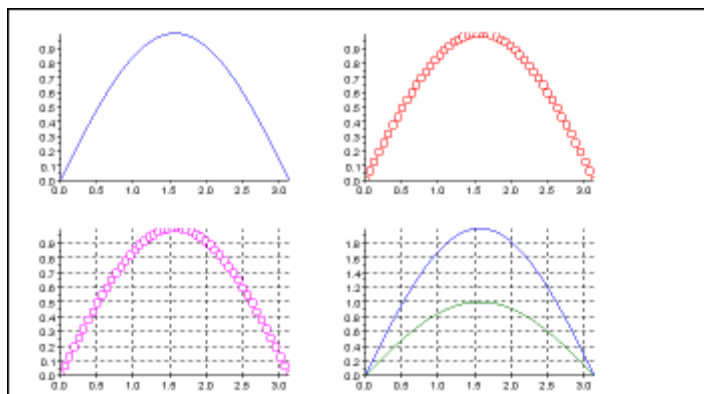
4 lub "in\_lower\_right" – prawy dolny róg

5 lub "by\_coordinates" – położenie legendy zdefiniowanie za pomocą myszki w oknie graficznym.

Kilka rozłącznych wykresów w jednym oknie – subplot

Przykład 4 wykresów rozmieszczonych w 2 kolumnach i 2 wierszach.

```
-->clf()
-->subplot(2,2,1);
-->plot(x,y)
-->subplot(2,2,2);
-->plot(x,y,'ro-.')
-->subplot(2,2,3);
-->plot(x,y,'mo-.')
-->xgrid()
-->subplot(2,2,4);
-->plot(x,2*y,x,y);
-->xgrid()
```



### Zadania:

Narysować trójkąt o wierzchołkach w punktach: (0,0), (10, 1), (5,7).

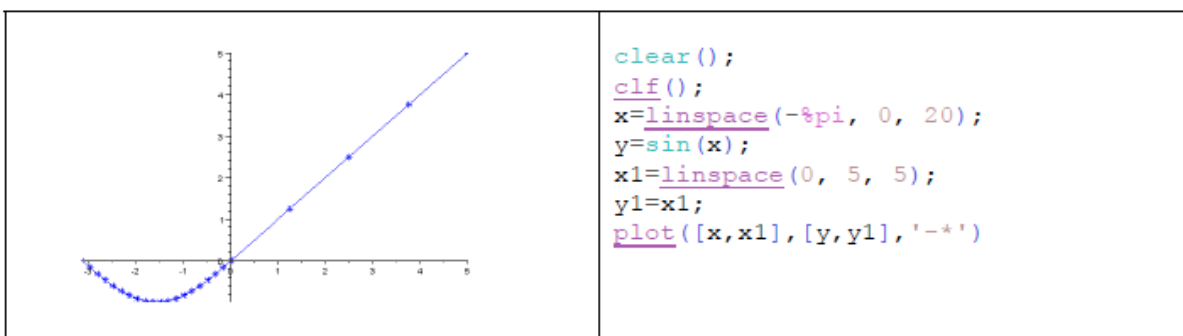
```
clear();
x=[0,10,5,0];
y=[0,1,7,0];
plot(x,y)
```

Narysować prostokąt o wierzchołkach w punktach (0,0), (10, 0), (10, 4), (0,4) oraz punktem w innym kolorze zaznaczyć jego środek.

```
clear();
clf();
x=[0,10,10,0,0];
y=[0,0,4,4,0];
plot(x,y,'r')
x1=[5];
y1=[2];
plot(x1,y1,'sg');
```

Narysować wykres funkcji opisanej wzorem:

$$y = \begin{cases} \sin(x), & x \in [-\pi, 0) \\ x, & x \in [0, 5) \end{cases}$$



### Przykład.1

Uruchamiamy w Scilabie edytor. Wprowadzamy poniższy kod generujący i wyświetlający funkcję sinusa:

```
clear; // wyczyszczenie pamięci
xdel(winsid()); //wyczyszczenie (zamknięcie) wszystkich okien graficznych
clc; //wyczyszczenie konsoli
t=linspace(0, 2*pi, 100); // (obliczenie od 0 do 2pi, liczba iteracji=100)
y=sin(t); // funkcja którą liczymy, czyli sinus
plot2d(t,y) // wygenerowanie wykresu liczonej funkcji y=f(t)
```

