Notatki 5

Gnuplot

Oczywiście istnieją bardziej nowoczesne programy do rysowania wykresów niż Gnuplot. Jest on jednak niezwykle użytecznym narzędziem, które umożliwia bardzo szybką wizualizację danych, a nawet ich analizę oraz dopasowywanie do nich funkcji matematycznych. Co więcej, Gnuplot bywa używany przez inne programy jako silnik do tworzenia wykresów – taka opcja istnieje chociażby w LaTeX-u (wykresy w `TikZ`). Dlatego też warto poznać przynajmniej jego podstawy i mieć świadomość istnienia bardziej zaawansowanych funkcji.

Gnuplot może działać w dwóch trybach. Po pierwsze, gdy wywołamy go poleceniem `gnuplot`, uruchomi się w trybie interaktywnym – w którym możemy wpisywać kolejne polecenia i uzyskiwać ich efekt. Po drugie, te same polecenia możemy zapisać w pliku i podać go jako argument (np. `gnuplot plik` lub `gnuplot --persist plik`). Zostaną one wówczas wykonane jak skrypt, od razu. Jest to bardzo przydatne jeśli chcielibyśmy zautomatyzować proces tworzenia wykresu (np. tworząc grafikę do publikacji/prezentacji) i nie wpisywać wszystkich komend ponownie gdy np. zmieniły się tylko dane wejściowe do zwizualizowania.

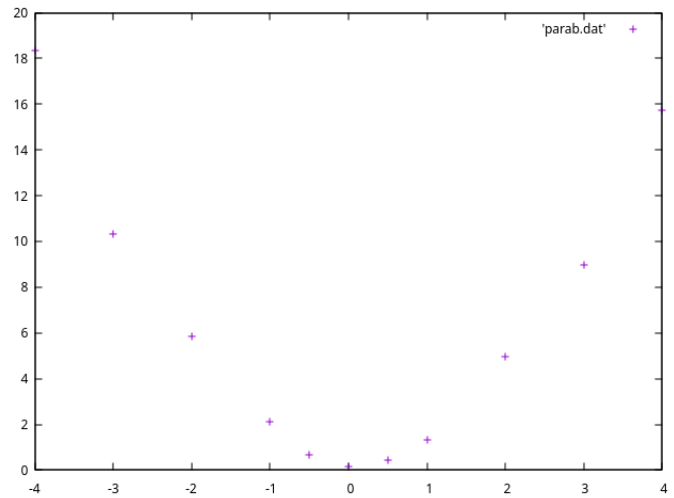
Na stronie umieściłem prosty plik `parab.dat`, który zawiera punkty z grubsza leżące na paraboli.

```
-4   18.34  1.02
-3   10.32  0.67
-2   5.86   0.34
-1   2.12   0.22
-0.5 0.67   0.11
 0   0.17   0.04
 0.5 0.43   0.14
 1   1.34   0.53
 2   4.95   0.89
 3   8.99   1.03
 4   15.75  3.92
```

Pierwsza kolumna zawiera wartości x, druga y, trzecia oszacowania błędów „pomiaru”. Spróbujmy swoich sił.

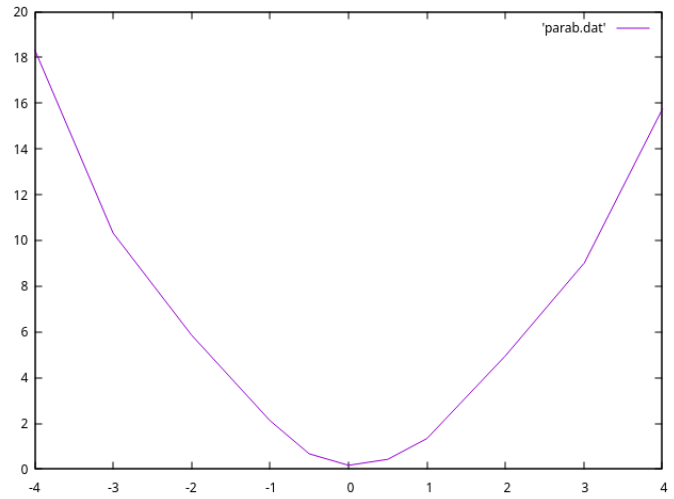
```
$ gnuplot

 G N U P L O T
 Version 5.4 patchlevel 4    last modified 2022-07-10

 Copyright (C) 1986-1993, 1998, 2004, 2007-2022
 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 'parab.dat'
```

Powinniśmy otrzymać (w nowym oknie) wykres zawartych w pliku punktów.



