

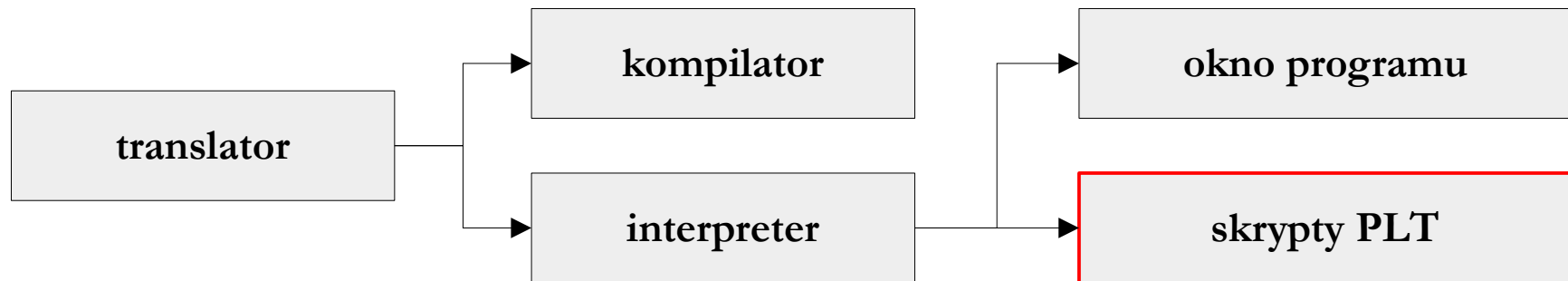


# Niezbędnik badacza: gnuplot

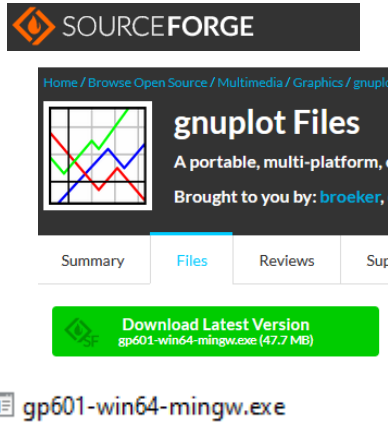
# Gnuplot

---

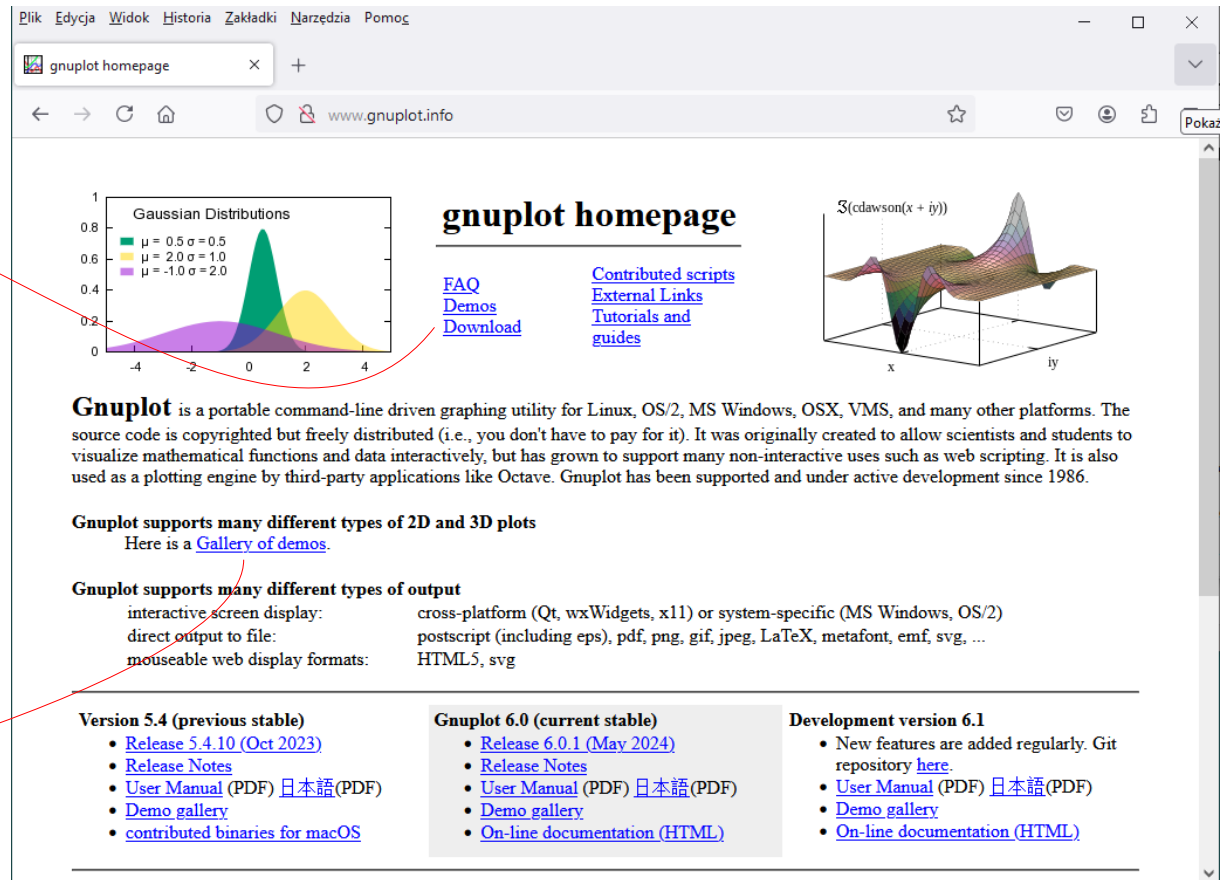
**gnuplot** – środowisko graficzne przeznaczone do tworzenia wykresów funkcji jednej lub dwóch zmiennych oraz do dopasowywania funkcji do danych. Gnuplot należy do programów typu **Open Source** i jest domyślnie wykorzystywany jako element postprocesingu w takich popularnych programach jak OpenFOAM czy YADE. Zaletą programu jest prostota łączenia różnych źródeł danych oraz łatwość generacji plików graficznych, między innymi w formacie EPS, wykorzystywanym np. podczas składania tekstów w środowisku TeX.



# Gnuplot – strona domowa



Windows



```
sudo apt-get install gnuplot
```

Linux/Ubuntu

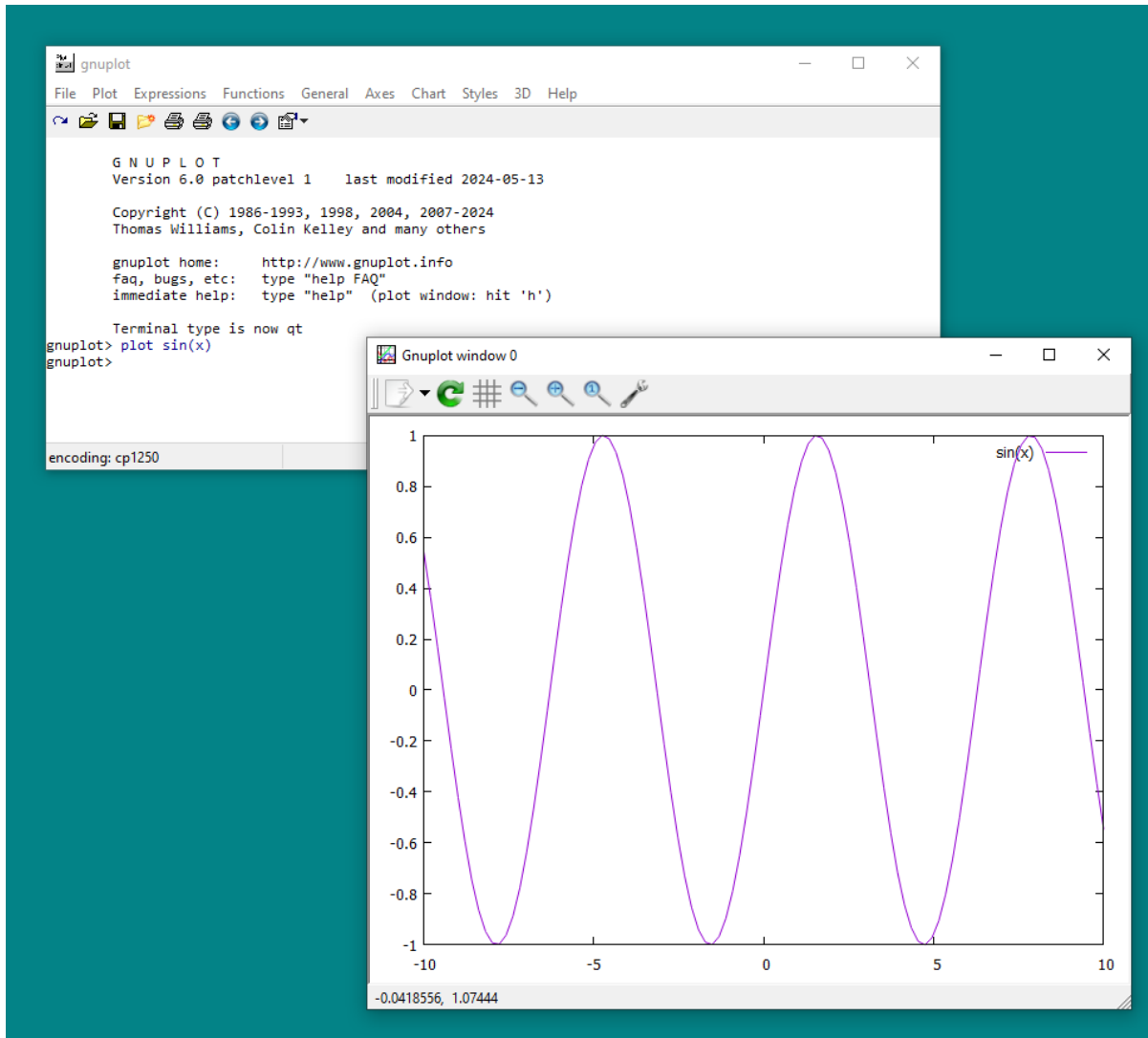
## 3D plots and surfaces

- [surfaces 1](#)
- [surfaces 2](#)
- [contours](#)
- [custom contour lines](#)
- [filled contours](#)
- [singularities](#)
- [hidden surfaces](#)
- [pm3d coloring](#)
- [pm3d hidden surfaces](#)
- [pm3d clipping](#)
- [2D projection of 3D surface](#)
- [shaded error region](#)
- [azimuth](#)
- [boxes](#)
- [circle and polygon 3D objects](#)
- [pm3d polygons](#)
- [lighting model](#)
- [hidden3d v. pm3d](#)

W sekcji „Gallery of demos” znajdują się przykłady zastosowań programu pogrupowane w tematyczne bloki.

<http://www.gnuplot.info/>

# Gnuplot – praca w oknie programu



Praca w oknie programu (Windows 10) – instrukcja generująca wykres przykładowej funkcji:

```
plot sin(x)
```

W systemach Windows okno programu uruchamia się plikiem **wgnuplot.exe**, znajdującym się w podkatalogu BIN.

Uwaga na nazwę pliku!

Lokalizacja pliku uruchamiającego program: C:\Program Files\gnuplot\bin\wgnuplot.exe

# Gnuplot – praca w oknie programu

```
Terminal - wojciech@wojciech-System-Product-Name: ~/Pulpit
Plik  Edycja  Widok  Terminal  Karty  Pomoc
wojciech@wojciech-System-Product-Name:~/Pulpit$ gnuplot

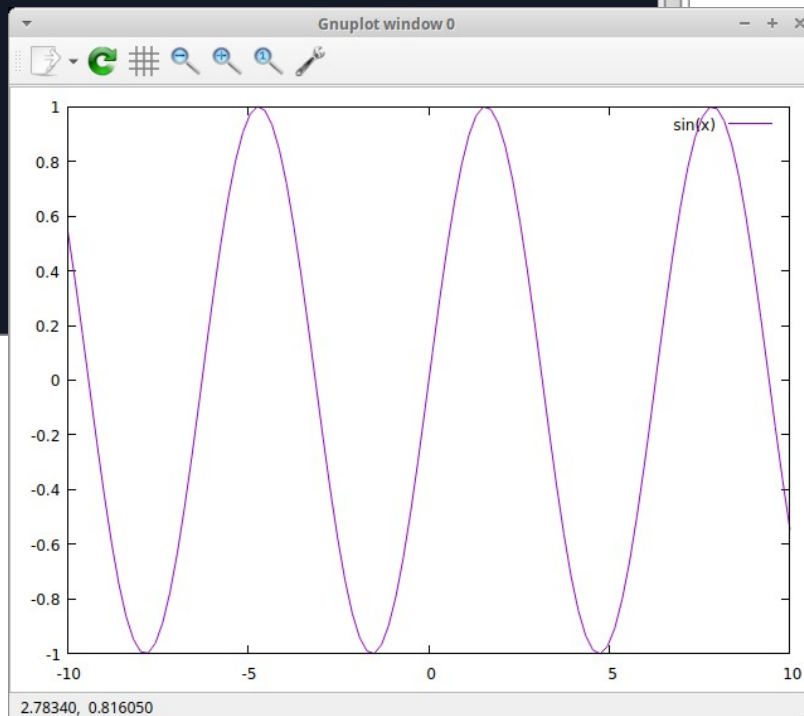
  G N U P L O T
  Version 5.4 patchlevel 2   last modified 2021-06-01

  Copyright (C) 1986-1993, 1998, 2004, 2007-2021
  Thomas Williams, Colin Kelley and many others

  gnuplot home:      http://www.gnuplot.info
  faq, bugs, etc:   type "help FAQ"
  immediate help:   type "help" (plot window: hit 'h')

Terminal type is now 'qt'
gnuplot> plot sin(x)
gnuplot> █
```

Tu jest wersja 5.4 ze względu na używanie starszej wersji xUbuntu.



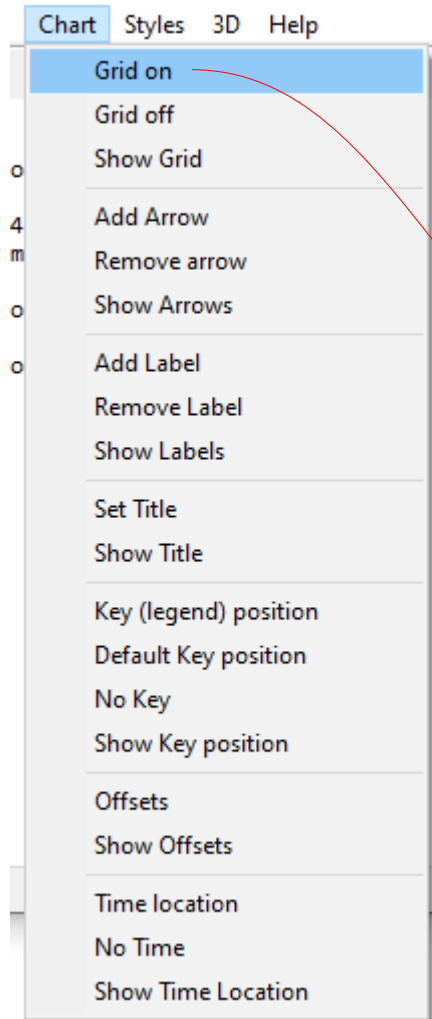
Praca w oknie programu (xUbuntu 22.04) – instrukcja generująca wykres przykładowej funkcji:

```
plot sin(x)
```

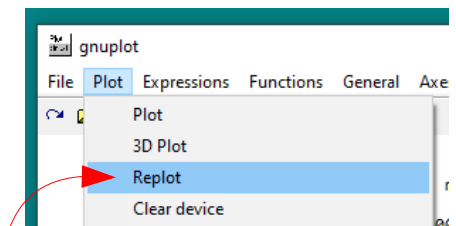
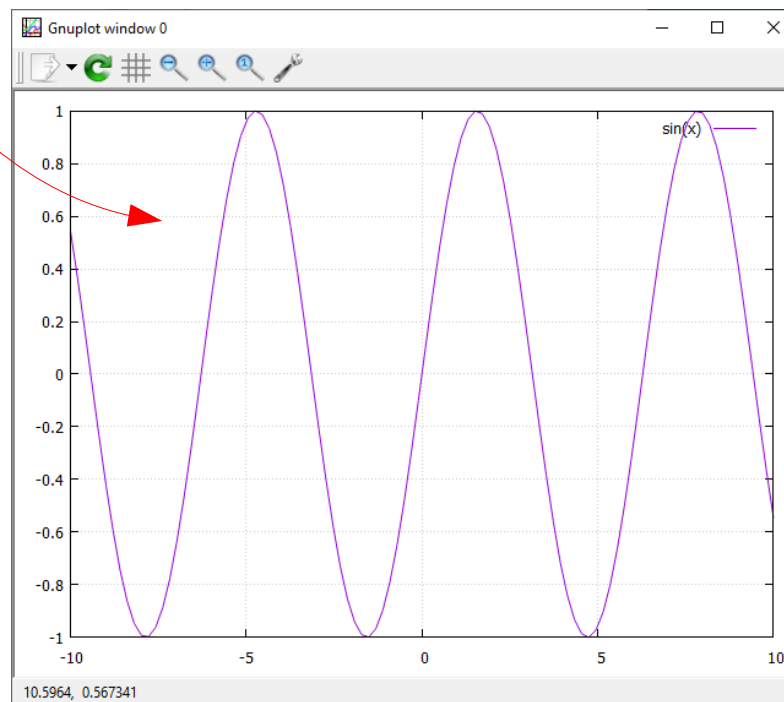
W systemach Linux program uruchamia się bezpośrednio w terminalu systemu operacyjnego. Polecenie uruchomienia programu ma postać:

```
gnuplot
```

# Gnuplot – praca w oknie programu



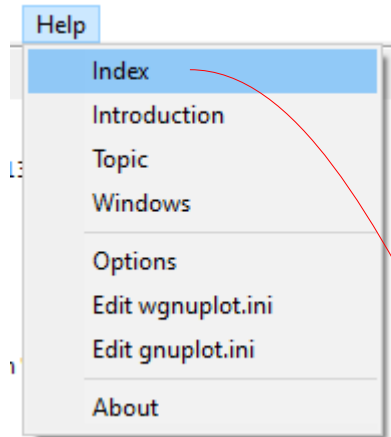
**Systemy Windows:** W pasku menu pogrupowane są najważniejsze polecenia – w przykładzie użyto opcji „Grid on”, która powoduje nałożenie na wykres delikatnej siatki, ułatwiającej odczytywanie danych.



