# Grafika w Octave

Marcin Szpyrka

Katedra Informatyki Stosowanej, AGH w Krakowie

2011/12



## Literatura

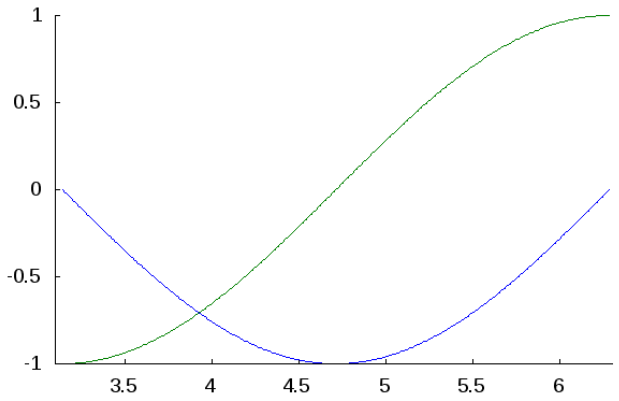
[1] John W. Eaton, David Bateman, Søren Hauberg: GNU Octave A high-level interactive language for numerical computations. Edition 3 for Octave version 3.0.5, July 2007 (pdf + Reference Card)

[2] Alfio Quarteroni, Fausto Saleri: Scientific Computing with MATLAB and Octave. Second Edition, Springer 2006

[3] P.J.G. Long: Introduction to Octave. Deparment of Engineering, University of Cambridge 2005 (pdf)

Funkcja plot w najprostszej wersji przyjmuje jako swoje argumenty wartości odciętych i rzędnych punktów należących do wykresu. Punkty te są następnie traktowane jako węzły łamanej.

```
plot([2 5 6 4 4])

x = [3:10];
y = [2 3 6 4 4 1 7 5];
plot(x, y)

x = [0 : 0.05 : 4];
plot(x, sqrt(x))

x = [1:5];
y = [2 4 5 3 2; 1 1 2 5 1];
plot(x, y)

x = [pi : 0.01 : 2*pi];
plot(x, [sin(x); cos(x)])

# lub
plot(x, sin(x), x, cos(x))
```






## Formatowanie wykresu – kolory i style linii

- Kolor linii wykresu można zmienić podając jednoliterowe oznaczenie jednego z predefiniowanych kolorów: y – żółty, m – purpurowy, c – zielononiebieski (cyan), r – czerwony, g – zielony, b – niebieski, w – biały i k – czarny.

```
plot(x, cos(x), 'r')
plot(x, cos(x), 'g')
plot(x, cos(x), 'r', x, sin(x),'g')
```

- Kolor linii wykresu można ustalić również podając trzy wartości z przedziału $[0, 1]$, określające kolor w schemacie RGB:

```
plot(x, cos(x), 'color', [0.4 0.5 0.2])
```

- Octave dostarcza zestaw opcji, które pozwalają definiować styl linii
  `- -- : -.`
  ale nie działają one w systemie X11.

- Zamiast linii ciągłej można wybrać jeden z dostępnych sposobów oznaczania wyliczonych punktów wykresu: + – symbol plusa, o – kółko, * – gwiazdka, x – znak x, . – kropka, ^ – znak potęgi, s – kwadrat, d – romb.

```
x = [0 : 0.1 : 2*pi];
plot(x, sin(x) + cos(x), '*')
```

## Formatowanie wykresu – osie układu współrzędnych

- Rysowanie punktu:

  ```
  plot(2,4,'r*')
  ```

- Rysowanie odcinka od punktu $(0,0)$ do $(1,2)$:

  ```
  plot([0 1],[0 2])
  ```

- Rysowanie zbioru odcinków:

  ```
  plot([1 2 3; 4 4 4], [2 2 2; 0 1 0])
  ```

  Są to odcinki: $(1,2)-(4,0)$, $(2,2)-(4,1)$ i $(3,2)-(4,0)$.



## Formatowanie wykresu – osie układu współrzędnych

Bezpośrednio po narysowaniu wykresu można zmodyfikować parametry dotyczące wyświetlania osi wykresu stosując polecenie axis.