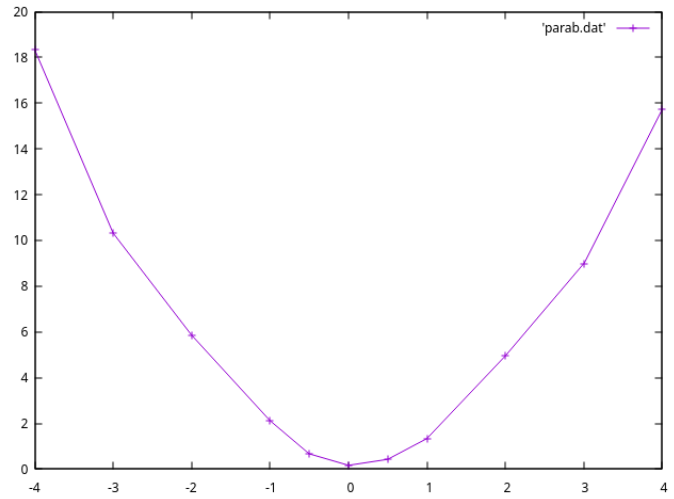
Możemy nim manipulować, np. przesuwać strzałkami na klawiaturze, przybliżać/oddalać kółkiem myszy, powiększać wybrany obszar zaznaczając prawym przyciskiem myszy. Domyślnie dane dyskretne rysowane są w postaci oddzielnych punktów. Możemy połączyć je łatwo liniami:

```
gnuplot> plot 'parab.dat' with lines
```



Czasami wygodne jest też wyrysowanie zarówno punktów, jak i linii:

```
gnuplot> plot 'parab.dat' with linespoints
```



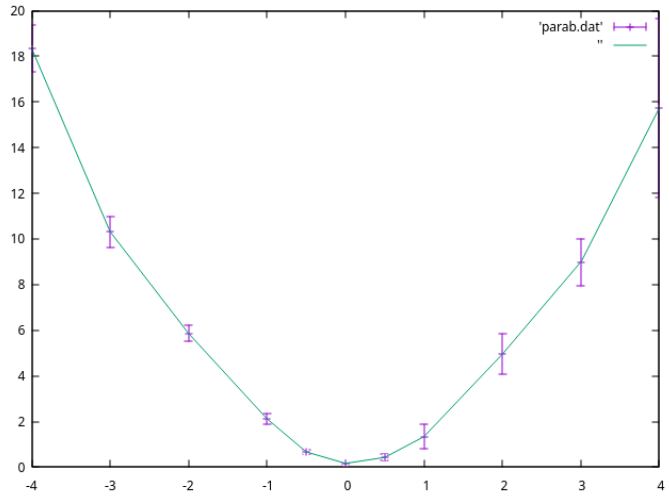
Większość poleceń w Gnuplocie można skracać – np. `w l` czy `w lp` zamiast dwóch pokazanych powyżej opcji.

Jak dotąd nie wykorzystywaliśmy trzeciej kolumny pliku, zawierającej oszacowania błędów. Możemy je narysować:

```
gnuplot> plot 'parab.dat' with errorbars
```

Same słupki błędów wyglądają tak sobie, więc możemy narysować drugi wykres – z tych samych punktów, ale połączonych linią.

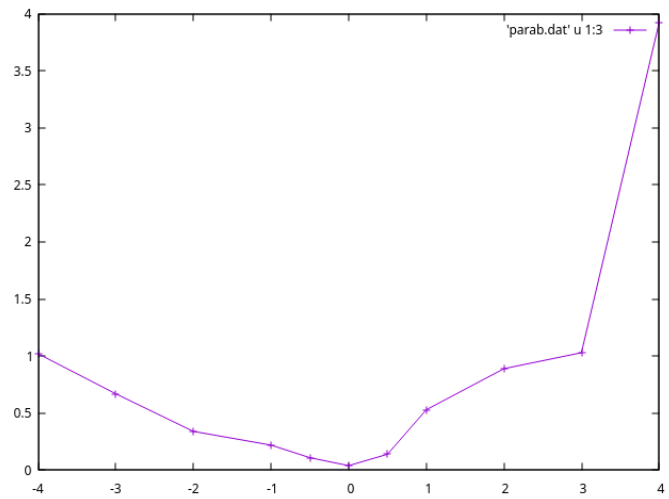
```
gnuplot> plot 'parab.dat' w e, '' w l
```



