gnuplot - wprowadzenie

Katarzyna Grzelak

listopad 2018

Wprowadzenie

Programy do opracowywania danych doświadczalnych (rysowanie funkcji, punktów z błędami, dopasowywanie zależności funkcyjnych do danych . . .):

- `gnuplot`
  - darmowy, dostępny dla systemów operacyjnych Linux (zwykle w standardowej dystrybucji !), Windows, MAC OS
  - polecenia wpisywane jak w terminalu lub zapisywane w pliku-skrypcie (!) $\Rightarrow$ można łatwo zautomatyzować procedurę wykonywania rysunków
  - niewyszukana grafika
  - w OKWF'ie uruchamiany z terminala poleceniem `gnuplot`
  - strona www `http://www.gnuplot.info/`

Wprowadzenie

- `Origin`
  - komercyjny program, licencja (Windows) Wydziału Fizyki dla studentów WF, licencja MISMaP dla studentów MISMaP
- `SciDAVis`
  - darmowy program, wersje na Linux,Windows,MacOS
  - w OKWF'ie uruchamiany z terminala poleceniem `scidavis`
  - strona www `http://scidavis.sourceforge.net/`
- ...

Polecenia programu gnuplot

Polecenia mogą być wprowadzane interakcyjnie albo zapisywane w pliku-skrypcie.

- `gnuplot`
  (polecenia wpisywane interakcyjnie)
- `gnuplot skrypt.gp`
  (okno graficzne zaraz znika po wykonaniu)
- `gnuplot -persist skrypt.gp`
  (okno graficzne zostaje *na zawsze*)
- `gnuplot skrypt.gp`
  (gdy wewnątrz skryptu na końcu jest linia `pause -1` skrypt czeka na naciśnięcie `ENTER`)
- Rozróżniane są wielkie i małe litery
- Wszystkie nazwy poleceń można skrócić (tak długo jak skrót polecenia jest jednoznaczny)
- Opcje poleceń muszą być podawane w określonej kolejności

Zapisywanie napisów i liczb

- Napisy zapisywane są w pojedynczych lub podwójnych apostrofach: `'dane.dat'`
- Liczby całkowite: 1, 15
- Liczby rzeczywiste: 1.,15.,1E0,1.5E1,5E-1
- Uwaga na wynik dzielenia liczb całkowitych, np. 1/2 = 0
- Zdefiniowana stała `pi`

Funkcje, operatory

- Zdefiniowane funkcje:
  `abs(x), sin(x), cos(x), tan(x), exp(x),sqrt(x)`
  `...`
- Własne definicje funkcji, np.: $a(x-b)^2$
  `f(x)=a*(x-b)**2`
- Operatory: a**b (potęgowanie), a*b (mnożenie), a/b (dzielenie, a-b, a+b ...

Rysowanie funkcji

- `plot sin(x)`
  `set samples 10`
  `plot sin(x)` lub `replot`
- Program oblicza wartość funkcji w tylu punktach ile ustawimy poleceniem `set samples`
- Powrót do domyślnych ustawień: `reset`
- `a=5;b=6`
  `h(x)=a*x+b`
  `plot h(x)`
- `g(x)=cos(a*x)`
  `plot a=0.2,g(x),a=0.4, g(x)`

Rysowanie funkcji

- skala logarytmiczna:

```
plot exp(-x)
set logscale y
replot
```

- dwa rysunki na jednym:

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

Rysowanie funkcji

- ```
  plot sin(x) with points
  plot sin(x) with boxes
  plot sin(x) with lines
  plot sin(x) with impulses
  plot [-pi:pi] sin(x)
  ```
  ```

Rysowanie funkcji, postać parametryczna

Przykład. Krzywa zdefiniowana w następujący sposób:

$$\begin{cases} x = 5\cos t \\ y = 2\sin t \end{cases}$$

```
set parametric
set xrange [-6:6]
set yrange [-6:6]
set trange [0:10]
set isosamples 60
plot 5*cos(t),2*sin(t)
```

Rysowanie funkcji w 3D