- Ustawienie zakresu dla osi *x*: `axis([-2 2])`
- Ustawienie zakresu dla osi *x* i *y*: `axis([-5 5 -2 2])`
- Wymuszenie takiej samej skali na obu osiach: `axis('equal')`
- Włączenie/wyłączenie osi: `axis('on')`, `axis('off')`
- Włączenie znaczników osi dla wybranej/ych osi i wyłączenie dla pozostałych: `axis('tic[x]')`
- Odwrócenie osi *y* (mniejsze wartości na górze): `axis('ij')`
- Odwrócenie osi *y* (większe wartości na górze): `axis('xy')`

Opcje można łączyć w ramach polecenia axis:

```
axis([-2 8 -2 2], 'square', 'tic[x]')
```

## Rysowanie wykresów

- fplot – użycie wskaźnika do funkcji: `fplot(@sin, [-10 10])`
- fplot – użycie nazwy funkcji: `fplot('sin', [-10 10])`
- Wykres słupkowy:

```
y = [2 3 3 4 3 5 6 2 1];
bar(y)
bar(y, 0.2)
```

- Wykres słupkowy (ułożenie poziome): `barh(y)`
- Wykres schodkowy: `stairs(y)`
- Wykres kołowy: `pie(y)`
- Wypełniony wykres powierzchniowy:

```
x = linspace(0, 10, 200);
y = sin(x);
y = y .* x;
area(y)
```



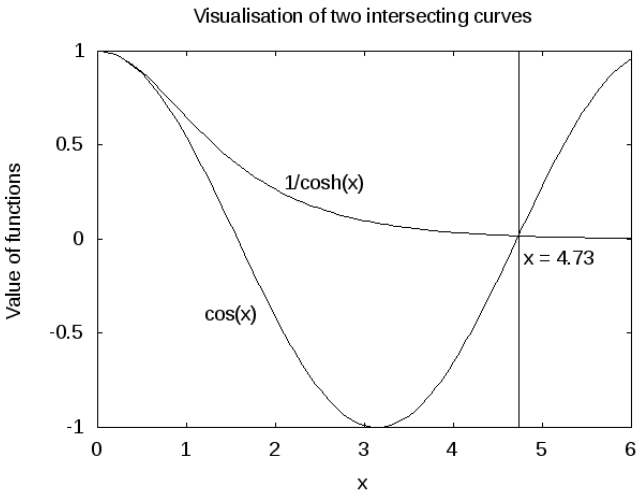
## Formatowanie wykresu

- Włączenie/wyłączenie ramki: `box('off')`, `box('on')`, `box`, `box on`, `box off`
- Wstawienie oznaczeń dla osi układu współrzędnych:

```
xlabel('X')
ylabel('Y')
xlabel 'X'
ylabel 'Y'
```

- Dodanie tytułu do wykresu: `title('f(x) = x*sin(x)')`
- Dodanie etykiety do wykresu: `text(50, 50, 'f(x)')`
- Włączenie/wyłączenie siatki: `grid('off')`, `grid('on')`, `grid`, `grid on`, `grid off`
- Włączenie/wyłączenie zamrażania wykresu (kolejny wykres będzie nakładany na obecny): `hold('off')`, `hold('on')`, `hold`, `hold on`, `hold off`

## Elementy wykresu

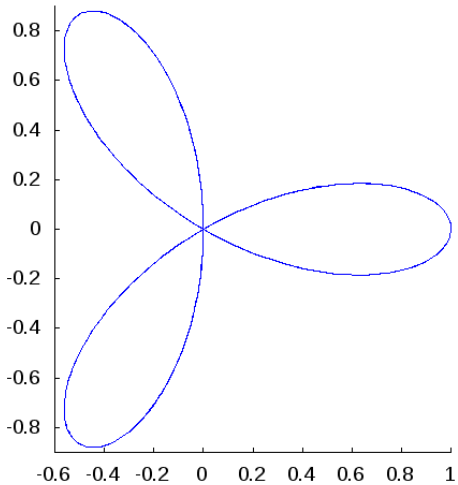



```
x = [0:0.05:6];
plot(x, cos(x), 'k', x, 1./cosh(x), 'k', [4.73 4.73], [-1 1], 'k')
xlabel('x')
ylabel('Value of functions')
title('Visualisation of two intersecting curves')
text(4.8, -0.1, 'x = 4.73')
text(2.1, 0.3, '1/cosh(x)')
text(1.2, -0.4, 'cos(x)')
```