replot

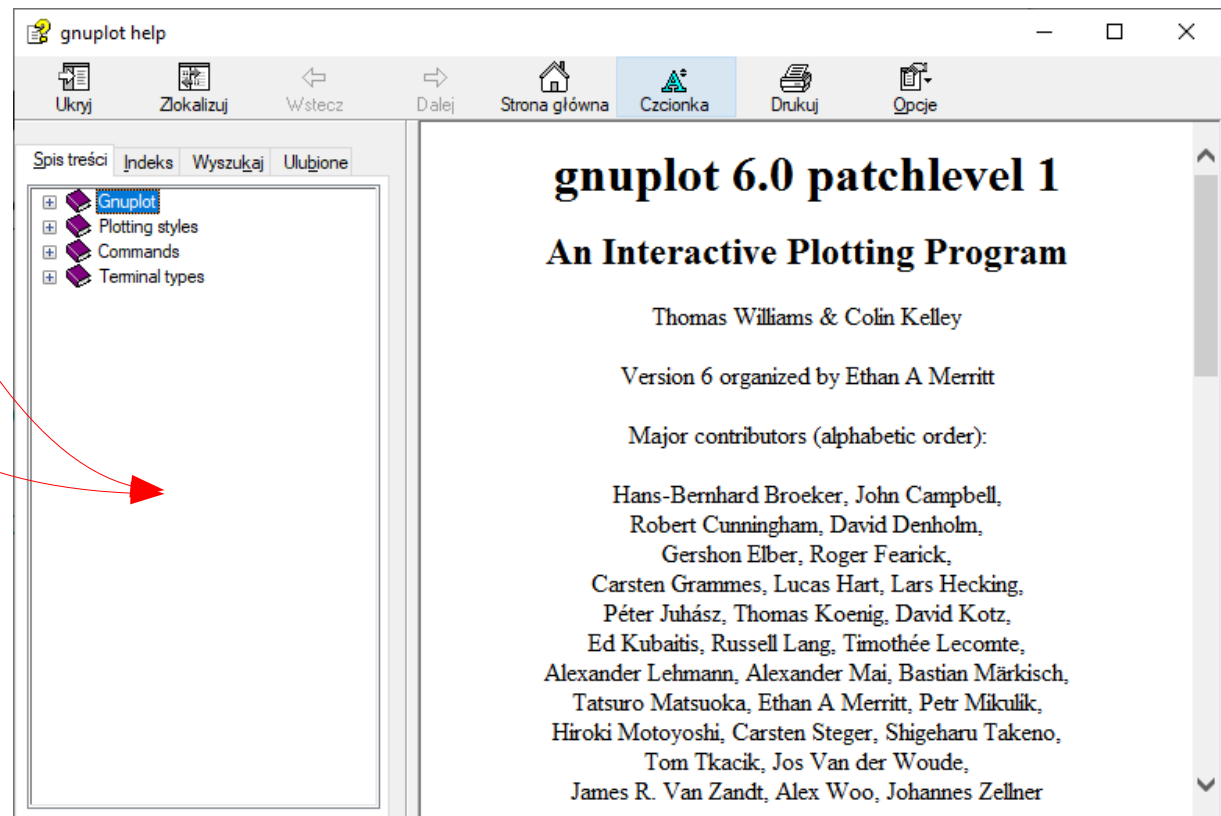
Aby zobaczyć zmiany, należy ponownie wygenerować wykres.

# Gnuplot – praca w oknie programu



**Systemy Windows:** Przydatnym narzędziem może być wbudowana pomoc w postaci pliku CHM (można go uruchamiać samodzielnie).

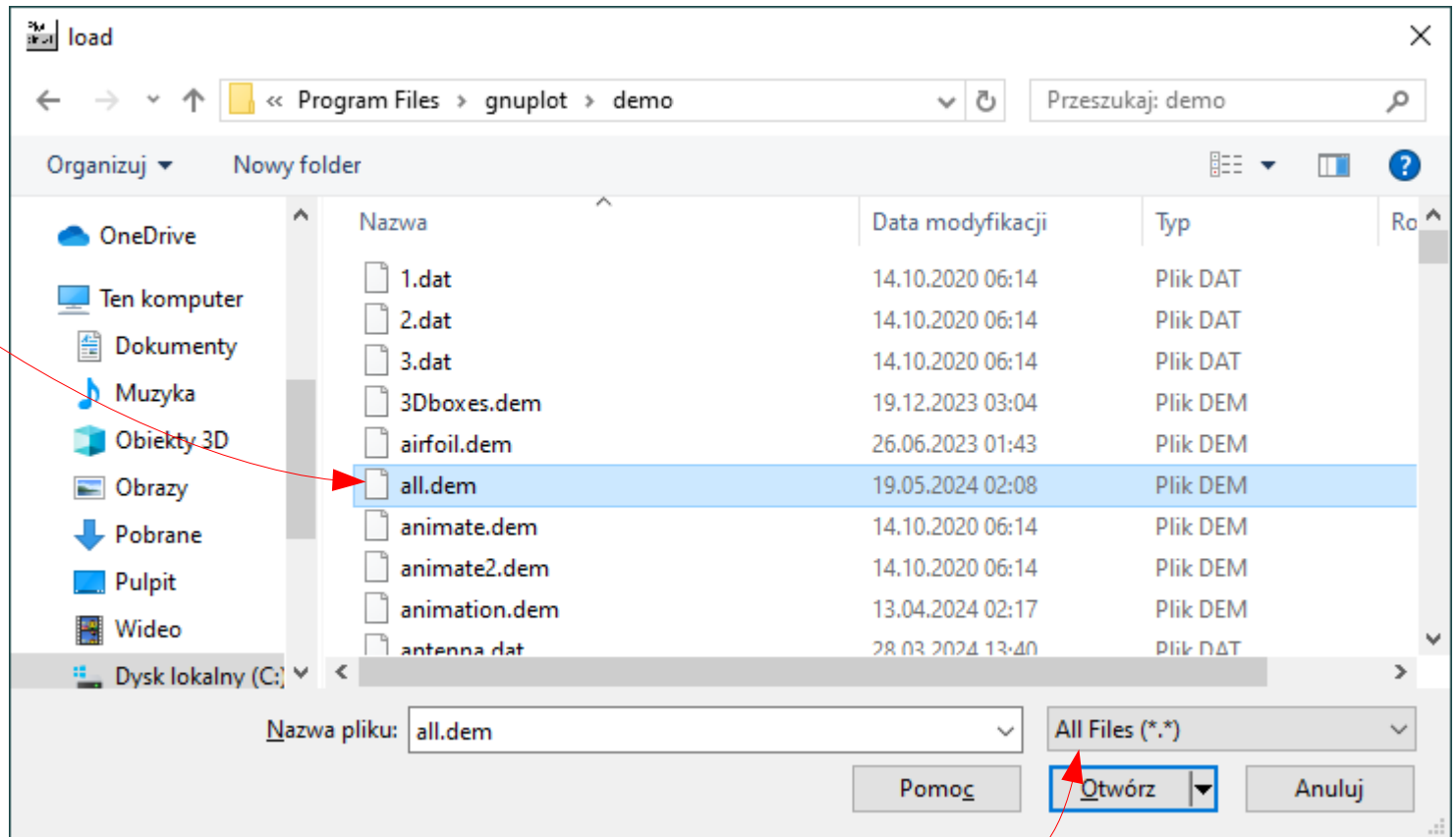
wgnuplot.chm  
wgnuplot.exe



# Gnuplot – praca w oknie programu



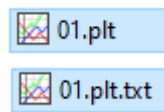
**Systemy Windows:** Przydatne może być uruchamianie i przeglądanie przykładowych skryptów, zlokalizowanych w katalogu DEMO.



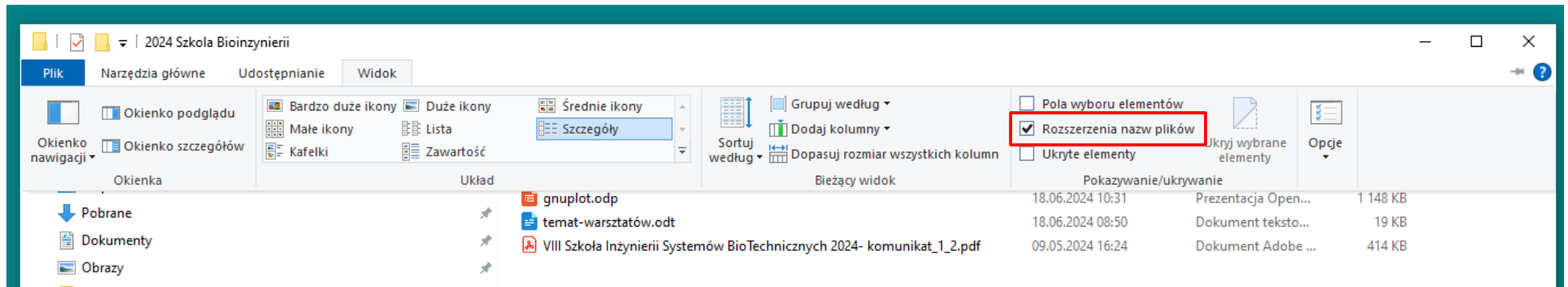
Aby zobaczyć skrypty, trzeba wybrać opcję „All Files (\*.\*)”.



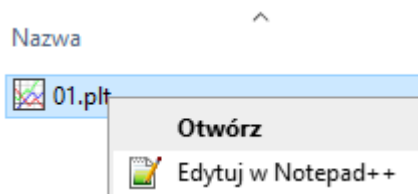
# Gnuplot – praca poprzez skrypty PLT



**Systemy Windows:** Aby tworzyć i edytować skrypty PLT należy włączyć widoczność rozszerzeń nazw plików (**ważne!**).



Do tworzenia i edycji skryptów PLT należy posiadać jakiś edytor plików tekstowych: notatnik, mousepad, notepad++, ....



```
1 plot sin(x)
2 pause mouse
3 gnuplot exit
```

Po każdej zmianie trzeba zapisać plik przed ponownym uruchomieniem skryptu!

# Gnuplot – materiały szkoleniowe

Cwiczenia  
Oprogramowanie  
Cwiczenia.zip  
sobieski-gnuplot.odp  
sobieski-gnuplot.pdf

edi-3.1.exe  
Gnuplot\_6.pdf  
gp601-win64-mingw.exe  
gs10031w64.exe  
irfanview\_lang\_polski.exe  
iview467\_plugins\_x64\_setup.exe  
iview467\_x64\_setup.exe  
npp.8.6.9.Installer.x64.exe  
python-3.12.5-amd64.exe

Z1 16.plt  
Z2 17.plt  
Z3 18.plt  
Z4 19.plt  
Z5 20.plt  
01.plt 21.plt  
02.plt 22.dat  
03.plt 22.plt  
04.plt 23.plt  
05.plt 24.plt  
06.dat 25.plt  
06.exe 26.plt  
06.exp 27.plt  
06.f90 28.plt  
06.plt 28.sum  
07.plt 29.exp  
08.plt 29.plt  
09.plt 29.txt  
10.plt 30.dat  
11.plt 30.exe  
12.plt 30.f90  
13.plt 30.plt  
14.plt 31.py  
15.plt

**sobieski-gnuplot** – prezentacja

**Cwiczenia** – zbiór 31 ćwiczeń praktycznych

**Z1-Z5** – przykłady rysunków z publikacji

**Cwiczenia.zip** – kopia ćwiczeń

W trakcie ćwiczeń sugeruje się nie tworzyć od nowa własnych skryptów, ale edytować i modyfikować te już istniejące – jak się coś popsuje, to kopia oryginału będzie cały czas dostępna.

# Gnuplot – terminale

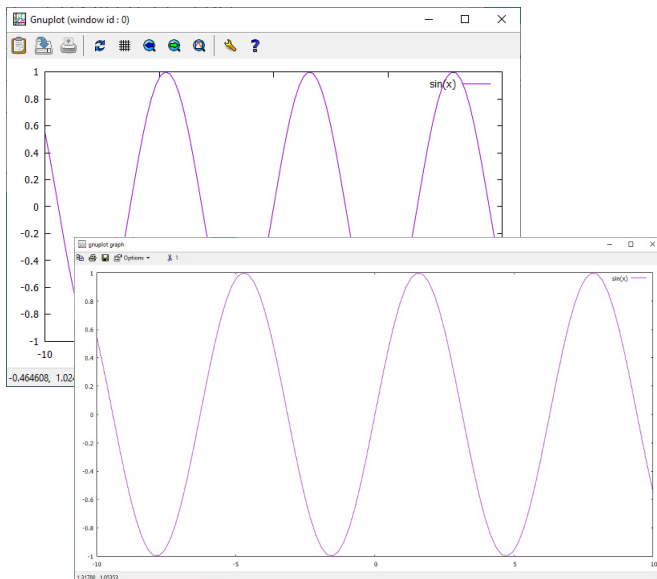
01:

```
set terminal wxt
plot sin(x)
pause mouse
gnuplot exit
```

**set terminal** – komenda definiuje sposób, w jaki gnuplot generuje wykresy. Terminale mogą być używane do bezpośredniego wyświetlania wykresów na ekranie (np.: windows, **wxt**, qt, x11) lub do zapisu wykresu do pliku (np.: postscript, pdf, **pngcairo**, svg, canvas).

Polecany terminal – działa tak samo w systemach Windows oraz Linux.

wxt

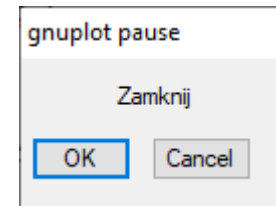


windows

**plot** – komenda generuje wykres 1 zmiennej; posiada wiele opcji umożliwiających sterowanie wyglądem wyświetlanych danych.

**pause** – komenda wstrzymuje działanie programu gnuplot, z opcjonalnymi parametrami zależnymi od używanego terminala (np. **pause mouse** powoduje czekanie na interakcję z myszą).

```
pause mouse
pause 3
pause -1 "Zamknij"
```



**gnuplot exit** – komenda kończy działanie programu gnuplot.

# Gnuplot – terminale

02:

```
set terminal pngcairo
set output '02.png'
plot sin(x)
#pause mouse #polecenie nie ma sensu
gnuplot exit
```

Jeżeli wybrany zostanie terminal generujący wykres do pliku, to trzeba podać nazwę tego pliku – służy do tego komenda **set output**.

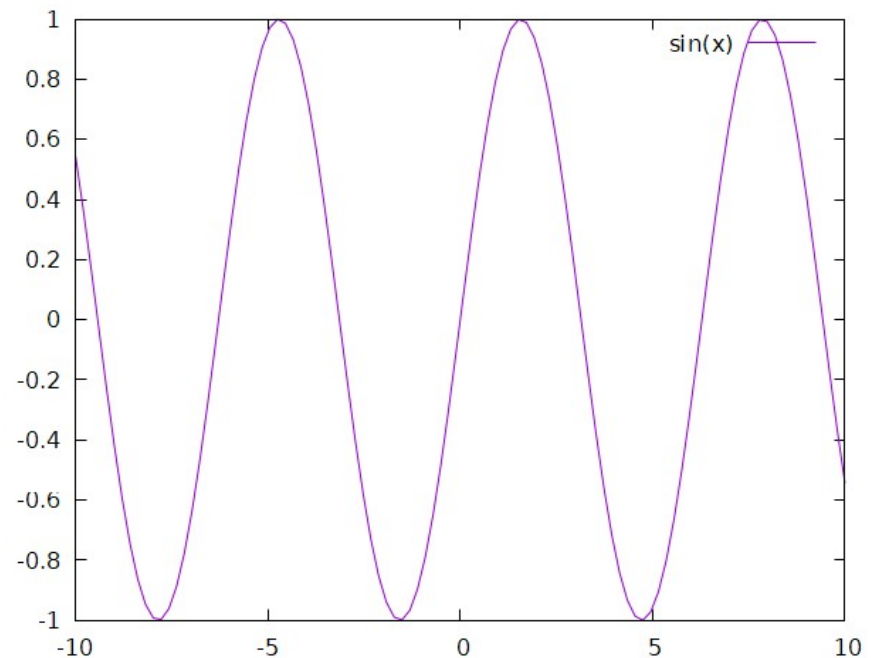
Komentarze rozpoczyna się znakiem #.

Nazwa

- 01.plt
- 02.plt
- 02.png

Zmiana terminala wymaga przeważnie zmiany rozszerzenia pliku wynikowego.