Jeśli rysujemy ten sam plik co poprzednio, mieliśmy prawo podać pustą nazwę (i z tego skorzystaliśmy). Gdybyśmy rysowali zawartość dwóch różnych plików na jednym wykresie, nazwa ta oczywiście musiałaby się tam znaleźć.

Dotychczas milcząco korzystaliśmy z konwencji, w której Gnuplot zakłada, że w pierwszej kolumnie są wartości x, w drugiej y, a w trzeciej ewentualne oszacowania błędu wartości y (jeśli takie są potrzebne). Oczywiście tak być wcale nie musi – wtedy podajemy jawnie numery kolumn po słowie kluczowym `using` (lub w skrócie `u`):

```
gnuplot> plot 'parab.dat' u 1:3 w lp
```



W ten sposób dostaliśmy wykres wielkości błędu y dla różnych x. Dane w kolumnach możemy też przetwarzać za pomocą funkcji matematycznych – np żeby narysować nie błąd, ale jego kwadrat, moglibyśmy napisać:

```
gnuplot> plot 'parab.dat' u 1:($3**2) w lp
```

Jak widać „tryb matematyczny” włączamy nawiasami, a następnie do konkretnych kolumn odwołujemy się znakiem dolara. Rozpoznawane jest całkiem sporo funkcji. Np. aby wyrysować logarytm dziesiętny wartości bezwzględnej, możemy napisać:

```
gnuplot> plot 'parab.dat' u 1:(log10(abs($2))) w lp
```

Akurat skala logarytmiczna jest na tyle częsta, że można to samo (z dokładnością do innego przedstawienia osi) uzyskać przez:

```
gnuplot> set logscale y
gnuplot> plot 'parab.dat' w lp
```

Podobnie oczywiście można włączyć skalę logarytmiczną na osi x lub obydwu. Uwaga! Teraz wszystkie kolejne wykresy będą rysowane w skali (pół)logarytmicznej. Aby to wyłączyć, podajemy:

```
gnuplot> unset logscale y
```

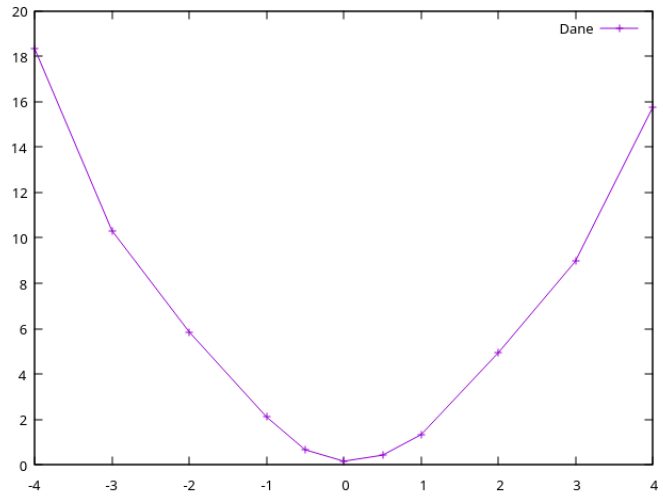
W tym miejscu warto podkreślić, że Gnuplot zapamiętuje także inne definicje i ustawienia jakie poczynimy w trakcie otwartej sesji. Czasami wygodniej więc wyłączyć go (`quit`) i otworzyć nową, czystą serię. Unaocznia to też dlaczego do powtarzalnych zadań wygodniej zapisywać nasze polecenia w pliku i wykonywać je wszystkie razem jako skrypt.