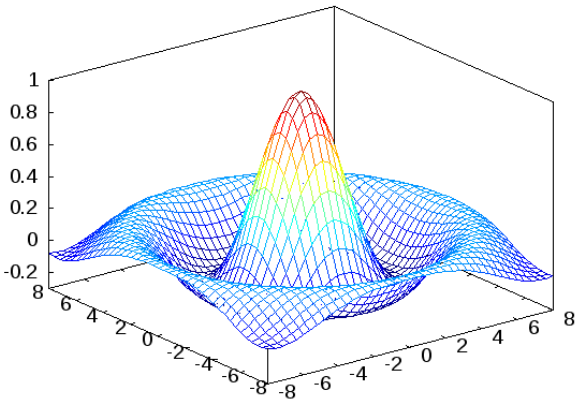


## Krzywe parametryczne

```
t = [0 : 0.01 : 2*pi];
x = cos(t);
y = sin(t);
plot(x,y)
axis('equal')

t = [0 : 0.01 : 2*pi];
r = cos(3 * t);
x = r .* cos(t);
y = r .* sin(t);
plot(x,y)
axis('equal')

t = [0 : 0.01 : 8*pi];
x = t .* cos(t);
y = t .* sin(t);
plot(x,y)
axis('equal')
```



```
x = y = linspace(-8, 8, 41)';
[xx, yy] = meshgrid(x, y);
z = sqrt(xx .^ 2 + yy .^ 2) + eps;
z = sin(z) ./ z;
mesh(x, y, z)
```




Funkcja meshgrid przygotowuje macierze z odpowiednimi wartościami z dziedziny funkcji $f(x, y)$. Funkcja mesh rysuje wykres powierzchniowy.

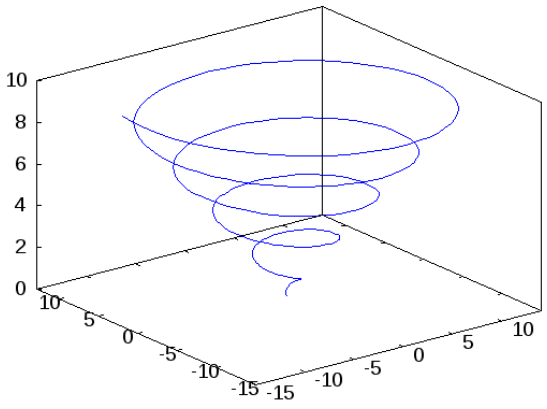


## Przykłady wykresów 3D

```
x = y = linspace(-3, 3, 51)';
[xx, yy] = meshgrid(x, y);
z = xx .* yy .* (xx .^ 2 - yy .^ 2) ./ (xx .^ 2 + yy .^ 2);
mesh(x, y, z)

x = y = linspace(-5, 5, 51)';
[xx, yy] = meshgrid(x, y);
z = cos(xx) .* cos(yy) .* exp(-sqrt(xx .^ 2 + yy .^ 2) ./ 4);
mesh(x, y, z)

x = y = linspace(-3, 3, 31)';
[xx, yy] = meshgrid(x, y);
z = -5 ./ (1 + xx .^ 2 + yy .^ 2);
mesh(x, y, z)

x = y = linspace(-5, 5, 51)';
[xx, yy] = meshgrid(x, y);
z = sin(xx .^ 2) + sin(yy .^ 2);
mesh(x, y, z)

x = y = linspace(-2, 2, 51)';
[xx, yy] = meshgrid(x, y);
z = sin(xx .^ 2) .* cos(yy .^ 2);
mesh(x, y, z)
```