[http://gnuplot.info/docs\\_6.0/Terminals.html](http://gnuplot.info/docs_6.0/Terminals.html)



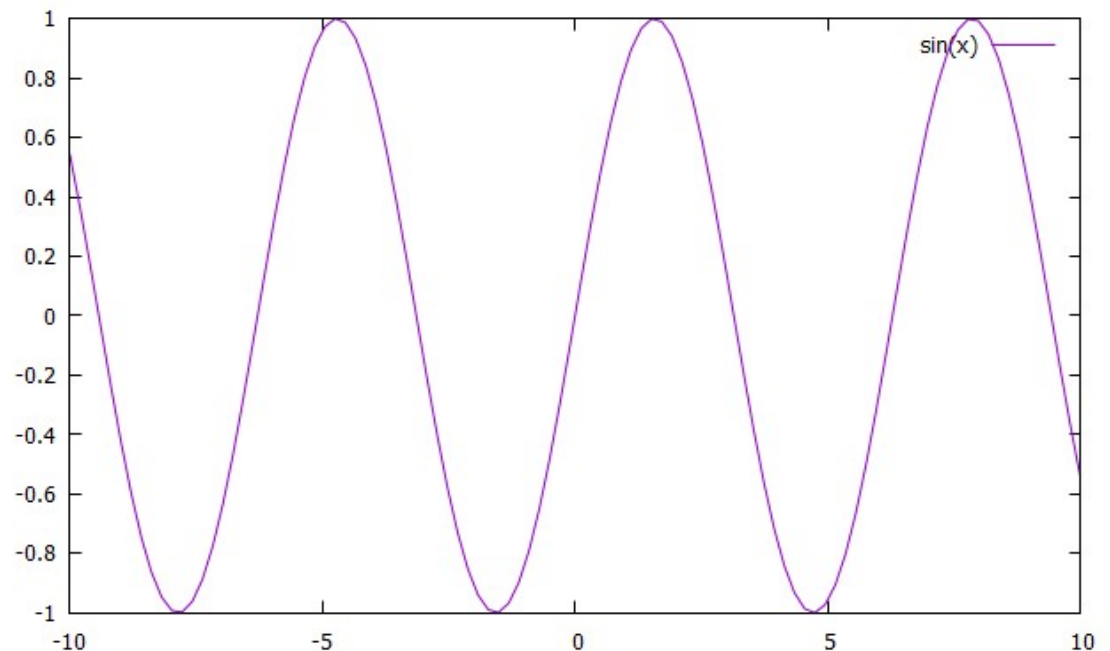
# Gnuplot – terminale

03:

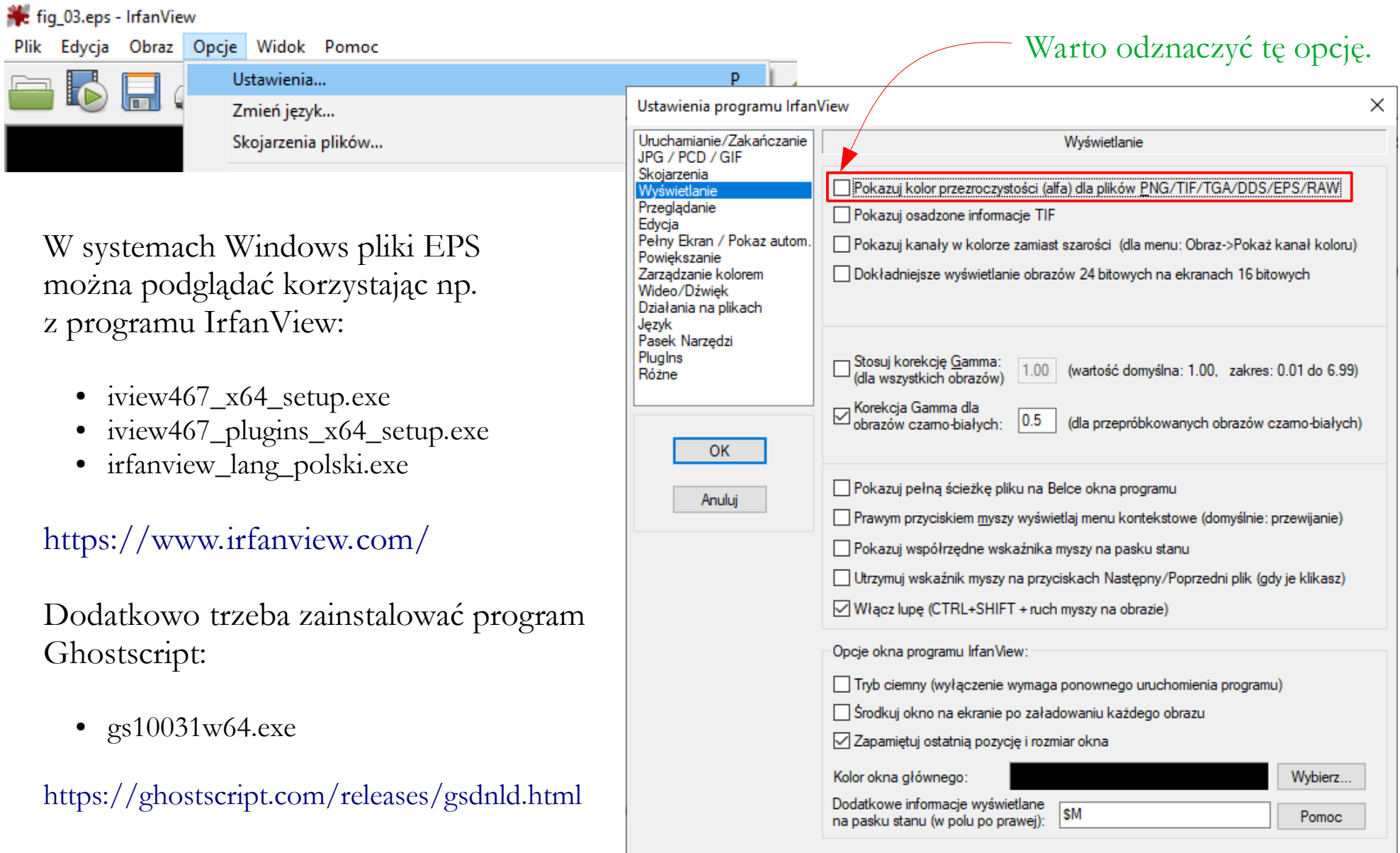
```
→ set terminal wxt  
plot sin(x)  
pause mouse  
set terminal pngcairo  
set output '03.png'  
→ replot  
set terminal postscript colour  
set output '03.eps'  
→ replot  
gnuplot exit
```

Do wizualizacji plików EPS w systemach Windows można wykorzystać np. program IrfanView.

Terminale można przełączać dowolnie w trakcie interpretacji pojedynczego skryptu PLT. Po każdej zmianie terminala należy ponownie wygenerować wykres instrukcją **plot**, lub – co jest polecane – poleceniem **replot**.



# Gnuplot – terminale



The image shows the IrfanView application window with the 'Opcje' menu open and the 'Ustawienia programu IrfanView' dialog box displayed. The 'Wyświetlanie' tab is selected, and the checkbox 'Pokaż kolor przezroczystości (alfa) dla plików PNG/TIF/TGA/DDS/EPS/RAW' is highlighted with a red box and a red arrow. A green text annotation 'Warto odznaczyć tę opcję.' points to the checkbox.

W systemach Windows pliki EPS można podglądać korzystając np. z programu IrfanView:

- `iview467_x64_setup.exe`
- `iview467_plugins_x64_setup.exe`
- `irfanview_lang_polski.exe`

<https://www.irfanview.com/>

Dodatkowo trzeba zainstalować program Ghostscript:

- `gs10031w64.exe`

<https://ghostscript.com/releases/gsdnld.html>

# Gnuplot – terminale

```

sobieski.tex - TeXworks
Plik Edytuj Znajdź Format Składaj Skrypty Okno Pomoc
pdfLaTeX+MakeIndex+BiBTeX
\label{eq12}
\end{equation}

where fit parameters $a$, $b$, $c$, and $d$ are equal to 1.07, 0.58, -0.21 and 1.2, respectively.

In Fig. \ref{fig14} (bottom), all the obtained values of the LBM tortuosity for the highest grid and for all repetitions are shown together with the results of the PSA analysis. The small, filled points represent the values of the LBM tortuosity. Circles mean that in the current case at least one free path exists in the X-direction. Squares are shown in cases in which the LBM tortuosity has a negative value.

\begin{figure*}
\centering
\includegraphics[scale=0.3,angle=-90]{Fig_14a.eps}
\includegraphics[scale=0.3,angle=-90]{Fig_14f.eps}
\caption{Distribution of the LBM tortuosity: all values of LBM tortuosity between 1.0 and 5.0 (upper); comparison with PSA method (bottom)}
\label{fig14}
\end{figure*}

It can be noted that if free paths exist in the main flow direction, then the LBM tortuosity has a value greater or equal to 1. Only in two cases the LBM tortuosity is negative (Fig. \ref{fig16}), regardless of the existence of free paths in the pore structure. In these cases, the flow in the X-direction is strongly limited and the lattice gas flows mainly in the Y-direction or returns. In both cases the size of the

```

Figure 14: Distribution of the LBM tortuosity: all values of LBM tortuosity between 1.0 and 5.0 (upper); comparison with PSA method (bottom)

$$\tau(\rho_p) = a + b \cdot \exp(c \cdot \rho_p^d), \quad (12)$$

where fit parameters  $a$ ,  $b$ ,  $c$ , and  $d$  are equal to 1.07, 0.58, -0.21 and 1.2, respectively.

In Fig. 14 (bottom), all the obtained values of the LBM tortuosity for the highest grid and for all repetitions are shown together with the results of the PSA analysis. The small, filled points represent the values of the LBM tortuosity. Circles mean that in the current case at least one free path exists in the X-direction. Squares are shown in cases in which the LBM tortuosity has a negative value.

It can be noted that if free paths exist in the main flow direction, then the LBM tortuosity has a value greater or equal to 1. Only in two cases the LBM tortuosity is negative (Fig. 16), regardless of the existence of free paths in the pore structure. In these cases, the flow in the X-direction is strongly limited and the lattice gas flows mainly in the Y-direction or returns. In both cases the size of the structural element is high.

18

Przykład wykorzystania plików EPS w TeXie (tu w programie MikTeX).



# Gnuplot – źródła danych

04:

```
set terminal wxt
plot sin(x), cos(x)
pause mouse
gnuplot exit
```

lub (co jest wygodniejsze w praktyce)

```
set terminal wxt
plot sin(x), \
      cos(x)
pause mouse
gnuplot exit
```

**UWAGA:** Po znaku „\” nie może być żadnego innego znaku, nawet spacji!

Komendy **plot** się nie powtarza.

Na końcu listy nie wstawia się znaku „\”.

Źródłem danych mogą być **funkcje wewnętrzne**: lista funkcji wewnętrznych dostępna jest np. w oficjalnej dokumentacji programu gnuplot.

## gnuplot 6.0

An Interactive Plotting Program

Thomas Williams & Colin Kelley

gnuplot 6.0

39

### Functions

Arguments to math functions in **gnuplot** can be integer, real, or complex unless otherwise noted. Functions that accept or return angles (e.g.  $\sin(x)$ ) treat angle values as radians, but this may be changed to degrees using the command `set angles`.

Math library and built-in functions		
Function	Arguments	Returns (c indicates complex result)
abs(x)	int or real	absolute value of $x$ , $ x $
abs(x)	complex	length of $x$ , $\sqrt{\text{real}(x)^2 + \text{imag}(x)^2}$
acos(x)		c $\cos^{-1} x$ (inverse cosine)
acosh(x)		c $\cosh^{-1} x$ (inverse hyperbolic cosine)
airy(x)	real	Airy function $\text{Ai}(x)$ for real $x$
arg(x)	complex	the phase of $x$ , $-\pi \leq \arg(x) \leq \pi$
asin(x)		c $\sin^{-1} x$ (inverse sin)
asinh(x)		c $\sinh^{-1} x$ (inverse hyperbolic sin)
atan(x)		c $\tan^{-1} x$ (inverse tangent)

[http://www.gnuplot.info/docs\\_6.0/Gnuplot\\_6.pdf](http://www.gnuplot.info/docs_6.0/Gnuplot_6.pdf)



# Gnuplot – źródła danych

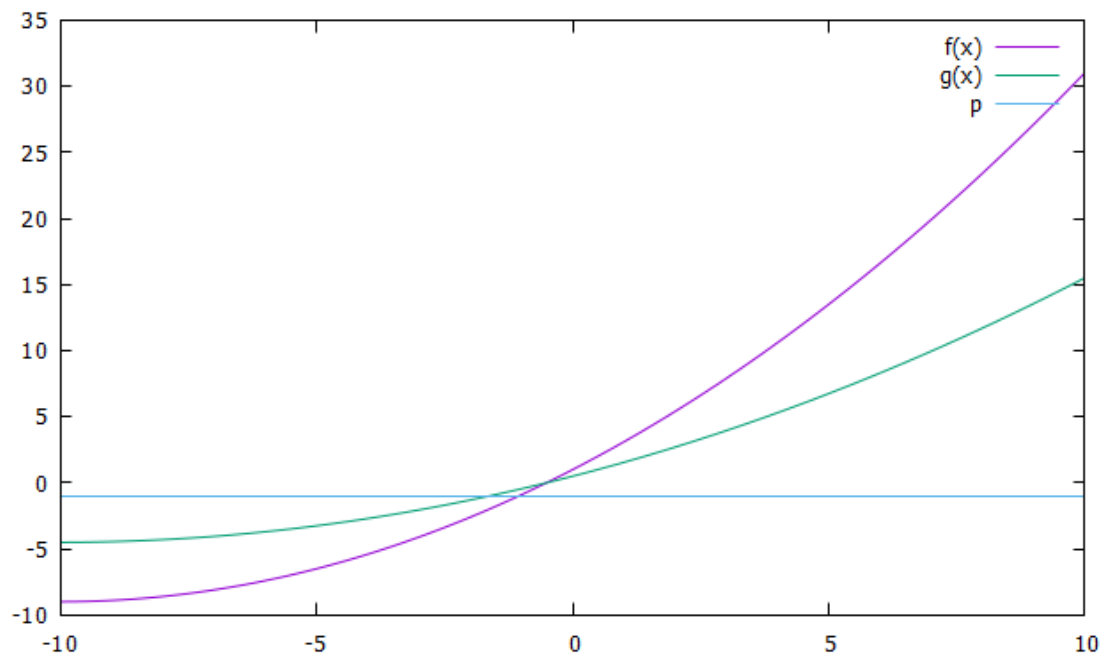
05:

```
set terminal wxt
p = -1.0
f(x) = 0.1*x**2.0 + 2*x - p
g(x) = f(x)/2.0
plot f(x), \
      g(x), \
      p
pause mouse
gnuplot exit
```

Tu wszystkie ustawienia są domyślne (poza terminalem), ale praktycznie każda komenda gnuplota zawiera opcje umożliwiające jej modyfikację.

Źródłem danych mogą być **funkcje użytkownika**:

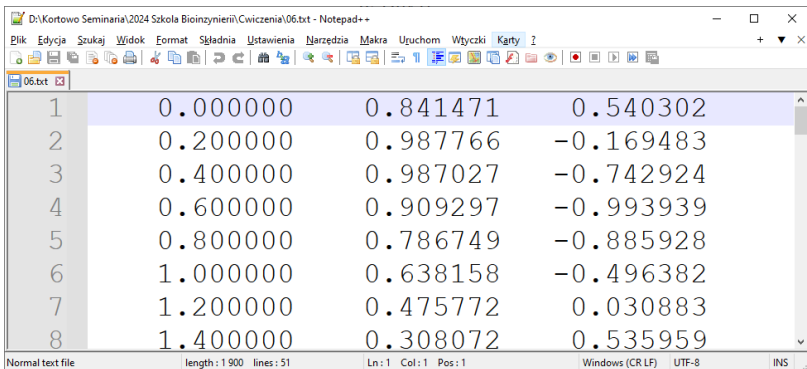
- funkcje muszą mieć postać: **nazwa(x)**
- nazwy funkcji muszą być unikalne
- w jednej funkcji można wywoływać inną
- do opisu funkcji można definiować dodatkowe stałe o postaci: **stała = wartość**
- stałe również można wyświetlać na rysunku



# Gnuplot – źródła danych

06:

```
set terminal wxt
plot '06.dat'
pause mouse
gnuplot exit
```



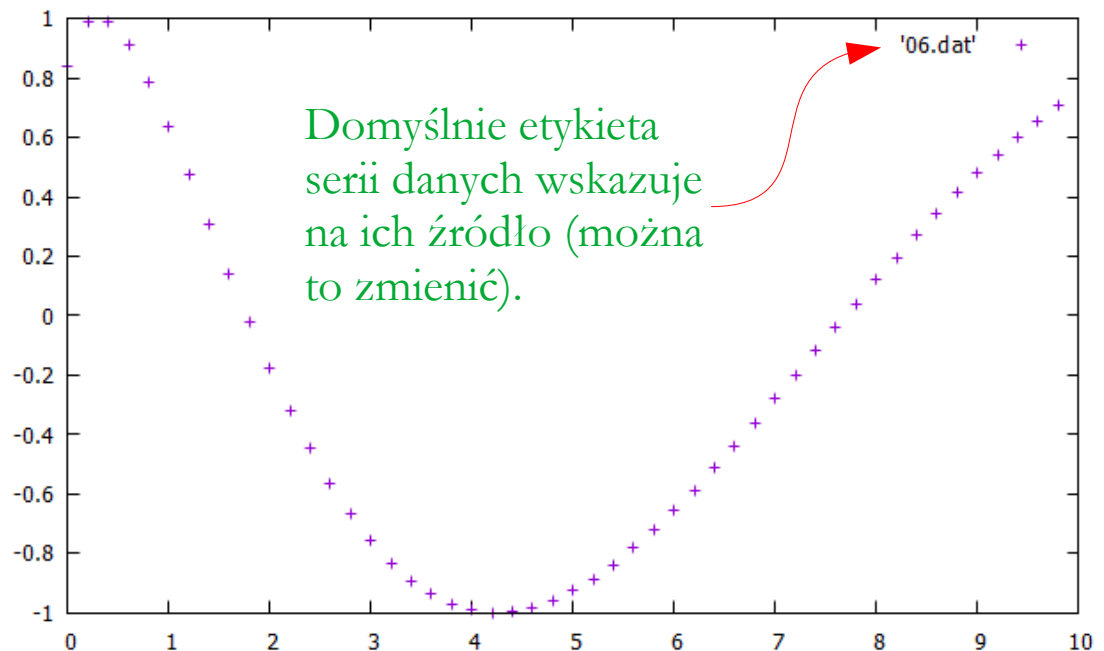
Line	Column 1	Column 2	Column 3
1	0.000000	0.841471	0.540302
2	0.200000	0.987766	-0.169483
3	0.400000	0.987027	-0.742924
4	0.600000	0.909297	-0.993939
5	0.800000	0.786749	-0.885928
6	1.000000	0.638158	-0.496382
7	1.200000	0.475772	0.030883
8	1.400000	0.308072	0.535959

Rozszerzenie nazwy pliku z danymi może być dowolne, ważne jest tylko, aby był to zwykły plik tekstowy (plik ASCII).

Zamiast rozszerzenia TXT korzystniej jest użyć takiego, który wskazuje na zawartość pliku, np. DAT.

Źródłem danych mogą być **pliki tekstowe**:

- nazwy plików nie mogą zawierać znaków narodowych ani spacji
- dane w plikach muszą być ułożone w kolumnach oddzielonych spacjami lub tabulatorami
- domyślnie brane są pod uwagę 2 pierwsze kolumny w pliku



# Gnuplot – źródła danych

---

**07:**

```
set terminal wxt
plot '-'
1 1
2 4
3 9
4 16
e #koniec wprowadzania danych
pause mouse
gnuplot exit
```

**08:**

```
set terminal wxt
array data[4]
data[1] = 1
data[2] = 4
data[3] = 9; data[4] = 16
plot for [i=1:4] data[i]
pause mouse
gnuplot exit
```

Źródłem danych mogą być **zmiennie proste wprowadzone ręcznie**:

- dane muszą mieć postać taką, jakby były zawarte w pliku tekstowym
- obszar wprowadzania danych rozpoczyna się sekwencją '-'
- obszar wprowadzania danych kończy się literą **e**

Źródłem danych mogą być **zmiennie indeksowane (tablice) wprowadzone ręcznie**:

- przed podaniem danych należy zadeklarować tablicę komendą **array**, tzn. podać jej nazwę i liczbę elementów
- dane można wprowadzać w kolejnych wierszach lub w jednej linii, rozdzielając je znakiem średnika

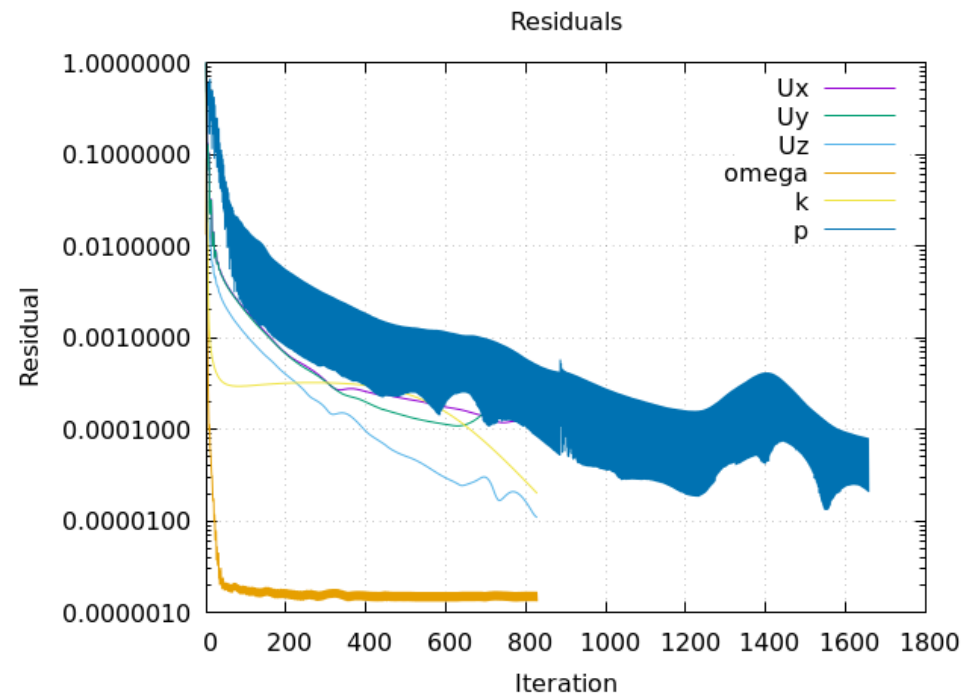
# Gnuplot – źródła danych

**xx:**

```
set logscale y
set title "Residuals"
set ylabel 'Residual'
set xlabel 'Iteration'
plot "< cat log | grep 'Solving for Ux' | cut -d' ' -f9 | tr -d ',' title 'Ux' with lines,\
"< cat log | grep 'Solving for Uy' | cut -d' ' -f9 | tr -d ',' title 'Uy' with lines,\
"< cat log | grep 'Solving for Uz' | cut -d' ' -f9 | tr -d ',' title 'Uz' with lines,\
"< cat log | grep 'Solving for omega' | cut -d' ' -f9 | tr -d ',' title 'omega' with lines,\
"< cat log | grep 'Solving for k' | cut -d' ' -f9 | tr -d ',' title 'k' with lines,\
"< cat log | grep 'Solving for p' | cut -d' ' -f9 | tr -d ',' title 'p' with lines
pause 10
reread
```

Źródłem danych mogą być **funkcje powłoki systemu operacyjnego** – szczegóły zależą od użytej powłoki.