### Przykład.2

Utworzyć i wykonać skrypt o poniższym kodzie. Zapisać pod nazwą *funkcja.sce* w katalogu roboczym. Wykonanie skryptu wykonujemy przez wywołanie w terminalu polecenia `exec ('funkcja.sce')`. Przetestować „tryby” wykonywania skryptu `{exec ('funkcja.sce',0), exec ('funkcja.sce',1)}`. Jest to skrypt do wygenerowania wykresu funkcji  $e^{-x} \sin(4x)$  ze zmiennym za każdym razem rozmiarem przedziału  $[a; b]$  i liczbą podprzedziałów.

```
clear; // wyczyszczenie pamięci
xdel; //wyczyszczenie aktualnej grafiki
clc; //wyczyszczenie konsoli
a=input(" Podaj lewy kraniec przedziału : ");
b=input(" Podaj prawy kraniec przedziału : ");
n=input(" Podaj ilość podprzedziałów : ");
x=linspace(a,b,n+1); // obliczenie odciętych
y=exp(-x).*sin(4*x); // obliczenie rzędnych
plot(x,y,"b") // rysunek funkcji b oznacza kolor blue charakterystyki
xlabel("y=exp(-x).*sin(4*x)", "y", "x")
```

### Przykład 3.

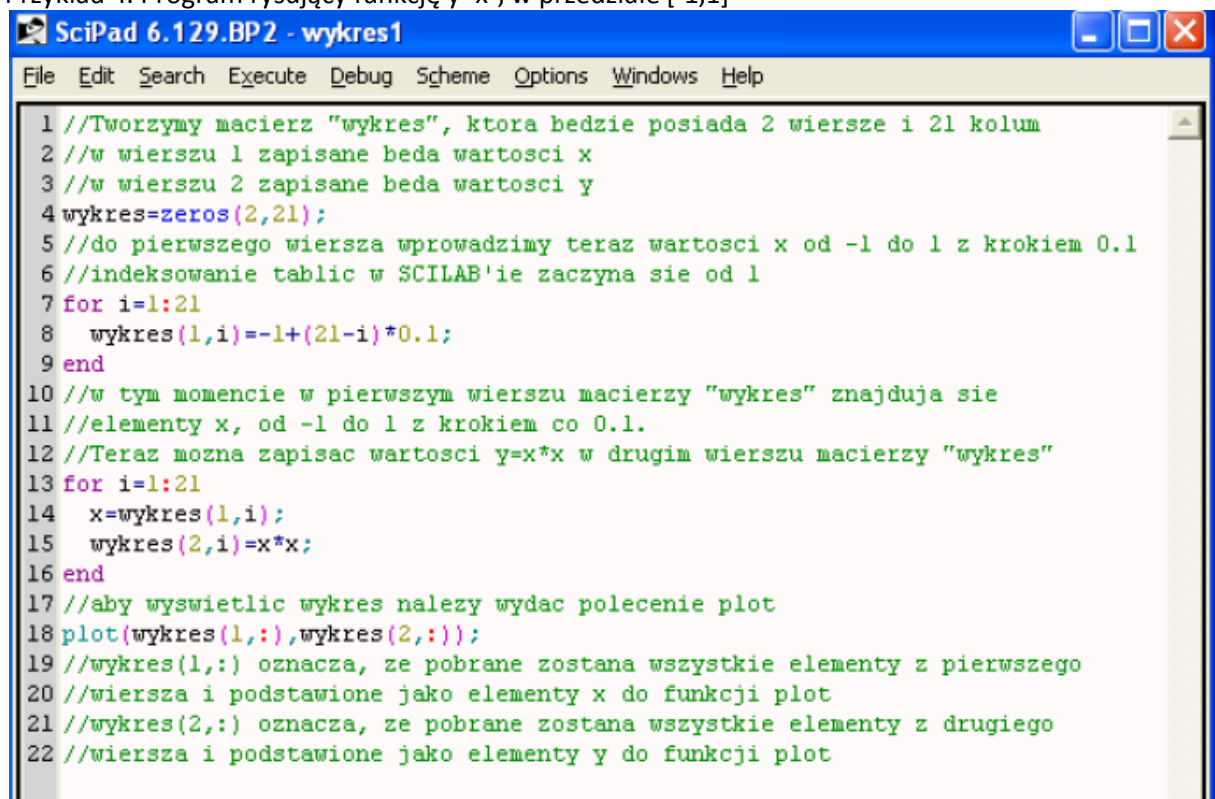
Wyczyść aktualne okno z grafika za pomocą polecenia: `--> clf()`.