Możemy też używać w działaniach różnych kolumn jednocześnie – np. aby narysować błąd względny:

```
gnuplot> plot 'parab.dat' u 1:($2/$3) w lp
```

Domyślnie na legendzie każdy wykres nosi nazwę taką jaką podaliśmy w poleceniu `plot`. Możemy jednak nadawać własne nazwy – wystarczy to zdefiniować słowem kluczowym `title`:

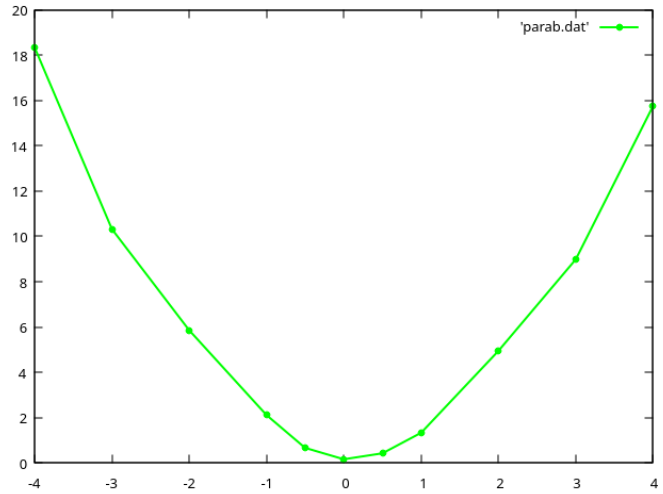
```
gnuplot> plot 'parab.dat' w lp title "Dane"
```



Podobnie możemy też zmieniać domyślny styl wykresu:

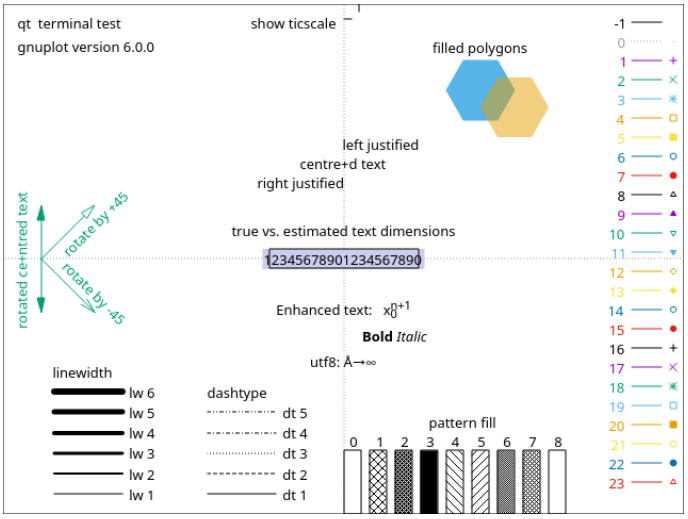
```
gnuplot> plot 'parab.dat' w lp linetype 7 linecolor 'green' linewidth 2
```

Powinno nam to dać zielony wykres z okrągłymi oznaczeniami punktów.



To obszerny temat, w który nie chcę się zagłębiać, ale warto wiedzieć, że możliwości dostosowywania są całkiem spore. Pewne pojęcie o możliwościach może nam dać polecenie `test`:

```
gnuplot> test
```

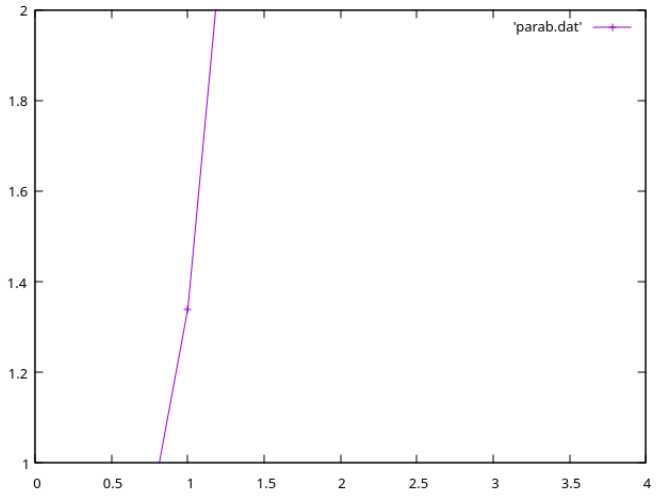


Jeśli chcemy zawęzić zakres, w jakim chcemy narysować wykres, możemy to zrobić dwojako. Po pierwsze, możemy zdefiniować go za pomocą odpowiedniego polecenia:

```
gnuplot> set xrange [0:4]
gnuplot> set yrange [1:2]
```