Przykład skryptu przechwytyjącego wartości rezyduów wyświetlanych w terminalu systemu operacyjnego (xUbuntu) podczas wykonywania obliczeń w programie OpenFOAM.



# Gnuplot – źródła danych

09:

```
set terminal wxt
f(x) = sin(x)
plot '06.dat' u 1:($2/2), \
      '06.dat' u 1:3, \
      '06.exp' u 1:2, \
      f(x)
pause mouse
gnuplot exit
```

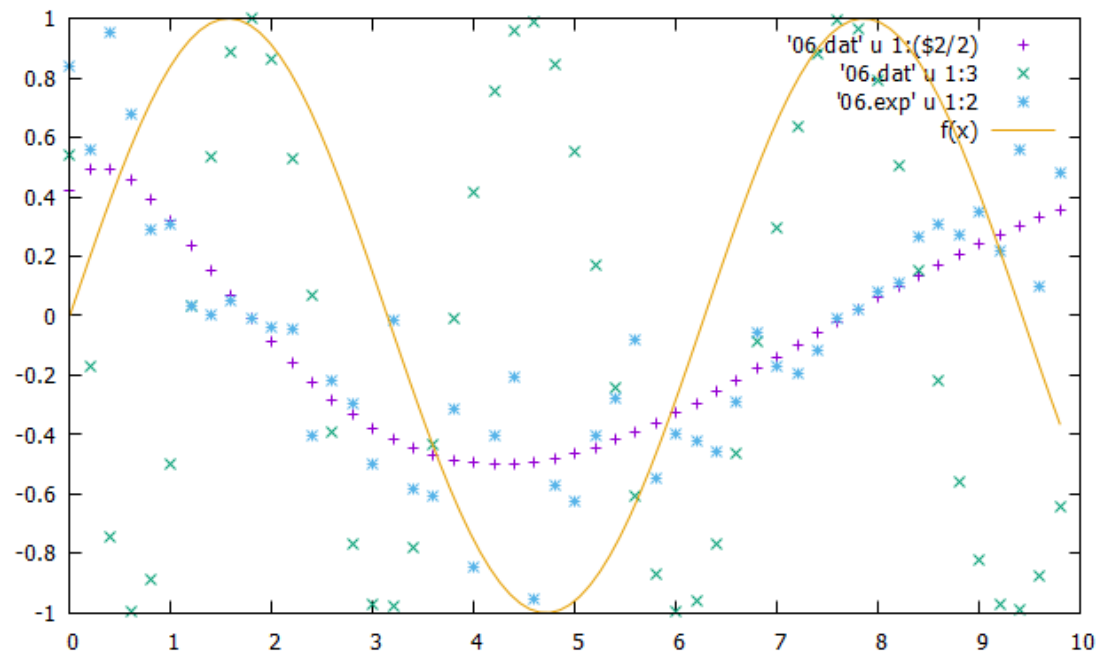
**u** (od use) – wskazanie, które kolumny w pliku mają być źródłem danych

**\$** – znak powodujący, że dane traktowane są jako zmienne, na których można wykonywać działania matematyczne (np. skalowanie wartości) – tu: dzielenie przez 2

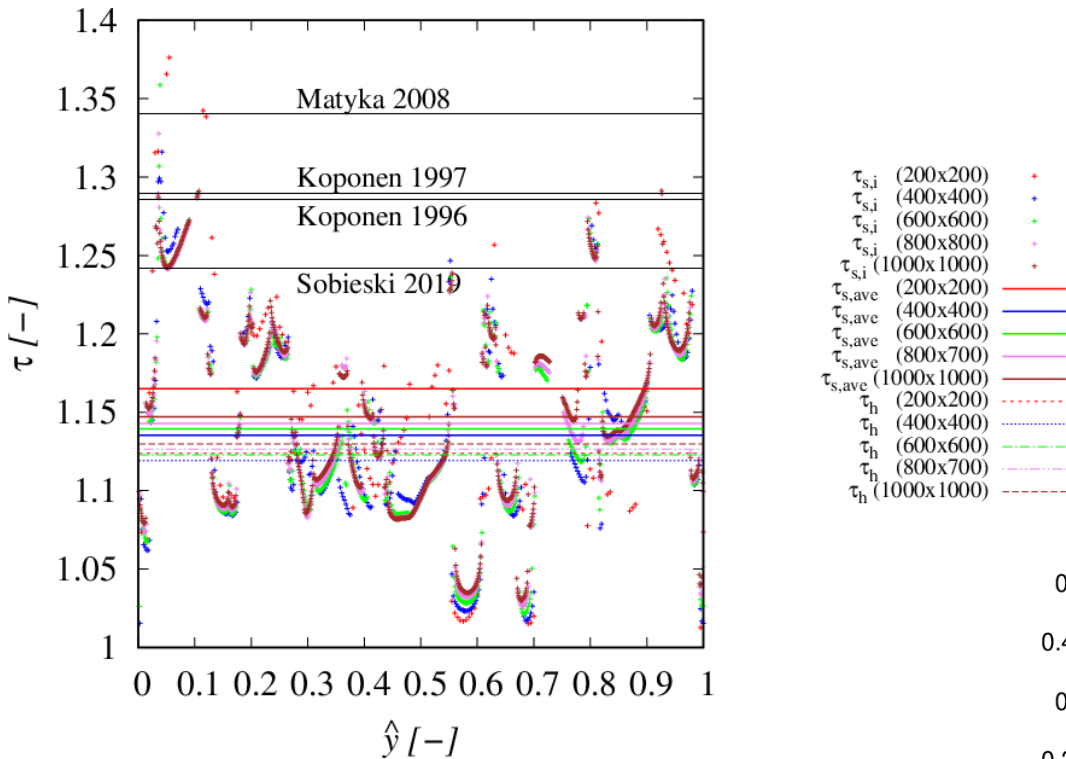
Wykres jest mało czytelny –  
trzeba popracować nad  
sposobem prezentacji  
poszczególnych serii danych.

Ogromną zaletą gnuplota jest możliwość łączenia wykresów generowanych na podstawie różnych źródeł danych (**uwaga na nazwy i rozszerzenia!**):

- można porównywać dane z modelu i eksperymentu
- można porównywać różne eksperymenty
- można porównywać różne modele
- ...



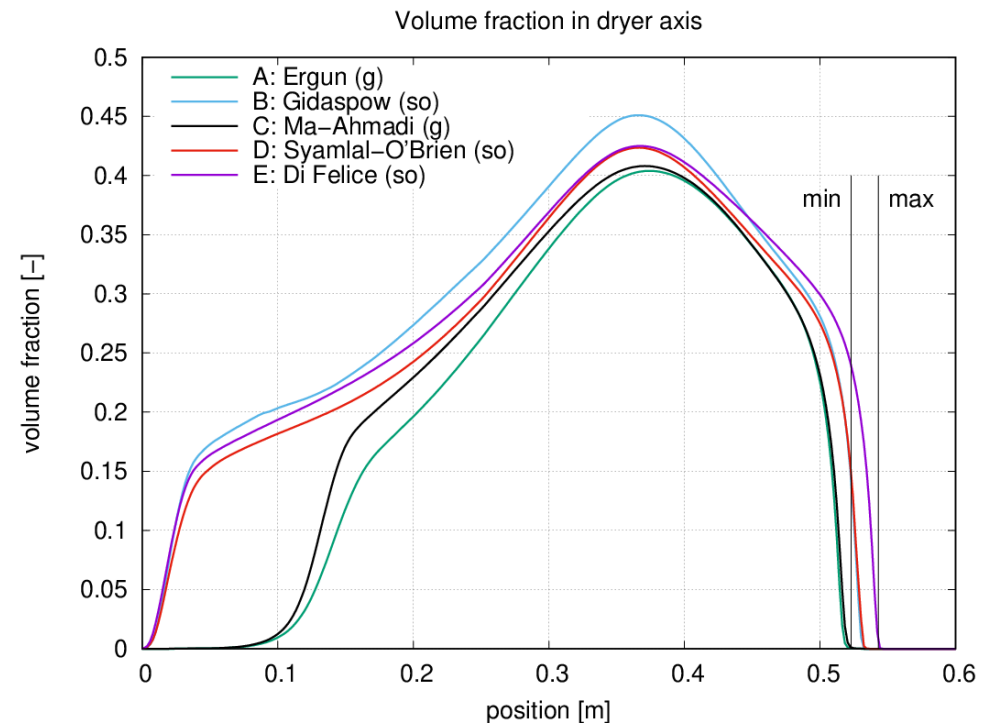
# Gnuplot – źródła danych



**Z2:** Porównanie rozkładu fazy granularnej w osi fontannowej suszarki do ziarna uzyskane w programie ANSYS Fluent dla różnych zestawów domknięć.

Przykłady łączenia źródeł danych.

**Z1:** Badania krętości linii prądu – porównanie wartości lokalnych (punkty) z wartościami średnimi (kolorowe linie ciągłe), alternatywną metodą obliczeniową (kolorowe linie przerywane) oraz wybranymi formułami empirycznymi (linie czarne).



# Gnuplot – źródła danych

**Z3:**

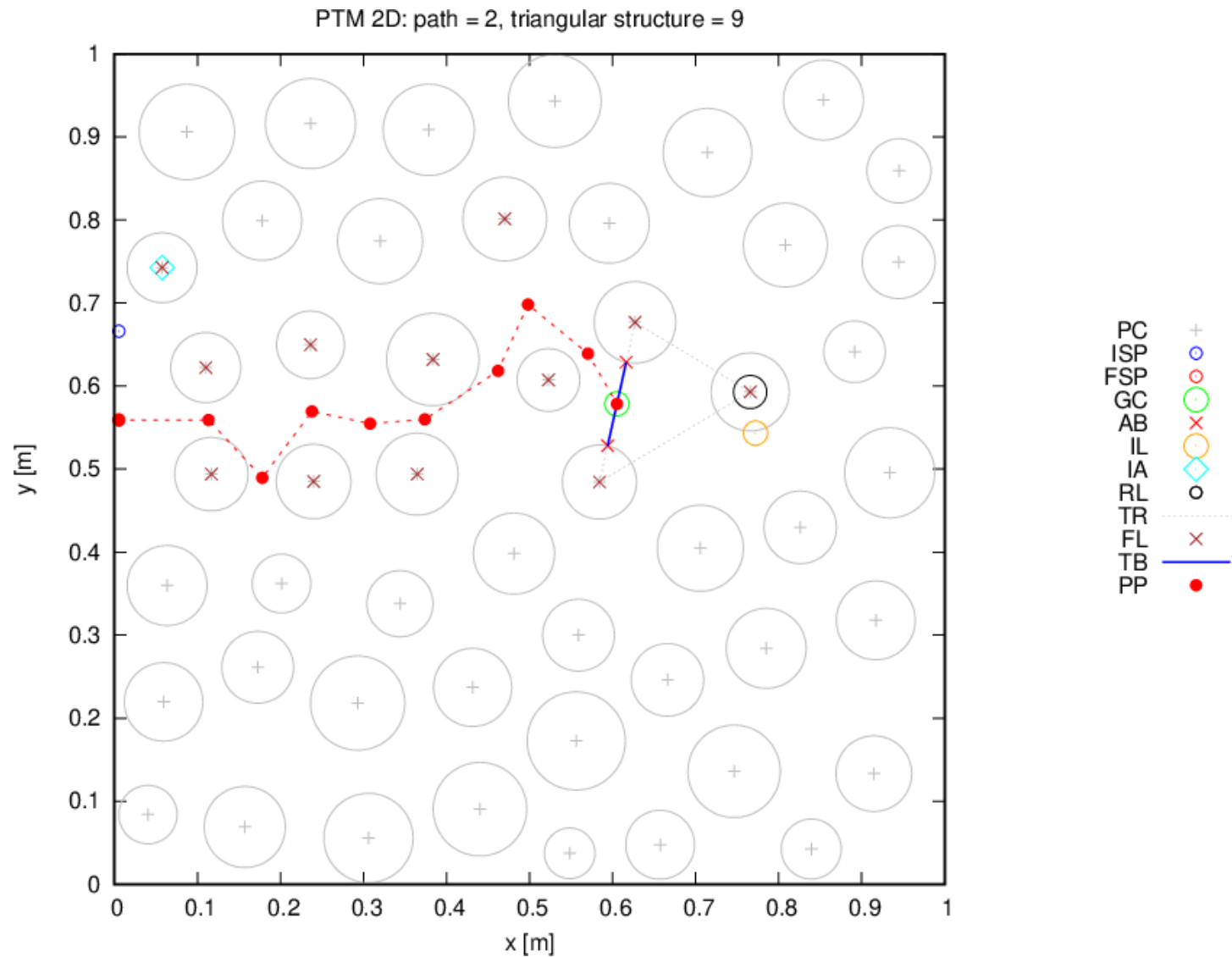
```
set terminal postscript colour enhanced font 'Arial, 12'  
set output '0.40_200_200_01_01_000002_000009.eps'  
set size ratio 1  
set title 'PTM 2D: path = 2, triangular structure = 9'  
set xlabel 'x [m]'  
set ylabel 'y [m]'  
set key outside right center  
plot '0.40_200_200_01_01.dat' u 1:2 title 'PC' with points pt 1 ps 1 lt 1 lc 'grey', \\  
    '0.40_200_200_01_01.dat' u 1:2:(0.5*$3) notitle with circles lt 1 lc 'grey', \\  
    '0.40_200_200_01_01.isp' title 'ISP' with points pt 6 ps 1 lt 1 lc 'blue', \\  
    '0.40_200_200_01_01.fsp' title 'FSP' with points pt 6 ps 1 lt 1 lc 'red', \\  
    '0.40_200_200_01_01.gc' title 'GC' with points pt 6 ps 2 lc 'green', \\  
    '0.40_200_200_01_01.AB' title 'AB' with points pt 2 ps 1 lc 'red', \\  
    '0.40_200_200_01_01.il' title 'IL' with points pt 6 ps 2 lc 'orange', \\  
    '0.40_200_200_01_01.of' title 'OF' with points pt 4 ps 2 lt 1 lc 'pink', \\  
    '0.40_200_200_01_01.oa' title 'IA' with points pt 12 ps 2 lt 1 lc 'cyan', \\  
    '0.40_200_200_01_01.rl' title 'RL' with circles lt 1 lc 'black', \\  
    '0.40_200_200_01_01.tr' title 'TR' with lines lt 0 lc 'grey', \\  
    '0.40_200_200_01_01.fl' title 'FL' with points pt 2 ps 1 lc 'brown', \\  
    '0.40_200_200_01_01.ctr' title 'TB' with lines lt 1 lw 2 lc 'blue', \\  
    '0.40_200_200_01_01-000002.pp' title 'PP' with points pt 7 ps 1 lc 'red', \\  
    '0.40_200_200_01_01-000002.pp' notitle with lines lt 0 lw 2 lc 'red'  
exit gnuplot
```

Przykład skryptu wykorzystywanego do tworzenia  
Algorytmu Wyszukiwania Ścieżki w wersji 2D.

efekt 

# Gnuplot – źródła danych

- Nazwa
- 0.40\_200\_200\_01\_01.AB
  - 0.40\_200\_200\_01\_01.ctr
  - 0.40\_200\_200\_01\_01.dat
  - 0.40\_200\_200\_01\_01.fl
  - 0.40\_200\_200\_01\_01.fsp
  - 0.40\_200\_200\_01\_01.gc
  - 0.40\_200\_200\_01\_01.il
  - 0.40\_200\_200\_01\_01.isp
  - 0.40\_200\_200\_01\_01.ia
  - 0.40\_200\_200\_01\_01.plt
  - 0.40\_200\_200\_01\_01.rl
  - 0.40\_200\_200\_01\_01.tr
  - 0.40\_200\_200\_01\_01\_000002\_000009.eps
  - 0.40\_200\_200\_01\_01\_000002.pp





# Gnuplot – wygląd serii danych

Teraz widać dlaczego dobrze jest opisywać kolejne serie danych w oddzielnych wierszach.