Wygeneruj serie danych  $x$ : `--> x=[0:0.1:2*pi]'`;

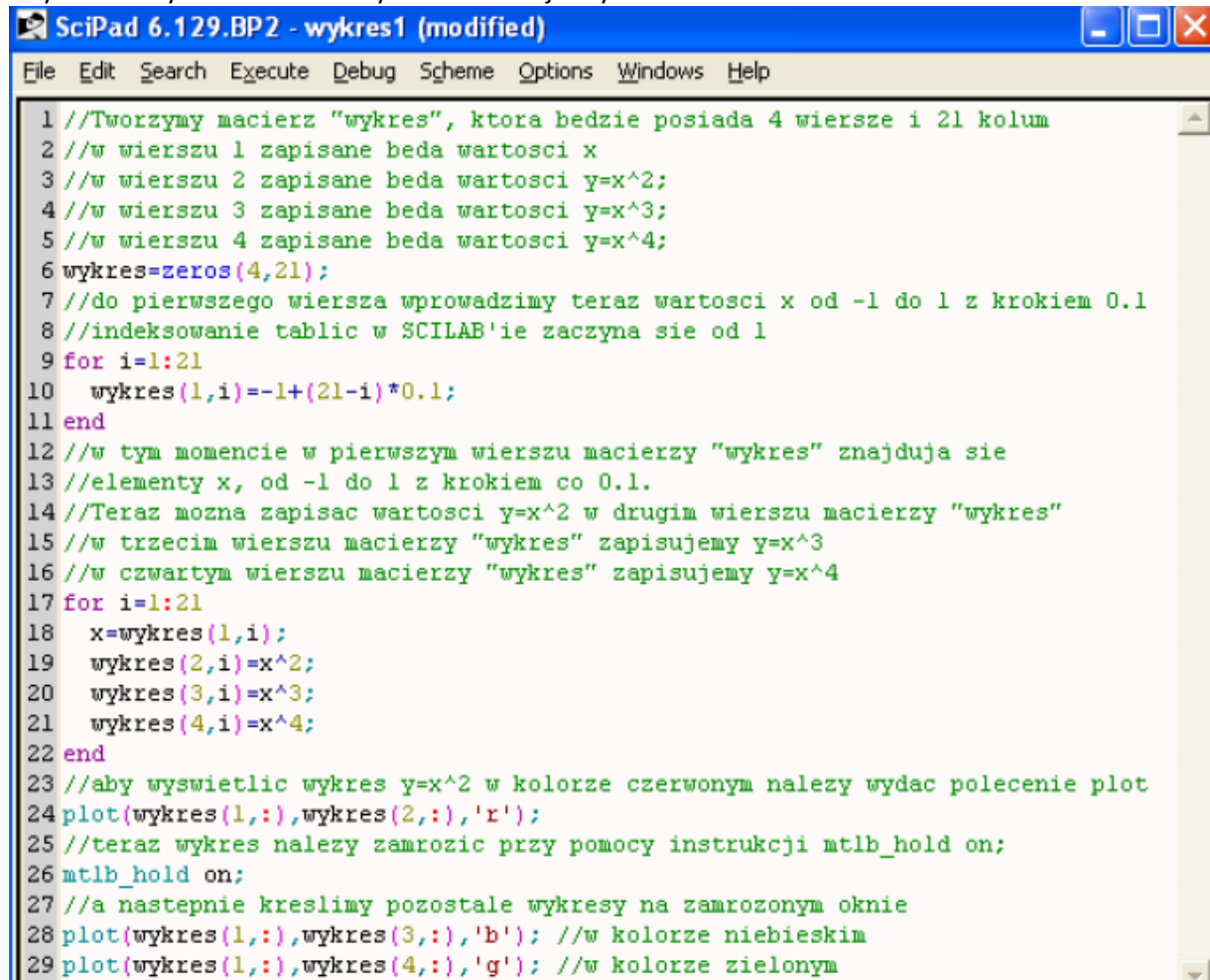
Wywołaj polecenia.

```
plot (x, sin(x), "b")
plot (x, sin(2*x), "g")
plot (x, sin(3*x), "r")
```