- ```
  splot sin(x*y)
  set hidden3d
  set isosamples 30,30
  replot
  set xrange[-3:3]
  set yrange[-3:3]
  set zrange[-2:5]
  ```
- ```
  set parametric
  splot 2*u,u**2+v,v**2
  ```

Dane z błędami i bez

- `plot 'dane.dat'`
- `plot 'dane.dat' with errorbars`
- `plot 'dane.dat' with xyerrorbars`
- `plot 'dane.dat' using 1:2:3 w ye`
- `plot 'dane.dat' using 2:1:3 w xe`
- Rysowanie: na osi `x` suma pierwszej i czwartej kolumny, na osi `y` wartości z piątej kolumny:
  `plot 'dane1.dat' u ($1+$4):5 with lines`

Opcje dla jednego rysunku

- Tytuł: `set title 'Beams'`
- Opis dolnej osi: `set xlabel 'Energy [GeV]'`
- Opis lewej osi: `set ylabel 'p1[GeV]'`
- Zmiana zakresu osi x: `plot [-5:10] sin(x)`
- Zmiana zakresu osi x i y: `plot [-5:10] [-1.5:1.5] sin(x)`

Dwa wykresy na jednym rysunku

- Przy nakładaniu na siebie dwóch rysunków o różnych skalach są cztery możliwe zestawy osi: `x1y1` (lewa i dolna), `x1y2` (prawa i dolna), `x2y2` (prawa i górna), `x2y1` (lewa i górna)
- Np. użycie dolnej i prawej osi: `axes x1y2`
- Przykład: `plot 'dane1.dat' u ($1+$4):5 with lines, 'dane2.dat' u ($1+$4):($15-$17) w l axes x1y2`
- Opis górnej osi: `set x2label 'Energy [GeV]'`
- Opis prawej osi: `set y2label 'p1[GeV]'`
- Kreski na prawej osi: `set y2tics`
- `set ytics nomirror`
- `set tics out`

Legenda

- Zmiana legendy: `plot sin(x) title 'sinus'`
- Włączanie: `set key`
- Wyłączanie: `unset key` lub `set nokey`

Zapisywanie do pliku . . .

. . . jest możliwe w wielu formatach: pdf,png,jpg,ps,eps,latex . . . Np. dla formatu pdf:

- `set terminal pdf`
- `set output 'sin.pdf'`
- `plot sin(x)` lub `replot`
- `set output`
- `set terminal x11` na Linux'ie lub `set term win` na Windows

Zapisywanie do pliku . . .

. . . jest możliwe w wielu formatach: pdf,png,jpg,ps,eps . . . Np. dla formatu eps:

- `set terminal postscript eps color`
- `set output 'sin.eps'`
- `plot sin(x)` lub `replot`
- `set output`
- `set terminal x11`
- Także `set terminal latex` i inne
- W Linux'ie w terminalu `epstopdf sin.eps`

Przekazywanie argumentów do wnętrza skryptu

Niezależnie od wersji programu gnuplot:

- `gnuplot -e "polecenie1; polecenie2; ..." skrypttest.gp`
- Przykład 1:
  `gnuplot -e " filename='dane.txt' " skryptdane.gp`
- Wnętrze skryptu:

```
plot filename w e
pause -1
```

- Przykład 2: `gnuplot -e "n=1;m=2;s=2" skrypt.gp`
- Wnętrze skryptu:

```
f(x)=n*exp(-(x-m)**2/s**2)
plot f(x)
pause -1
```

Przekazywanie argumentów do wnętrza skryptu

Do wnętrza skryptu można przesłać argumenty wywołania podobnie jak w bash Uwaga - zależność składni od wersji programu gnuplot!

- Wywołanie:
  `gnuplot -e "call 'skrypt2.gp' 1 5 7"`
  - Wersja ≤ 4:
    Wnętrze skryptu:
    ```
    n=$0; m=$1; s=$2
    f(x)=n*exp(-(x-m)**2/s**2)
    plot f(x)
    ```
  - Wersja 5:
    Wnętrze skryptu:
    ```
    n=ARG1; m=ARG2; s=ARG3
    f(x)=n*exp(-(x-m)**2/s**2)
    plot f(x)
    ```

Dopasowywania zależności funkcyjnych do danych

Metoda najmniejszych kwadratów:
Poszukiwanie wartości parametrów $a_1, \ldots, a_l$, które minimalizują wyrażenie:

$$X^2 = \sum_{i=1}^{n} \frac{[y_i - f(x; a_1, \ldots, a_l)]^2}{\sigma_i^2}$$

f - dopasowywana funkcja
y_i - wynik pomiaru
σ_i - błąd pomiaru y_i

Jakość dopasowania: $\chi^2 / liczba\_stopni\_swobody$

Dopasowywanie krzywych

Dopasowywanie zależności funkcyjnych do danych doświadczalnych.
Kolejne etapy:

- Definicja funkcji
  np. `ft(x)=a*x**2+b*x+c`
- Definiowanie początkowych wartości parametrów (tak aby funkcja możliwie najlepiej opisywała dane)
  np. `a=15; b=2.5; c=1.5`
- Sprawdzenie jak dobrze te parametry zostały wybrane
  np. `plot "dane.dat" w e, ft(x)`
- Dopasowanie (uwaga - przy opcji using muszą byc zdefiniowane dokładnie (!) 3 kolumny, trzecia kolumna to błedy y)
  np. `fit ft(x) "dane.dat" using 1:2:3 via a,b,c`
- Sprawdzenie jakości dopasowania:
  - Wizualne
    np. `plot "dane.dat" w e, ft(x)`
  - Sprawdzenie czy χ^2/liczba_stopni_swobody jest bliskie 1

Ustawienia

- Zapisywanie błędów parametrów dopasowania do zmiennej:
  `set fit errorvariables`
  Przykład na wypisywanie wartości parametrów i ich błędów na ekran:

```
set fit errorvariables
n=1;m=2;s=4
f(x)=n*exp(-(x-m)**2/s**2)
plot f(x)
fit f(x) 'funkcja1.dat' u 1:2:3 via n,m,s
print "n= ", n ," n_err= ", n_err
pause -1
```

- Wyniki dopasowania zapisywane do pliku o nazwie fit.log:
  FIT_LOG='fit.log'
- Przykłady zmian warunków na zakończenie poszukiwania minimum wyliczanej funkcji:
  - FIT_LIMIT = 1e-6
  - FIT_MAXITER = 50