10:

```
set terminal wxt
plot sin(x) title 'funkcja SIN' with lines lt 1 lw 1 dt 2 lc 'red', \
      cos(x) notitle with lines lt 1 lw 3 dt 3 lc '#0000FF'
pause mouse
gnuplot exit
```

**title** – tu: ustawić własną nazwę dla bieżącego źródła danych

**notitle** – wyłącza widoczność nazwy bieżącego źródła

**with** – określa sposób prezentacji danych

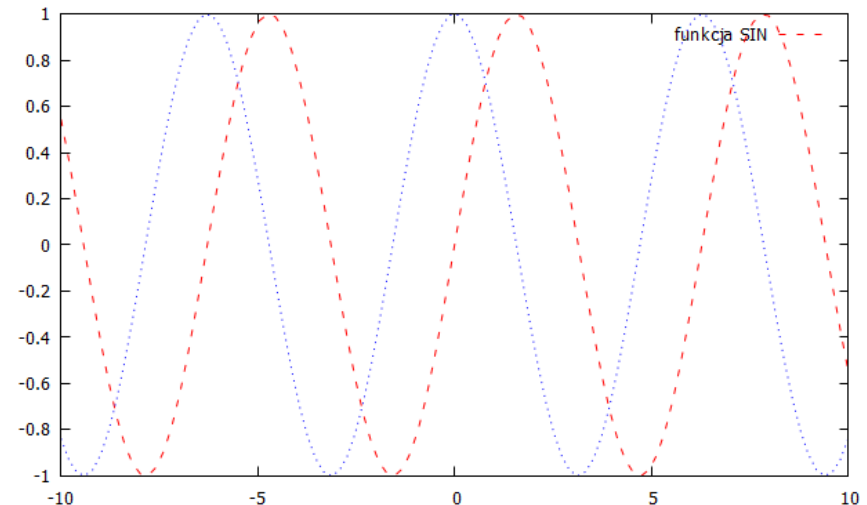
**lt** (od **linetype**) – określa rodzaj linii wykresu

**lw** (od **linewidth**) – określa grubość linii wykresu

**dt** (od **dashtype**) – określa styl linii wykresu

**lc** (od **linecolor**) – określa kolor linii wykresu

Możliwe opcje: lines, points, dots, impulses, steps, boxes, histograms, linespoints, filledcurves, errorbars.



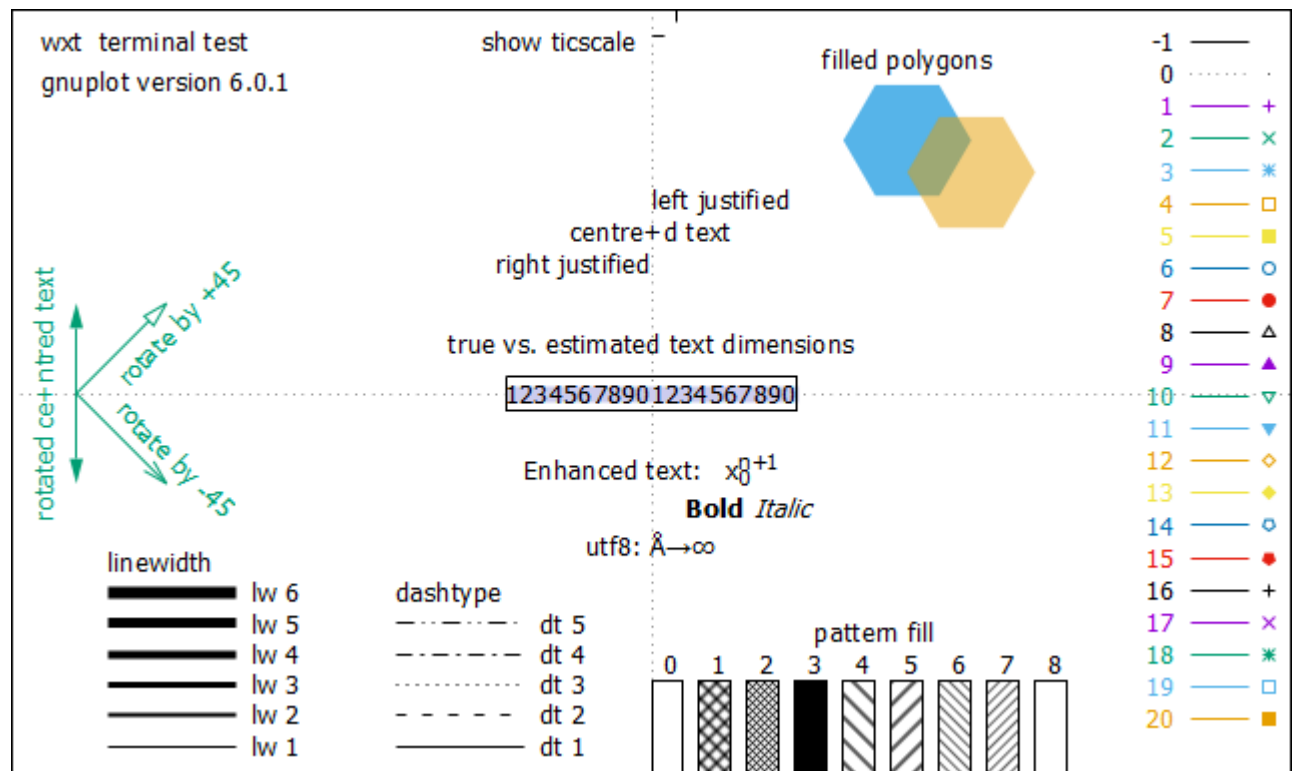
# Gnuplot – wygląd serii danych

11:

```
set terminal wxt
test
pause mouse
gnuplot exit
```

Skąd wiadomo jakie są grubości czy typy linii albo znaczniki punktów?

Czasami taki sam efekt można uzyskać na różne sposoby – przykładowo kolory można definiować na trzy sposoby: poprzez numer, nazwę albo kod RGB.



# Gnuplot – wygląd serii danych

12:

```
set terminal wxt
plot sin(x) title 'funkcja SIN' with points lt 1 lw 2 pt 6 ps 3 lc 7,\
      cos(x) notitle with impulses lt 1 lw 1 lc 14
pause mouse
gnuplot exit
```

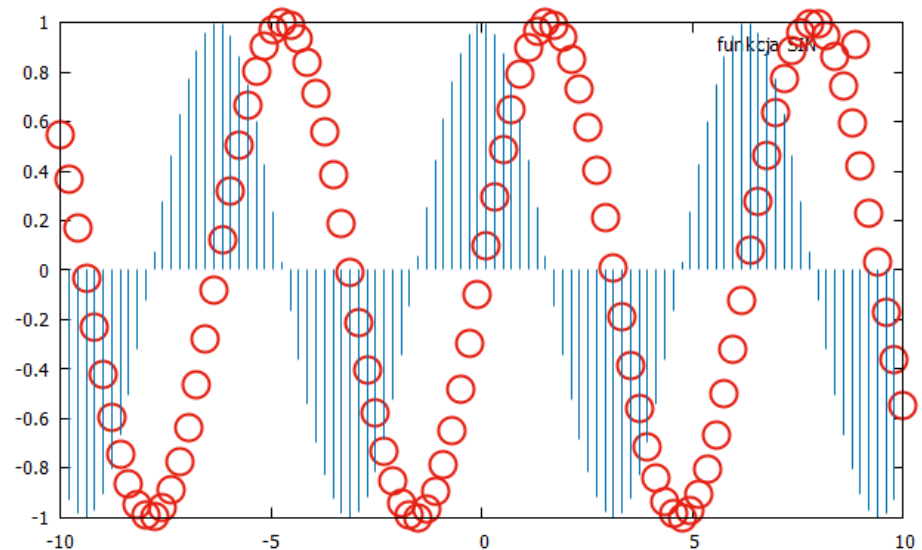
**impulses** – wyświetlanie danych w postaci pionowych linii

**pt** (od **pointtype**) – kształt znacznika punktu

**ps** (od **pointsize**) – rozmiar znacznika punktu

UWAGI:

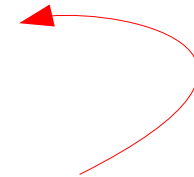
- w pierwszym wykresie usunięto opcję **dt** (znaczniki muszą być rysowane linią ciągłą)
- kolory zdefiniowano numerami (patrz **test** na poprzednim slajdzie)



# Gnuplot – konfiguracja osi wykresu

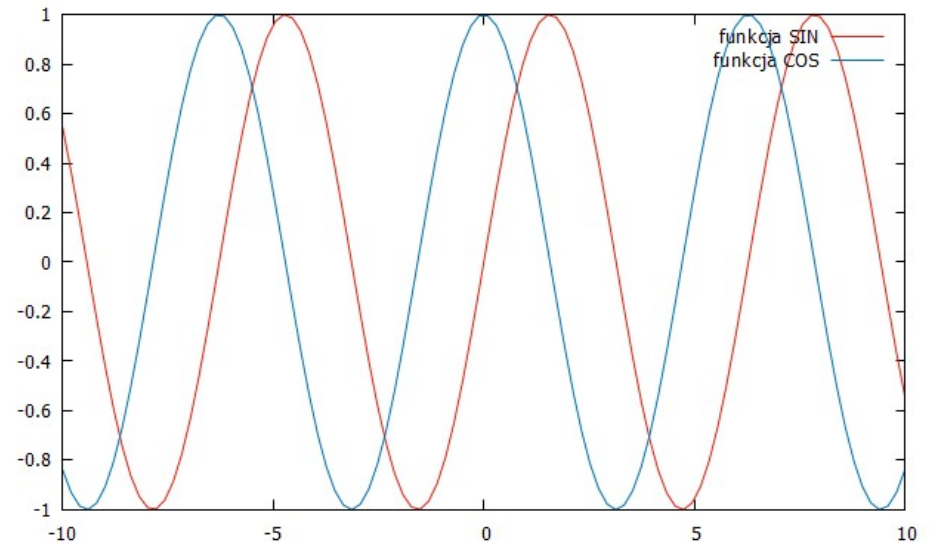
13:

```
set terminal wxt
plot sin(x) title 'funkcja SIN' with lines lt 1 lc 7, \
      cos(x) title 'funkcja COS' with lines lt 1 lc 14
pause mouse
gnuplot exit
```



Skrypt wyjściowy do dalszych ćwiczeń  
(niepotrzebne wpisy zostały usunięte).

UWAGA: szczegóły  
konfigurowania poszczególnych  
elementów rysunkowych mogą  
zależać od wersji Gnuplota,  
systemu operacyjnego oraz  
zastosowanego terminala.



# Gnuplot – konfiguracja osi wykresu

14:

```
set encoding utf8
set terminal wxt enhanced font 'Arial, 14'
set title 'Mój pierwszy porządy wykres'
set xlabel 't [s]'
set ylabel 'R [Ω]'
plot sin(x) title 'R_1^a = f(t)' with lines lt 1 lc 7, \
      cos(x) title 'R_2^a = g(t)' with lines lt 1 lc 14
pause mouse
gnuplot exit
```

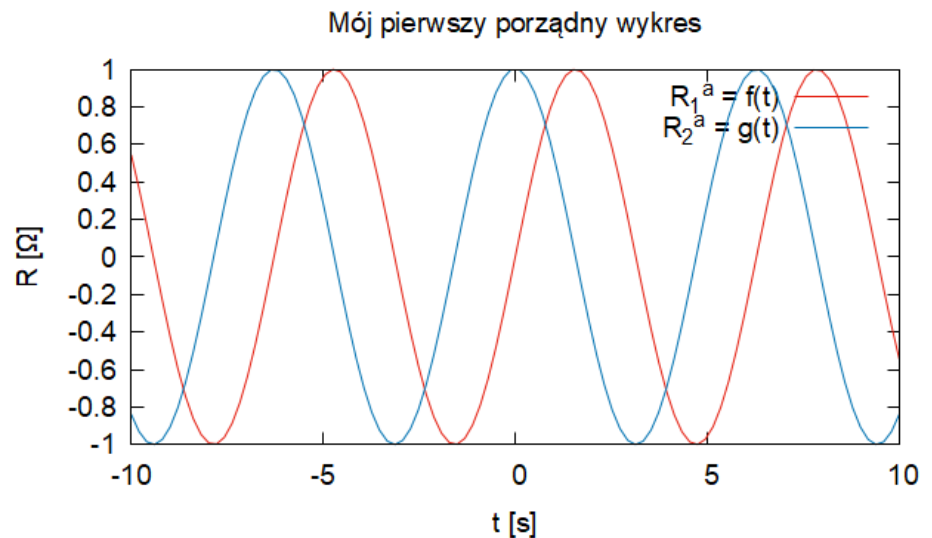
Do modyfikacji ustawień służą polecenia **set** oraz **unset**.

**encoding** – ustawia stronę kodową znaków

**enhanced font** – umożliwia stosowanie różnych czcionek, formatowanie tekstu oraz wstawianie znaków specjalnych

**title** – tu: wstawia tytuł wykresu

**xlabel, ylabel** – wstawia tytuł osi



# Gnuplot – konfiguracja osi wykresu

15:

```
...  
#set logscale y # W tym przykładzie źle wygląda  
set xrange [0:50]  
set xtics 5  
set yrange [-1.5:1.5]  
set ytics 0.5  
set grid  
...
```

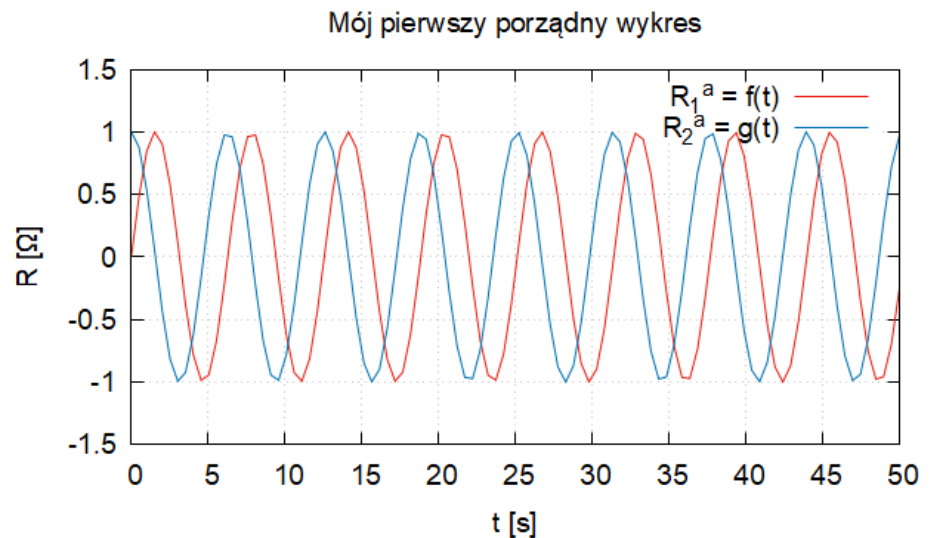
Kolejność wpisów nie jest ważna – istotne jest tylko, aby konfigurację wykonać przed użyciem komendy **plot**.

**xrange, yrange** – ustawia zakres osi

**xtics, ytics** – ustawia gęstość podziałki

**grid** – nakłada na wykres siatkę zgodnie z przyjętymi gęstościami podziałek osi

**logscale** – ustawia skalę logarytmiczną



# Gnuplot – konfiguracja legendy

16:

```
...  
set terminal wxt enhanced font 'Arial, 14'  
...  
set key font ",10"  
set key horizontal  
set key bottom center  
...
```

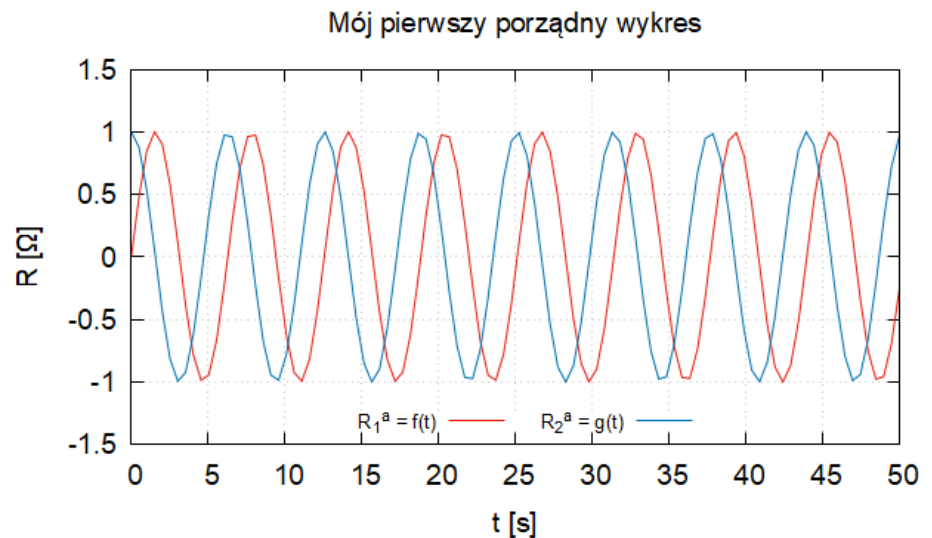
Do tej pory rozmiar czcionki wynosił 14 punktów.

**key font** – zmienia rodzaj i rozmiar czcionki (tu rodzaj czcionki pozostaje bez zmian)

**key horizontal** – ustawia pozycje legendy poziomo (nie ma komendy **key vertical**)

**key bottom center** – pierwszy wpis ustawia legendę na dole (bottom), w środku (center) lub u góry wykresu (top)

**key bottom center** – drugi wpis ustawia legendę po lewej (left), pośrodku (center) lub po prawej stronie wykresu (right)



# Gnuplot – konfiguracja legendy

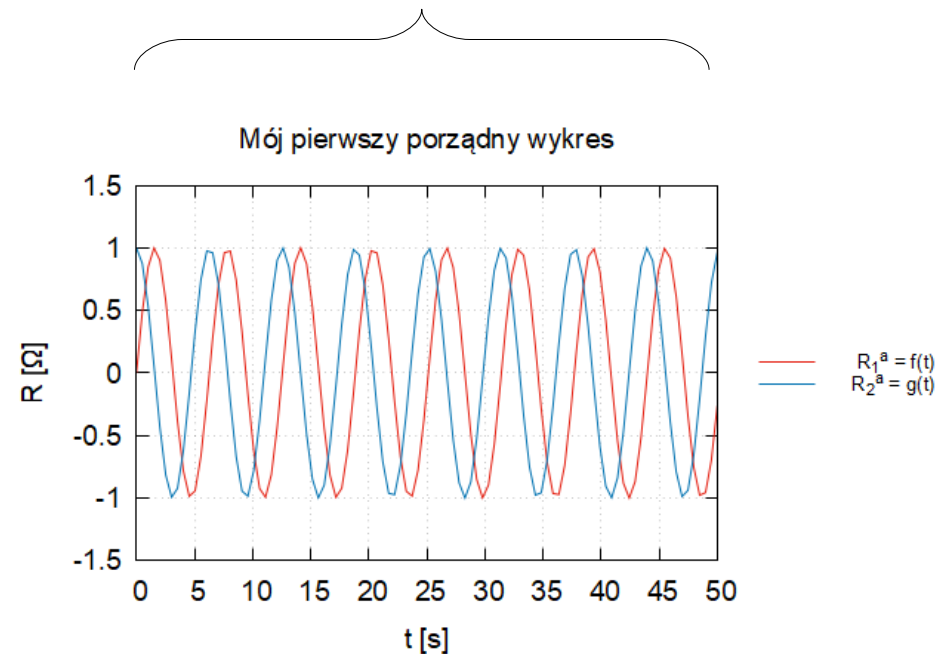
17:

```
...  
set key right center outside reverse  
...
```

**outside** – wstawia legendę poza oknem wykresu w miejscu wskazanym przez wpisy poprzedzające (tu po prawej na środku)

**reverse** – odwraca kolejność elementów w legendzie: najpierw są linie lub znaczniki, a potem nazwy (domyślnie jest odwrotnie)

Konsekwencją użycia komendy **outside** jest zmniejszenie się rozmiaru okna wykresu.