## Krzywe parametryczne w 3D – przykłady

```
t = linspace(0, 20, 201);
x = sin(t);
y = cos(t);
z = t ./ 3;
plot3(x, y, z)
axis off

t = linspace(0, 30, 301);
x = sin(t) .* t ./ 2;
y = cos(t) .* t ./ 2;
z = t ./ 3;
plot3(x, y, z)
axis off

t = linspace(0, 30, 301);
x = sin(t) .^ 2;
y = cos(t) .* t ./ 2;
z = t ./ 3;
plot3(x, y, z)

t = linspace(0, 30, 301);
x = sin(t) .^ 2;
y = cos(t) .* 2;
z = t ./ 3;
plot3(x, y, z)
```

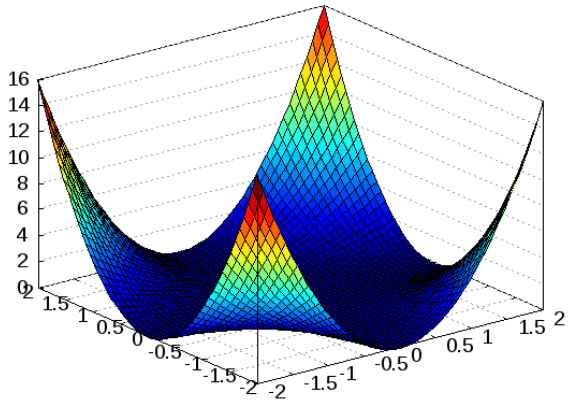





## Rysowanie wykresów 3D

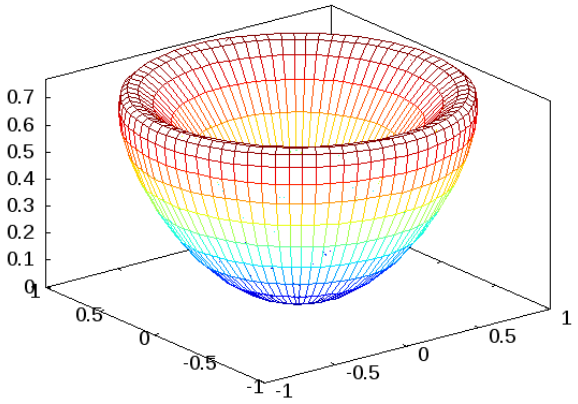
- Użycie funkcji surf zamiast mesh pozwala uzyskać wykres powierzchniowy, zamiast siatkowego.

```
x = y = linspace(-2, 2, 51)';
[xx, yy] = meshgrid(x, y);
z = (xx .^ 2) .* (yy .^ 2);
surf(x, y, z)
```



## Powierzchnie parametryczne

```
t = linspace(0, 2*pi, 81);
p = linspace(0, pi/2, 21);
[tt,pp] = meshgrid(t, p);
r = 2 * cos(pp);
x = r .* sin(pp) .* cos(tt);
y = r .* sin(pp) .* sin(tt);
z = r .* cos(pp);
mesh(x,y,z)

t = linspace(0, 2*pi, 81);
p = linspace(0, pi/2, 21);
[tt,pp] = meshgrid(t, p);
r = 2 * cos(pp);
x = r .* sin(pp) .* cos(tt);
y = r .* sin(pp) .* sin(tt);
z = r .* sin(pp);
mesh(x,y,z)

t = linspace(0, 2*pi, 81);
p = linspace(0, pi/2, 21);
[tt,pp] = meshgrid(t, p);
r = 2 * cos(pp);
x = r .* sin(pp) .* cos(tt);
y = r .* sin(pp) .* sin(tt);
z = r .* sin(pp) .* cos(pp);
mesh(x,y,z)
```






## Zapisywanie wykresu do pliku

Zapis utworzonego wykresu do pliku jest realizowany przez funkcję print:

```
print -d... nazwa-pliku
```

W miejsce `...` należy wpisać format graficzny, np.: eps, fig, png, jpg, gif, svg.

```
print -dfig wykres.fig
print -dpng wykres.png
```

Przed nazwą pliku można dodać opcje wydruku, np. -mono zamienia rysunek na czarno-biały,