Przykład 4. Program rysujący funkcję  $y=x^2$ , w przedziale  $[-1,1]$



Przykład 5. Wykreślenie kilku wykresów 2D w jednym oknie.



```
1 //Tworzymy macierz "wykres", ktora bedzie posiada 4 wiersze i 21 kolum
2 //w wierszu 1 zapisane beda wartosci x
3 //w wierszu 2 zapisane beda wartosci y=x^2;
4 //w wierszu 3 zapisane beda wartosci y=x^3;
5 //w wierszu 4 zapisane beda wartosci y=x^4;
6 wykres=zeros(4,21);
7 //do pierwszego wiersza wprowadzimy teraz wartosci x od -1 do 1 z krokiem 0.1
8 //indeksowanie tablic w SCILAB'ie zaczyna sie od 1
9 for i=1:21
10     wykres(1,i)=-1+(21-i)*0.1;
11 end
12 //w tym momencie w pierwszym wierszu macierzy "wykres" znajduja sie
13 //elementy x, od -1 do 1 z krokiem co 0.1.
14 //Teraz mozna zapisac wartosci y=x^2 w drugim wierszu macierzy "wykres"
15 //w trzecim wierszu macierzy "wykres" zapisujemy y=x^3
16 //w czwartym wierszu macierzy "wykres" zapisujemy y=x^4
17 for i=1:21
18     x=wykres(1,i);
19     wykres(2,i)=x^2;
20     wykres(3,i)=x^3;
21     wykres(4,i)=x^4;
22 end
23 //aby wyswietlic wykres y=x^2 w kolorze czerwonym nalezy wydac polecenie plot
24 plot(wykres(1,:),wykres(2:,:), 'r');
25 //teraz wykres nalezy zamrozic przy pomocy instrukcji mtlb_hold on;
26 mtlb_hold on;
27 //a nastepnie kreslimy pozostale wykresy na zamrozonym oknie
28 plot(wykres(1,:),wykres(3:,:), 'b'); //w kolorze niebieskim
29 plot(wykres(1,:),wykres(4:,:), 'g'); //w kolorze zielonym
```

Polecenie **subplot(x,y,z);**

x – oznacza liczbę wierszy z wykresami

y – oznacza liczbę kolumn z wykresami

z – oznacza numer okna w którym rysujemy.

Poniżej znajduje się kod demonstrujący działanie instrukcji **subplot**



Przykład 6.

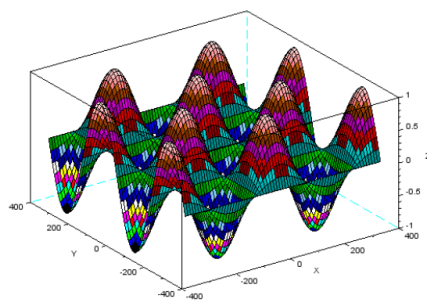
```
SciPad 6.129.BP2 - wykres1 (modified)
File Edit Search Execute Debug Scheme Options Windows Help

1 //Tworzymy macierz "wykres", ktora bedzie posiada 4 wiersze i 21 kolum
2 //w wierszu 1 zapisane beda wartosci x
3 //w wierszu 2 zapisane beda wartosci y=x^2;
4 //w wierszu 3 zapisane beda wartosci y=x^3;
5 //w wierszu 4 zapisane beda wartosci y=x^4;
6 wykres=zeros(4,21);
7 //do pierwszego wiersza wprowadzimy teraz wartosci x od -1 do 1 z krokiem 0.1
8 //indeksowanie tablic w SCILAB'ie zaczyna sie od 1
9 for i=1:21
10     wykres(1,i)=-1+(21-i)*0.1;
11     x=wykres(1,i);
12     wykres(2,i)=x^2;
13     wykres(3,i)=x^3;
14     wykres(4,i)=x^4;
15 end
16 //tworzymy wykresy w 2 wierszach i 1 kolumnie i aktualnie rysujemy w 1 z nich
17 subplot(2,1,1);
18 //aby wyswietlic wykres y=x^2 w kolorze czerwonym nalezy wydac polecenie plot
19 plot(wykres(1,:),wykres(2:,:), 'r');
20 //teraz wykres nalezy zamrozic przy pomocy instrukcji mtlb_hold on;
21 mtlb_hold on;
22 //a nastepnie kreslimy pozostale wykresy na zamrozonym oknie
23 plot(wykres(1,:),wykres(3:,:), 'b'); //w kolorze niebieskim
24 //teraz zmieniamy okno i rysujemy w nim wykres y=x^4
25 subplot(2,1,2);
26 plot(wykres(1,:),wykres(4:,:), 'g'); //w kolorze zielonym
```

Przykład 7.