# Gnuplot – konfiguracja legendy

18:

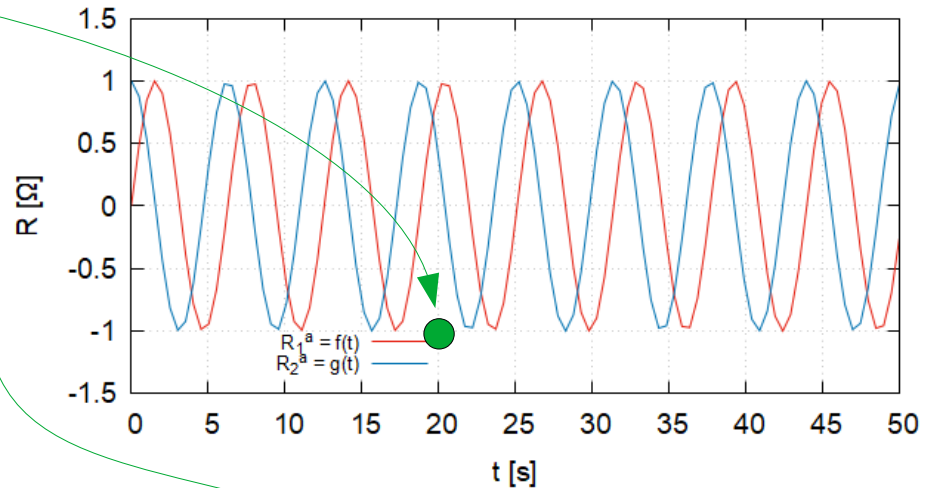
```
...  
set key at 20, -1  
set key at screen 0.6, screen 0.8  
...
```

**key at** – ustawia legendę tak, aby prawy górny narożnik legendy znalazł się w wyznaczonym miejscu – współrzędne odnoszą się do liczb widocznych na osiach układu współrzędnych

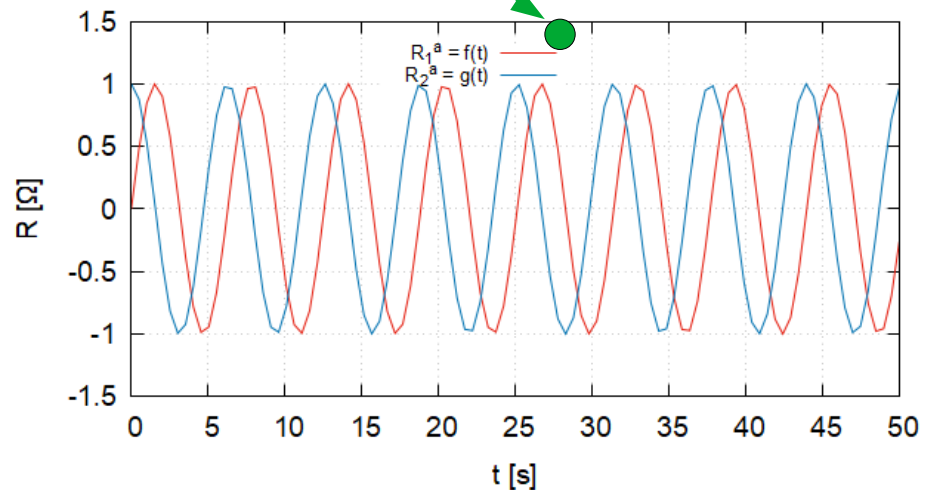
**key at screen** – ustawia legendę tak, aby prawy górny narożnik legendy znalazł się w wyznaczonym miejscu – współrzędne odnoszą się do całkowitego rozmiaru okna (nie samej ramki, w której rysowany jest wykres)

Dodając znak komentarza # na początku odpowiedniego wiersza skryptu, można obejrzeć działanie obu wariantów.

Mój pierwszy porządy wykres



Mój pierwszy porządy wykres



# Gnuplot – wstawianie tekstów i równań

19:

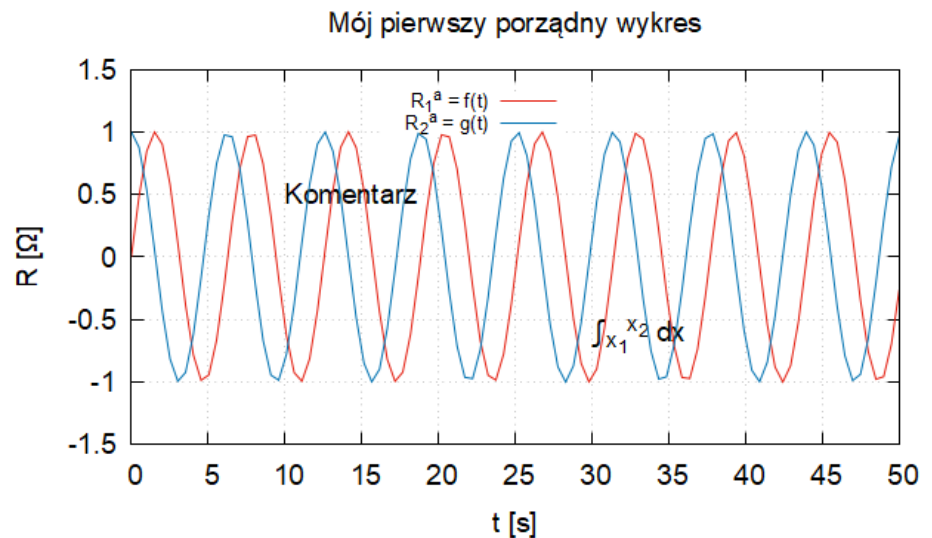
```
...  
set label "Komentarz" at 10, 0.5  
set label "{/Symbol \362}_{x_1}^{\{x_2\} {dx}}" at graph 0.6,0.3  
...
```

**at** – ustawia element tak, aby element znalazł się w wyznaczonym miejscu – współrzędne odnoszą się do liczb widocznych na osiach układu współrzędnych

**at graph** – ustawia element tak, aby element znalazł się w wyznaczonym miejscu – współrzędne odnoszą się do ułamka długości danej osi

Przy wstawianiu etykiety współrzędne nie odnoszą się do konkretnego narożnika pola, w którym ta etykieta jest wstawiana.

Skąd wiadomo, jaki to symbol?



# Gnuplot – wstawianie tekstów i równań

Numery znaków  
czcionki Symbol.

	!	∇	#	∃	%	&	ə	(	)	*	+	,	-	.	/
40	41	42	43	44	45	46	47	50	51	52	53	54	55	56	57
0	1	2	3	4	5	6	7	8	9	:	;	,	=	>	?
60	61	62	63	64	65	66	67	70	71	72	73	74	75	76	77
≡	A	B	X	Δ	E	Φ	Γ	H	I	∅	K	Λ	M	N	O
100	101	102	103	104	105	106	107	110	111	112	113	114	115	116	117
Π	Θ	P	Σ	T	Υ	ζ	Ω	Ξ	Ψ	Z	[	∴	]	⊥	_
120	121	122	123	124	125	126	127	130	131	132	133	134	135	136	137
-	α	β	χ	δ	ε	φ	γ	η	ι	φ	κ	λ	μ	ν	ο
140	141	142	143	144	145	146	147	150	151	152	153	154	155	156	157
π	θ	ρ	σ	τ	υ	ϖ	ω	ξ	ψ	ζ	{		}	~	
160	161	162	163	164	165	166	167	170	171	172	173	174	175	176	177
	Υ	'	≤	/	∞	f	♣	♠	♥	♣	↔	←	↑	→	↓
240	241	242	243	244	245	246	247	250	251	252	253	254	255	256	257
°	±	"	≥	×	∞	∂	•	÷	≠	≡	≈	...		—	┘
260	261	262	263	264	265	266	267	270	271	272	273	274	275	276	277
ℵ	ℑ	℔	℘	⊗	⊕	∅	∩	∪	⊃	⊇	⊄	⊂	⊆	∈	∉
300	301	302	303	304	305	306	307	310	311	312	313	314	315	316	317
∠	∇	®	©	™	Π	√	.	¬	∧	∨	⇔	⇐	⇑	⇒	↑
320	321	322	323	324	325	326	327	330	331	332	333	334	335	336	337
∠	∠	®	©	™	Σ	∫		∫	∫		∫	∫	{		
340	341	342	343	344	345	346	347	350	351	352	353	354	355	356	357
	∫	∫	∫		∫	∫		∫	∫		∫	∫		∫	∫
360	361	362	363	364	365	366	367	370	371	372	373	374	375	376	377

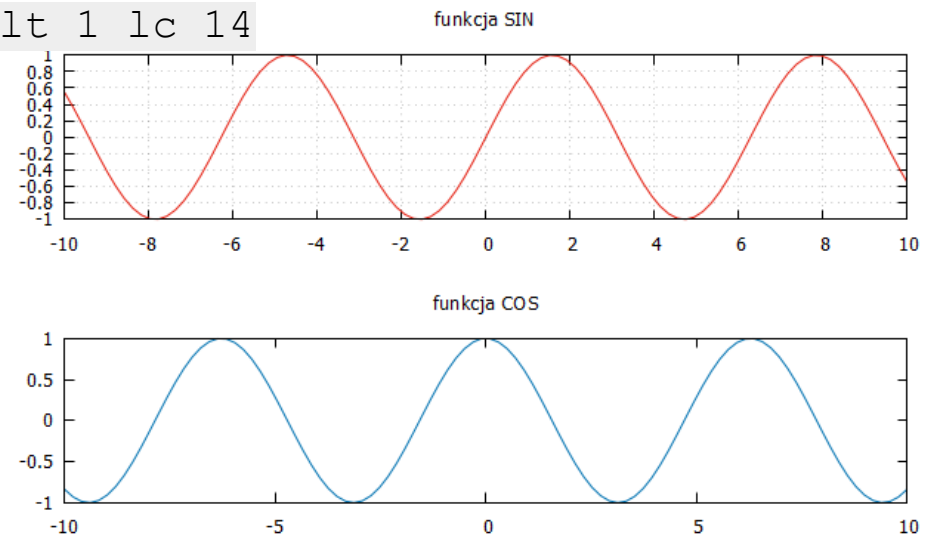
# Gnuplot – komenda multiplot

20:

```
set terminal wxt
set multiplot layout 2,1
# pierwszy wykres (górny):
  set title 'funkcja SIN'
  set xtics 2
  set grid
  plot sin(x) notitle with lines lt 1 lc 7
# drugi wykres (dolny):
  set title 'funkcja COS'
  set xtics 5
  set ytics 0.5
  unset grid
  plot cos(x) notitle with lines lt 1 lc 14
unset multiplot
pause mouse
gnuplot exit
```

**Metoda 1:** Podział okna na wiersze i kolumny – tu są 2 wiersze i 1 kolumna.

Oba okna wykresów można konfigurować niezależnie.



# Gnuplot – komenda multiplot

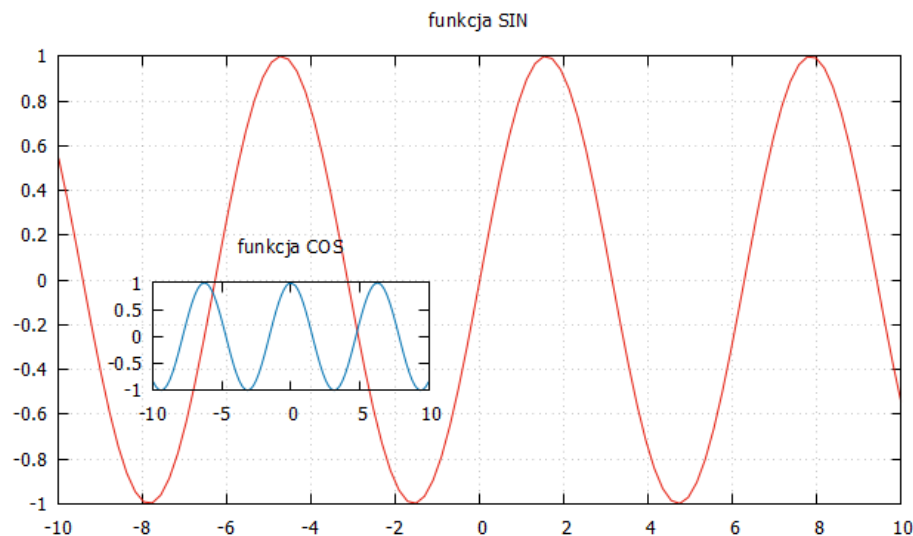
21:

```
...  
# pierwszy wykres (górny):  
  set size 1,1  
  ...  
# drugi wykres (dolny):  
  set size 0.4,0.4  
  set origin 0.1,0.4  
  ...  
...
```

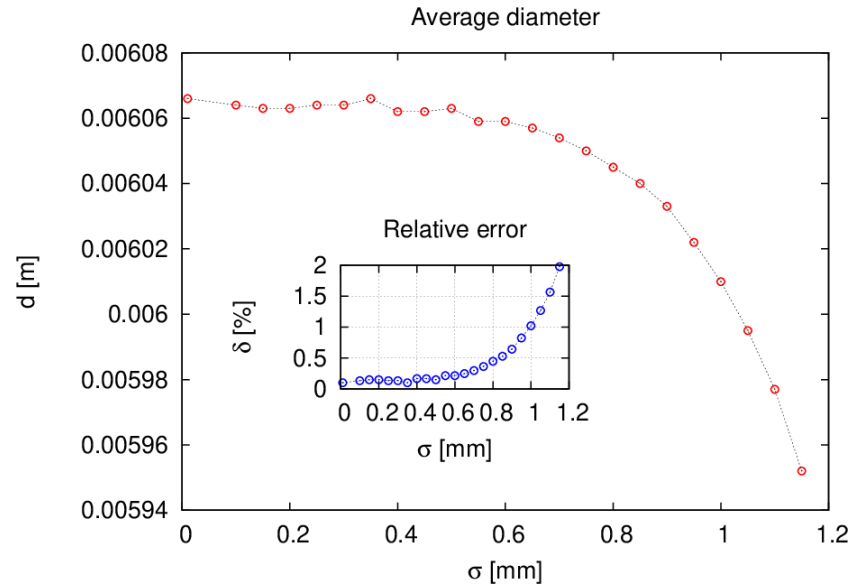
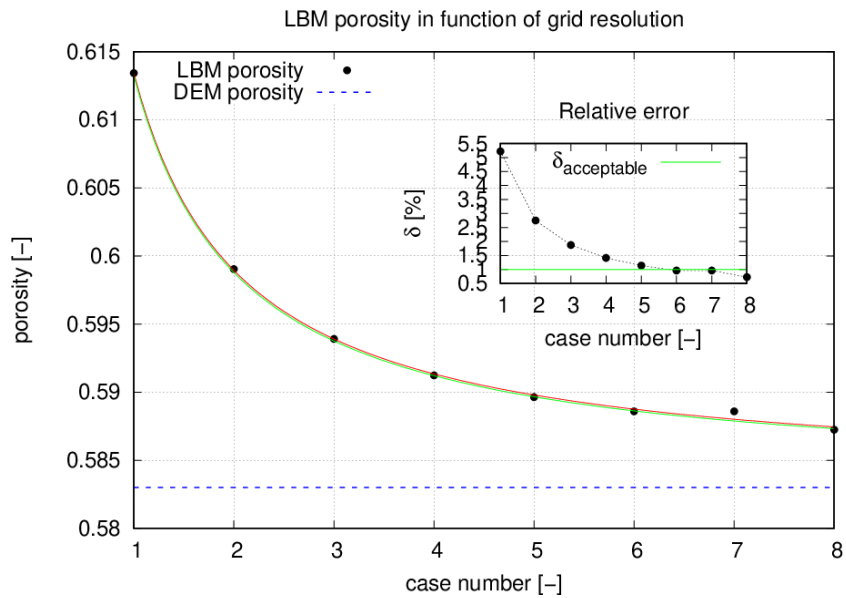
**set size** – ustawia rozmiar okna wykresu jako procent całkowitego obszaru rysunkowego, np. 0.4, 0.4 to 40% szerokości i 40% wysokości

**origin** – ustawia położenie okna wykresu względem lewego dolnego rogu obszaru rysunkowego, np. 0.1, 0.2 to 10% od lewej i 20% od dołu

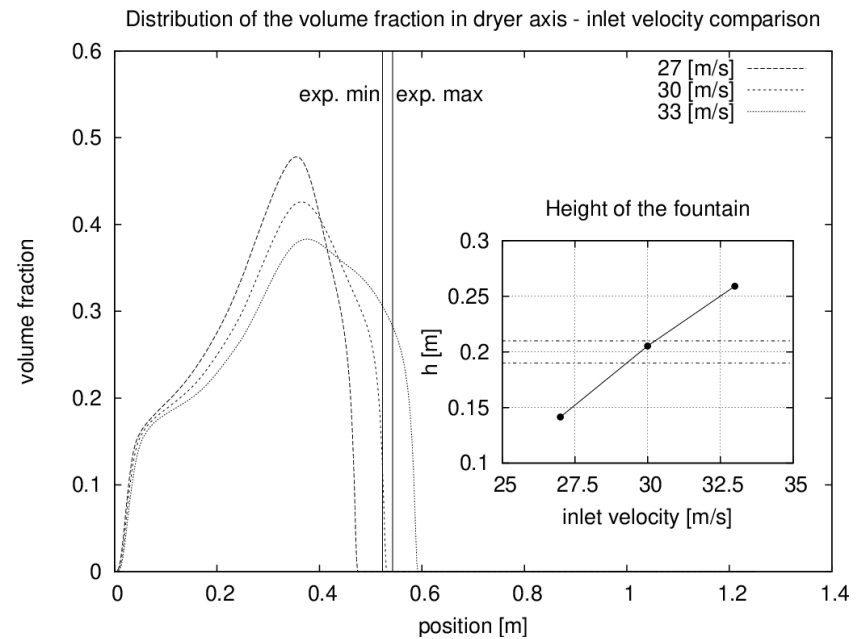
**Metoda 2:** Niezależne definiowanie rozmiarów i położenia okien.



# Gnuplot – komenda multiplot



Przykłady stosowania komendy multiplot.



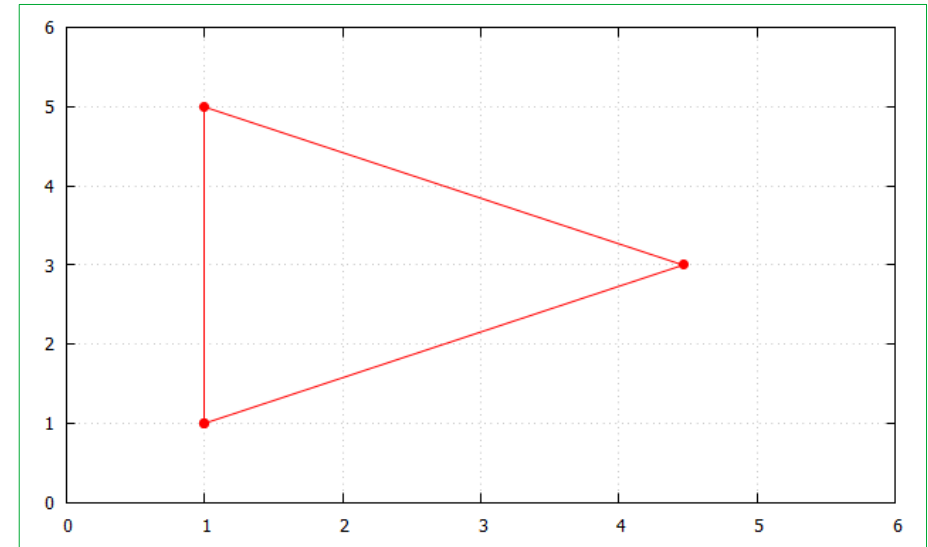
# Gnuplot – rozmiar i proporcje okna

22:

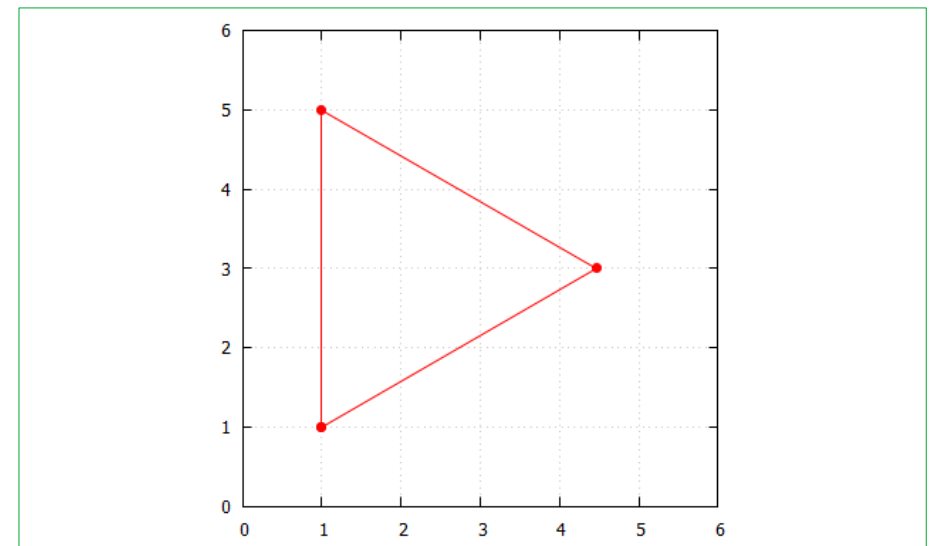
```
set terminal wxt size 1200, 800
set xrange [0:6]
set yrange [0:6]
set grid
plot '22.dat' notitle with \
linespoints lt 1 pt 7 lc 'red'
pause mouse
set size ratio 1
replot
pause mouse
exit gnuplot
```

**size** (w terminalu) – ustawia rozmiar okna rysunkowego (aby je zobaczyć, na rysunkach obok granice okien zaznaczono kolorem zielonym)

**size ratio 1** – zachowuje proporcje osi X i Y na 1:1; jednostki na obu osiach mają tę samą długość



Na tym rysunku nie widać, że trójkąt jest równoboczny!



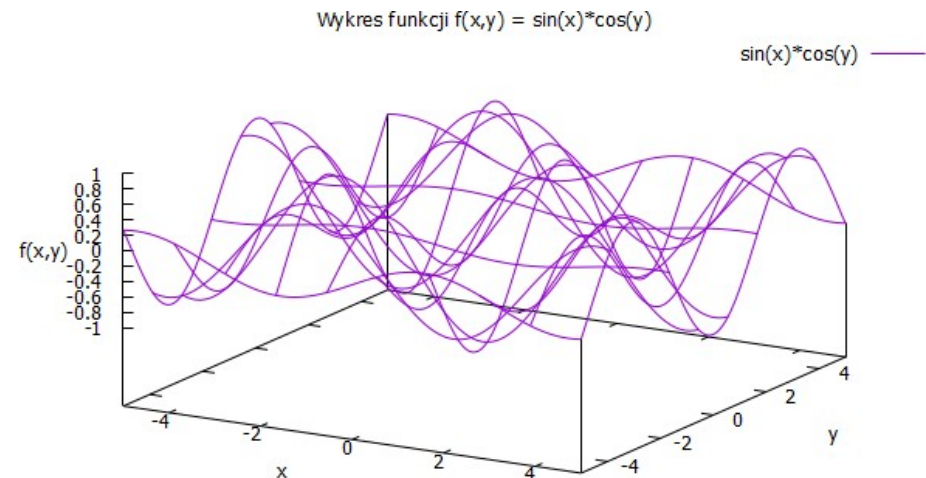
# Gnuplot – funkcje dwóch zmiennych

23:

```
set terminal wxt enhanced
set title 'Wykres funkcji  $f(x,y) = \sin(x)*\cos(y)$ '
set xlabel 'x'
set ylabel 'y'
set zlabel 'f(x,y)'
set xrange [-5:5]
set yrange [-5:5]
plot sin(x)*cos(y) with lines
pause mouse
exit gnuplot
```

Ogólnie opcja **enhanced** umożliwia stosowanie różnych rozszerzeń.

**splot** – komenda służąca do rysowania wykresów funkcji dwóch zmiennych. Domyślnie powierzchnie rysowane są w postaci siatki





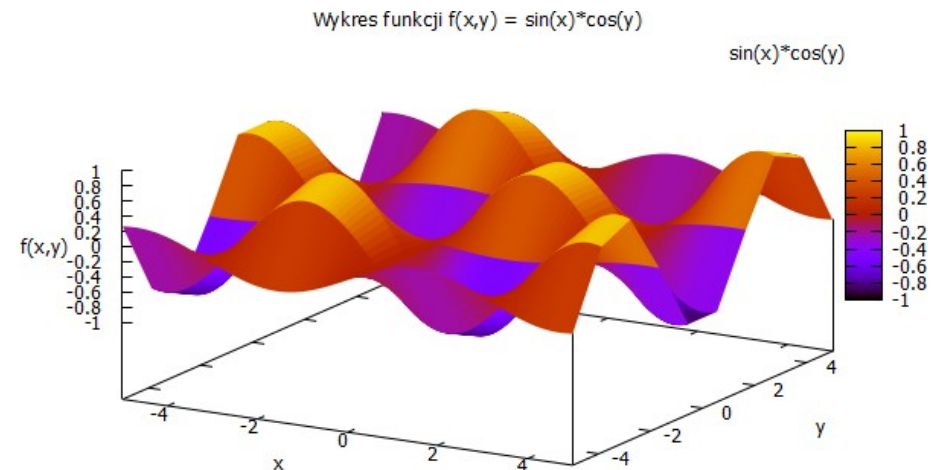
# Gnuplot – funkcje dwóch zmiennych

24:

```
set terminal wxt enhanced
set title 'Wykres funkcji  $f(x,y) = \sin(x)*\cos(y)$ '
set xlabel 'x'
set ylabel 'y'
set zlabel 'f(x,y)'
set xrange [-5:5]
set yrange [-5:5]
set view 60, 30, 1, 1 #tu wartosci domyslne
splot sin(x)*cos(y) with pm3d
pause mouse
exit gnuplot
```

**pm3d** – komenda służąca do rysowania powierzchni ciągłych i kolorowania ich wg wartości funkcji

**view** – komenda umożliwiająca zmianę punktu widzenia obserwatora; kolejne liczby oznaczają, kąt obrotu wokół osi x, liczby oznaczają, kąt obrotu wokół osi z, skalę osi x, skalę osi z



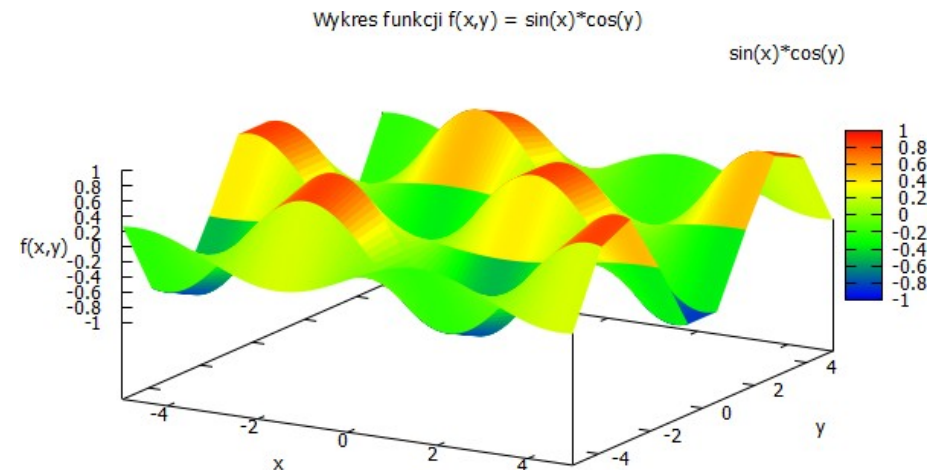
# Gnuplot – funkcje dwóch zmiennych

25:

```
set terminal wxt enhanced
set title 'Wykres funkcji  $f(x,y) = \sin(x)*\cos(y)$ '
set xlabel 'x'
set ylabel 'y'
set zlabel 'f(x,y)'
set xrange [-5:5]
set yrange [-5:5]
set palette defined ( 0 'blue', 1 'green', 2 'yellow', 3 'red' )
#set palette rgb 33,13,10
splot sin(x)*cos(y) with pm3d
pause mouse
exit gnuplot
```

**palette** – komenda służąca do definiowania własnych palet kolorów – tu paleta składa się z trzech kolorów, które przechodzą płynnie z jednego w drugi, wg zadanej kolejności. Zamiast nazw kolorów można używać kodów RGB, np.:

```
set palette rgb 33,13,10
```



# Gnuplot – funkcje dwóch zmiennych

26:

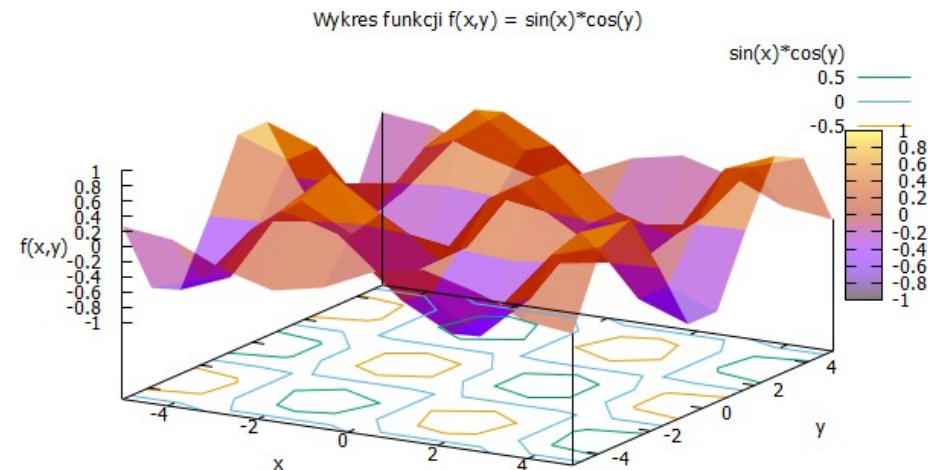
```
...  
set pm3d #at b # at t  
set hidden3d  
set contour base  
set style fill transparent solid 0.5  
unset surface  
splot sin(x)*cos(y) with lines  
...
```

**hidden3d** – powoduje, że linie i krawędzie w trybie 3d nie są widoczne

**contour base** – rzutuje izoliny stałych wartości na dolną płaszczyznę układu współrzędnych

**style fill transparent solid** – definiuje wypełnienie powierzchni jako półprzezroczyste

**unset surface** – wyłącza widoczność powierzchni w postaci siatki



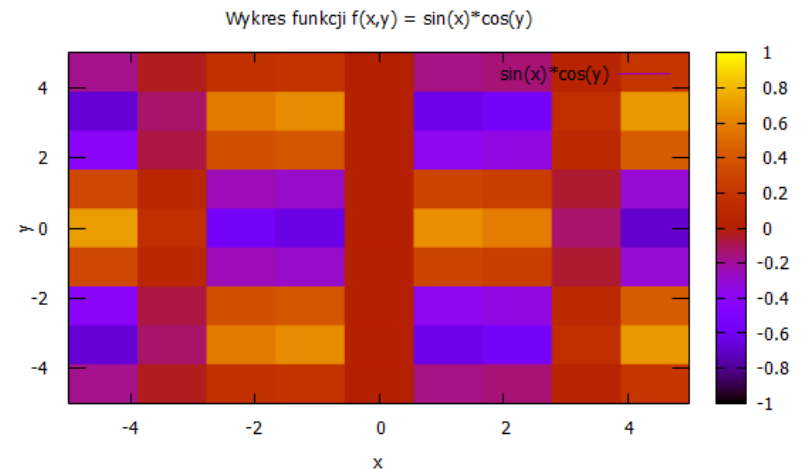
# Gnuplot – funkcje dwóch zmiennych

27:

```
...  
set view map  
set pm3d  
set hidden3d  
splot sin(x)*cos(y) with lines  
...
```

Pozostałe zasady konfigurowania rysunków są w większości takie same jak przy instrukcji **plot**.

**view map** – ustawia tryb rysowania na płaską mapę

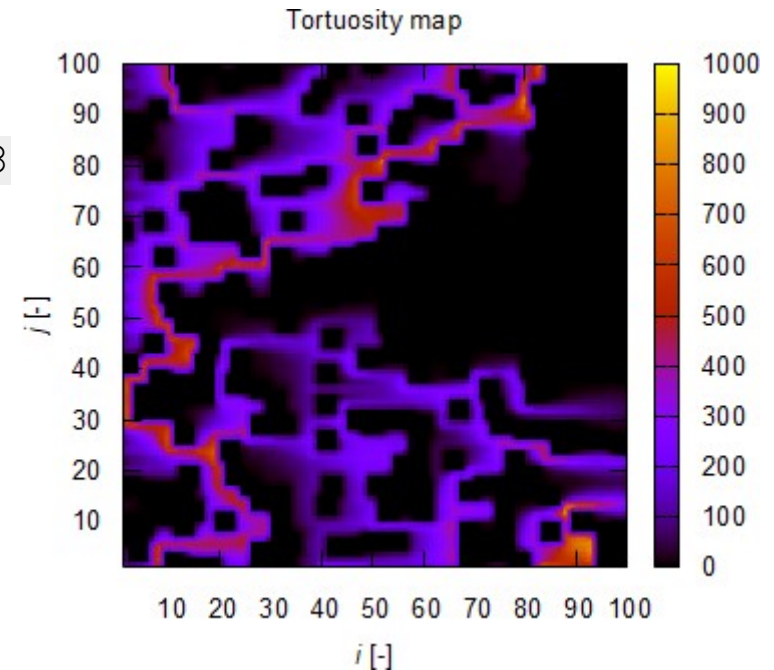


# Gnuplot – funkcje dwóch zmiennych

28:

```
set terminal wxt enhanced font 'Arial' 18
set size ratio 1
set title 'Tortuosity map'
set xlabel '{/Arial-Italic i} [-]'
set ylabel '{/Arial-Italic j} [-]'
set xrange [1: 100]
set yrange [1: 100]
set pm3d
set nokey
set hidden3d
set contour base
set view map
splot '28.sum' u 1:2:4 notitle with lines lt 1 lc palette
pause mouse
exit gnuplot
```

Przykład skryptu, w którym źródłem danych jest plik – tu wykorzystuje się 1, 2 i 4 kolumnę danych, jako odpowiednio: x, y oraz  $f(x,y)$ .



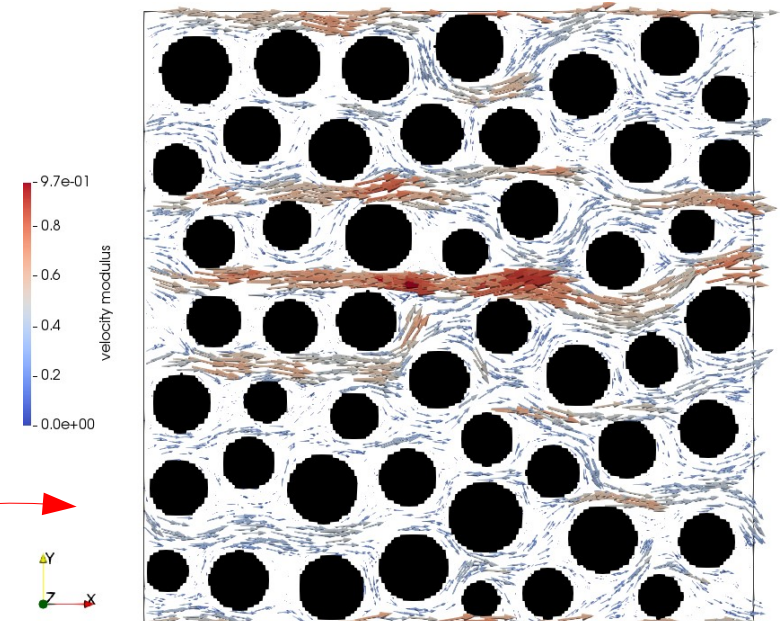
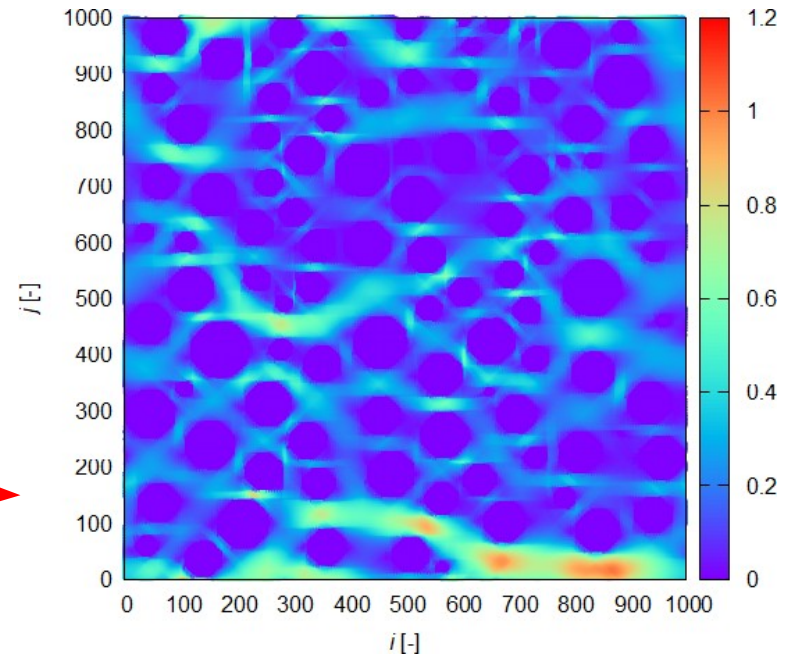
1	1	1	1	112
2	1	2	1	100
3	1	3	1	100
4	1	4	1	100
5	1	5	1	0
6	1	6	1	0

length: 31 Ln: 1 Col: 1 Pos: 1 Unix (LF) UTF-8 INS

# Gnuplot – funkcje dwóch zmiennych

**Z4:**

```
set encoding utf8
set size ratio 1
set xlabel '{/Arial-Italic i} [-]'
set ylabel '{/Arial-Italic j} [-]'
set terminal pngcairo enhanced font 'Arial,12'
set output 'Rys_11_10.png'
unset zrange
set zrange [0:]
set xrange [:1000]
set yrange [:1000]
set view map
set pm3d
set hidden3d
set palette rgbformulae 33,13,10
plot '0.50_100_100_01_01-vec.dat' \
  u 1:2:3:4:(10*sqrt($3**2+$4**2)) \
  notitle with vectors lc palette z
exit gnuplot
```



Gnuplot może być użyty do wizualizacji pól skalarnych lub wektorowych w przestrzeni 2D, ale zadanie to można zrealizować znacznie lepiej stosując np. pliki VTK i program ParaView.