- Wyczyść aktualne okno z grafiką za pomocą polecenia: --> `clf()`.
- Wygeneruj serię danych x: --> `x=[-360:10:360];`.
- Wygeneruj serię danych y: --> `y=[-360:10:360];`.
- Stwórz tablicę funkcji  $z = \sin(x) \cdot \cos(y)$ :  
--> `z=sind(x')*cosd(y);`.
- Stwórz wykres powierzchniowy za pomocą polecenia:  
--> `surf(x,y,z)`.

Wynik:





## Funkcje

Jeśli blok tych samych operacji jest wykonywany wielokrotnie, to warto te operacje zapisać w postaci funkcji. Dzięki temu będzie można ją wykorzystać wielokrotnie.

Najprostszy sposób wywołania funkcji to:

wynik = funkcja (parametr).

Parametry wejściowe funkcji nie są przez nią modyfikowane. Funkcje mogą przyjmować wiele parametrów wejściowych i zwracać więcej niż jedna wartość. W takim przypadku ogólniejsza postać

wywołania jest następująca:

[wyn\_1, ..., wyn\_n] = funkcja (param\_1, ..., param\_m).

Do definiowania funkcji wykorzystuje się słowa kluczowe `function` i `endfunction`. Każda funkcja składa się z nagłówka oraz ciała funkcji. Nagłówkiem funkcji jest jej nazwa wraz ze zdefiniowanymi parametrami wejściowymi oraz wyjściowymi. Na ciało funkcji składają się wszystkie polecenia zawarte między nagłówkiem funkcji a słowem kluczowym `endfunction`.

Funkcje można definiować na 3 możliwe sposoby:

- bezpośrednio w konsoli Scilab
- w oddzielnym pliku uruchamianym w edytorze
- za pomocą polecenia `exec`

### Przykład 1

- Przejdź do konsoli Scilab.
- Zdefiniuj poniższą funkcję:

```
function r = kwadrat ( x )  
r = x^2  
endfunction
```
- Wywołaj powyższą funkcję i sprawdź, czy zwróciła poprawny wynik.

### Przykład 2

Stwórz funkcję `suma` oraz ją wywołaj

```
function r = suma ( x )  
r = x+x  
endfunction
```

### Przykład 3

Stwórz funkcję `suma` oraz ją wywołaj

- Zdefiniuj następującą funkcję:

```
function r = pierwiastek (x)
if (x < 0) then
disp("Podano liczbę ujemną!")
return
end
r = sqrt(x)
endfunction
```

- Wykonaj powyższą funkcję dla następujących wartości  $x=-4$ ,  $x=0$  oraz  $x=4$ .

#### Przykład 4

Stwórz funkcję srednia oraz ją wywołaj.

```
function z=srednia(x, y)
z=(x+y)*0.5;
endfunction
```

#### Małe zadania:

- dla danej wartości promienia  $r$  wyznaczyć pole powierzchni oraz objętość kuli;
- dla danej wartości  $a$  oznaczającej bok sześcianu, obliczyć pole powierzchni, obwód oraz długość przekątnej sześcianu;
- dla dwóch wartości oznaczonych  $a$  i  $b$  obliczyć pole i obwód prostokąta oraz długość jego przekątnej;
- dla dwóch wartości  $r$  i  $h$  obliczyć pole powierzchni oraz objętość cylindra o promieniu podstawy  $r$  oraz wysokości  $h$ .
- dla danej wartości promienia  $r$  wyznaczyć pole powierzchni trójkąta równobocznego wpisanego w okrąg o takim promieniu.
- dla danej wartości promienia  $r$  wyznaczyć pole powierzchni kwadratu wpisanego w okrąg o takim promieniu.