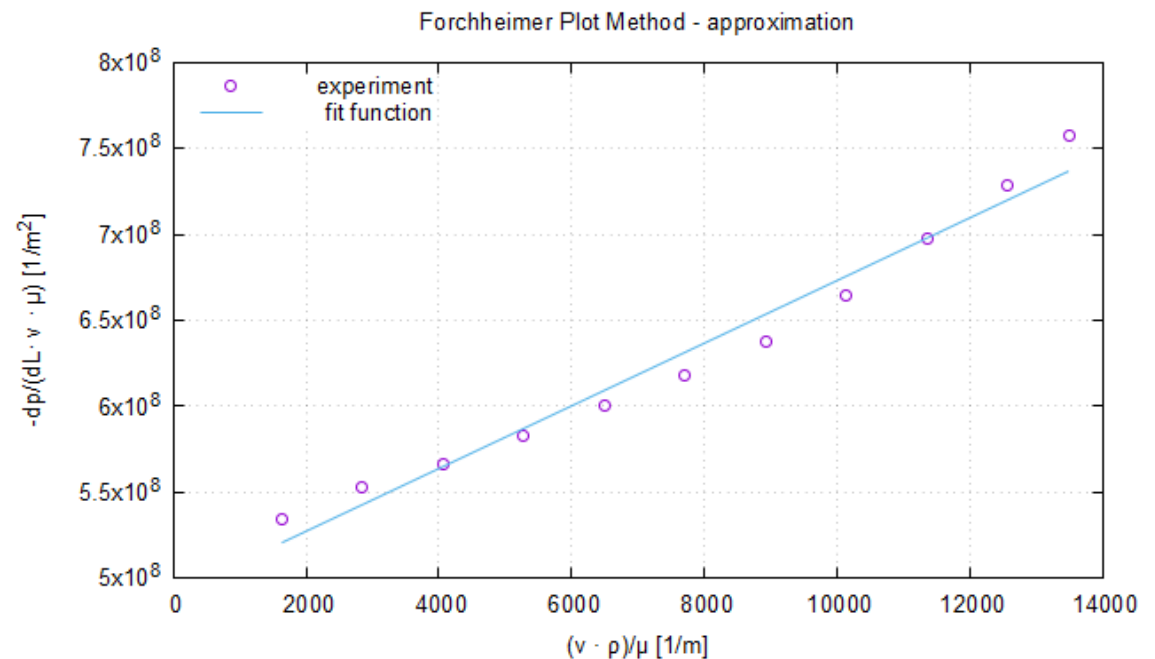
# Gnuplot – dopasowanie funkcji do danych

29:

```
set terminal wxt enhanced font 'Arial' 14
...
f(x) = a*x+b
fit f(x) '29.exp' via a,b
plot '29.exp' title 'experiment' with points lt 1 pt 6 ps 1, \
      f(x) title 'fit function' with lines lt 3
pause mouse
set print '29.txt' # Otwiera plik
print "a =", a
print "b =", b
set print # Zamyka plik
exit gnuplot
```

Gnuplot do dopasowywania funkcji do danych wykorzystuje Metodę Najmniejszych Kwadratów.

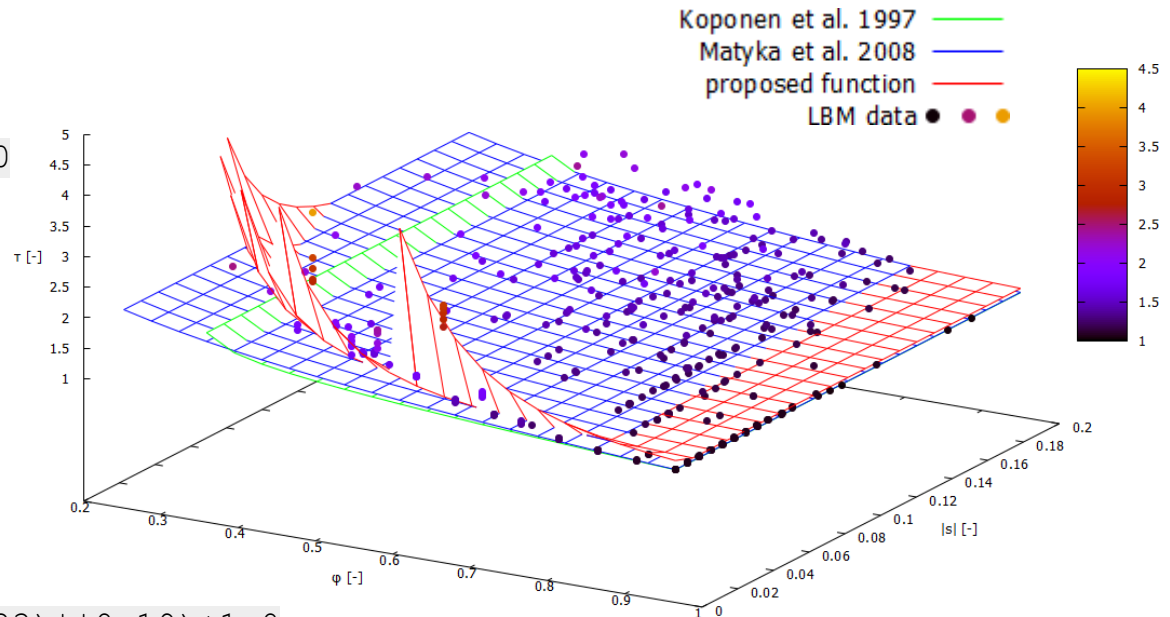
Tu dodatkowo współczynniki funkcji są zapisane do pliku.



# Gnuplot – dopasowanie funkcji do danych

**Z5 (długo się uruchamia):**

```
set terminal wxt size 1200, 800
set xlabel '{/Symbol f} [-]'
set ylabel '|s| [-]'
set zrange [1:5]
set samples 21
set isosample 21
set pm3d explicit
set pm3d interpolate 10,10
set hidden3d
```



```
f97(x,y) = 0.65*(1.0-x)/((x-0.33)**0.19)+1.0
```

```
fmatp(x,y) = 1 - 0.77*log(x)
```

```
h(x,y) = a/((x**b)*(y**c)) + f/((x**d)*(y**e))
```

```
fit h(x,y) '150_150_tau_sum.dat' u 2:3:5 via a,b,c,d,e,f
```

```
set title 'Data comparison for grid 150({\267}4)x150({\267}4) (all repetitions)'
```

```
set zlabel '{/Symbol t} [-]'
```

```
splot f97(x,y) title 'Koponen et al. 1997' with lines lt 1 lc 'green', \
```

```
fmatp(x,y) title 'Matyka et al. 2008' with lines lt 1 lc 'blue', \
```

```
h(x,y) title 'proposed function' with lines lt 1 lc 'red', \
```

```
'150_150_tau_sum.dat' u 2:3:5 title 'LBM data' with points palette pt 7 lt 1
```

```
pause mouse
```

```
exit gnuplot
```

$$\tau(\phi, |s|) = \frac{a}{\phi^b \cdot |s|^c} + \frac{d}{\phi^e \cdot |s|^f}$$

Przykład dopasowania danych pochodzących z serii 15 000 symulacji, do funkcji z 6 stałymi + porównanie z formułami z literatury.



# Gnuplot – integracja z językami programowania

30:

```
set terminal png enhanced font 'Arial' 12
set output '30_00000006.png'
set title 'Wykres y = f(x,t) dla t = 0.500000'
set xlabel 'os x [-]'
set ylabel 'os y [-]'
set grid
plot '30.dat' notitle with lines lt 1
exit gnuplot
```

W każdej iteracji nazwa pliku musi być inna!

Iteracja numer	1	30_00000001.png
Iteracja numer	2	30_00000002.png
Iteracja numer	3	30_00000003.png
Iteracja numer	4	30_00000004.png
Iteracja numer	5	30_00000005.png

```
!tworzenie skryptu PLT:
open(1, file='30.plt')
write(1,*) 'set terminal png enhanced font ''Arial'' 12'
write(tmp, fmt='(I8.8)') iter
write(1,*) 'set output ''30_//trim(tmp)//.png'''
write(tmp, fmt='(F12.6)') t
write(1,*) 'set title ''Wykres y = f(x,t) dla t = '//trim(tmp)//''''
write(1,*) 'set xlabel ''os x [-]'''
write(1,*) 'set ylabel ''os y [-]'''
write(1,*) 'set grid'
write(1,*) 'plot ''30.dat'' notitle with lines lt 1'
write(1,*) 'exit gnuplot'
close(1)
```

```
!uruchomienie skryptu - Windows:
call system('30.plt')
!uruchomienie skryptu - Linux:
!call system('gnuplot 30.plt')
```

DO i = 1, n:

Stwórz plik PLT  
Oblicz nowe dane  
Zapisz nowe dane  
Uruchom plik PLT

Fragment kodu (tu FORTRAN) generującego skrypt PLT, który jest następnie uruchamiany z poziomu programu.

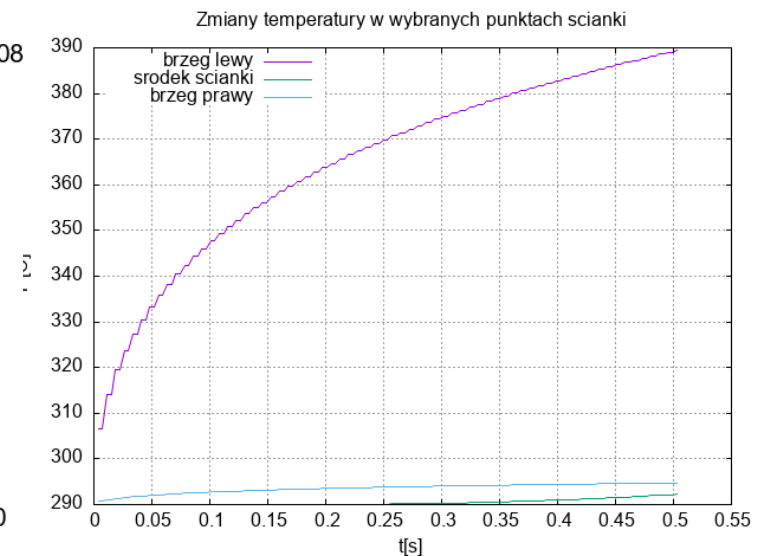
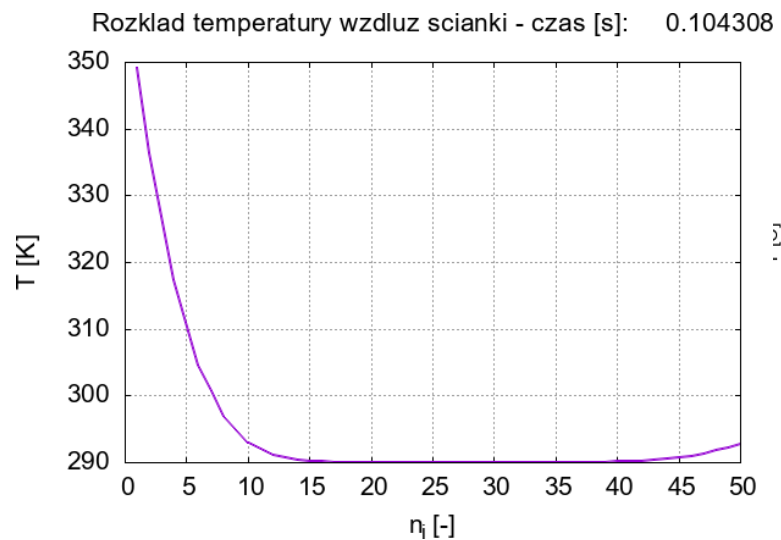
# Gnuplot – integracja z językami programowania

Z6:

```
subroutine make_script( numer, t)
implicit none
integer(kind=4)      :: numer
real(kind=4)        :: t
character(len=12)    :: tmp
open(1, file='fourier.plt')
write(1, '(A)') 'set terminal png enhanced font 'Arial' 16'
write(unit=tmp, fmt='(I12.12)') numer
write(1, '(A)') 'set output 'wyniki\\'//trim(tmp)//'.png'''
write(unit=tmp, fmt='(F12.6)') t
write(1, '(A)') 'set title 'Rozklad temperatury wzdluz scianki - czas [s]: '//trim(tmp)//''
write(1, '(A)') 'set xlabel 'n_i [-]'''
write(1, '(A)') 'set ylabel 'T [K]'''
write(1, '(A)') 'set grid'
write(1, '(A)') 'plot 'fourier.txt' notitle with lines lt 1 lw 2'
write(1, '(A)') 'exit gnuplot'
close(1)
end subroutine
```

Fragment kodu (tu FORTRAN) programu służącego do obliczania profilu temperatury w jednorodnej ścianie płaskiej.

- Nazwa
- wyniki
  - fourier.avi
  - fourier.dat
  - fourier.exe
  - fourier.f90
  - fourier.inc
  - fourier.o
  - fourier.p
  - fourier.plt
  - fourier.txt
  - fourier-p.plt



# Gnuplot – integracja z językami programowania

31:

```
import subprocess
```

```
# Komenda do uruchomienia Gnuplota:
```

```
gnuplot_command = """
```

```
set terminal pngcairo
```

```
set output '31.png'
```

```
plot sin(x)
```

```
"""
```

```
# Uruchomienie Gnuplota:
```

```
process = subprocess.Popen(['gnuplot'], stdin=subprocess.PIPE, text=True)
```

```
process.communicate(input=gnuplot_command)
```

Jak uruchomić przykład (xUbuntu 22.04):

- nadaj plikowi **31.py** uprawnienia do uruchamiania

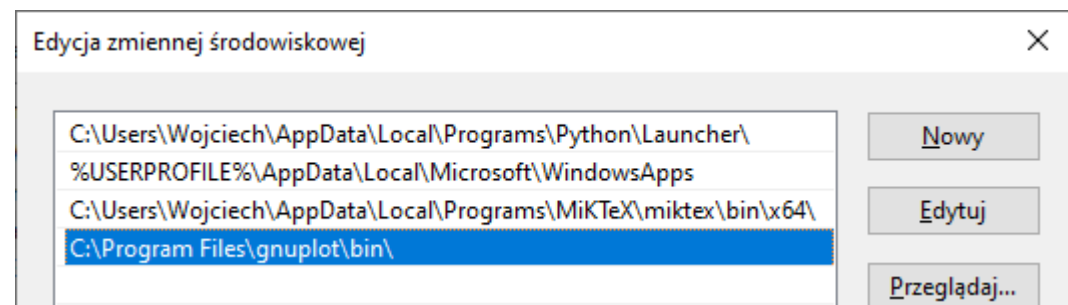
Uruchamianie:  Zezwolenie na uruchamianie jako program

- w terminalu wydaj polecenie:

```
$ python3 31.py
```

Jak uruchomić przykład (Windows 10):

- zainstaluj środowisko Python
- dodaj położenie programu Gnuplot do zmiennej środowiskowej PATH
- uruchom skrypt **31.py**



Przykład skryptu Pythona, w którym wywoływane jest środowisko Gnuplot.

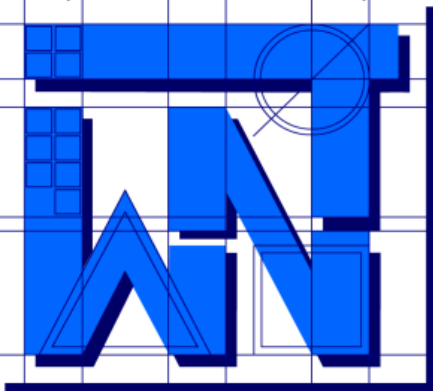
# Gnuplot – podsumowanie

---

- Program z obszaru Wolnego Oprogramowania
- Brak kosztów, częste aktualizacje
- Brak ograniczeń odnośnie zastosowań
- Wieloplatformowość: Windows, Unix/Linux, Mac OS, ...
- Obsługa różnych formatów wyjściowych
- Elastyczność i szerokie możliwości konfiguracyjne
- Rozbudowany język skryptowy
- Integracja z wieloma językami programowania (np. Python, C, Fortran)
- Duża społeczność użytkowników i wsparcie
- Możliwość tworzenia skomplikowanych wykresów 2D i 3D
- Możliwość dopasowywania funkcji do danych
- Znajomość innego języka programowania nie jest konieczna

Istnieją biblioteki graficzne o zbliżonym przeznaczeniu, które mają inne/większe możliwości niż Gnuplot, ale wymagają znajomości jakiegoś języka programowania – przeważnie Pythona: DISLIN, Matplotlib, Plotly, Bokeh, Altair, ...

Wydział Nauk Technicznych



UNIVERSITY OF WARMIA AND MAZURY IN OLSZTYN  
The Faculty of Technical Sciences  
POLAND, 10-957 Olsztyn, M. Oczapowskiego 11  
tel.: (48)(89) 5-23-32-40, fax: (48)(89) 5-23-32-55  
URL: <http://www.uwm.edu.pl/edu/sobieski/> (in Polish)



---

**Dziękuję za uwagę**

**Wojciech Sobieski**

---

VIII SZKOŁA INŻYNIERII SYSTEMÓW BIOTECHNICZNYCH  
Guzowy Piec, 11-14 września 2024 r.