Od tego momentu wszystkie kolejne wykresy będą rysowane w podanym zakresie x i y. Możemy też podawać je dla każdego wykresu osobno – np. to samo co wyżej uzyskamy przez:

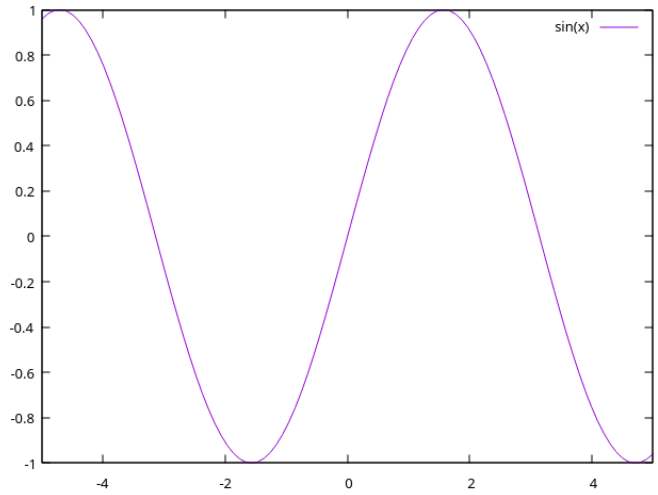
```
gnuplot> plot [0:4][1:2] 'parab.dat' w lp
```



Z pewnością nie jest to zbyt dobry wykres, niemniej ilustruje to co chcemy przetestować. Jak wspomniałem na samym początku, możliwe jest również ręczne skalowanie myszą (prawy przycisk).

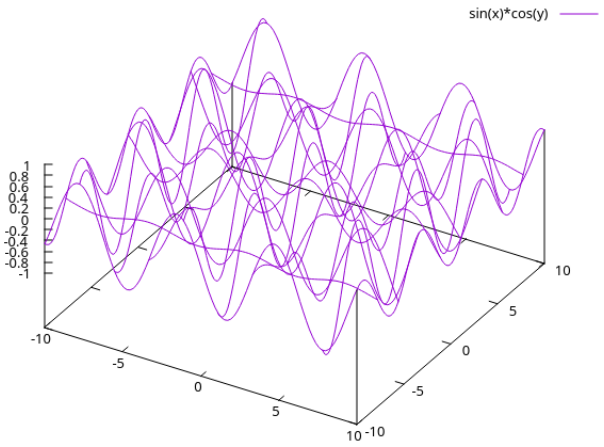
Gnuplot umożliwia oczywiście nie tylko rysowanie danych dyskretnych, ale również funkcji matematycznych.

```
gnuplot> plot [-5:5] sin(x)
```



Możemy to robić także w trzech wymiarach.

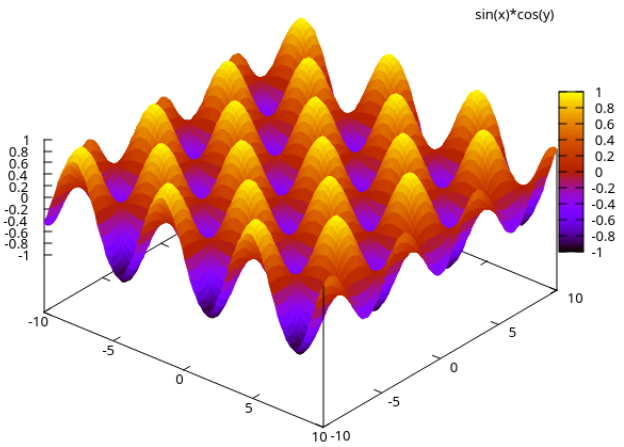
```
gnuplot> splot [-10:10][-10:10] sin(x)*cos(y)
```



W tej postaci nie jest to zbyt czytelne, toteż spróbujmy inaczej wyrenderować powierzchnię wykresu:

```
gnuplot> set isosamples 100
gnuplot> splot [-10:10][-10:10] sin(x)*cos(y) w pm3d
```

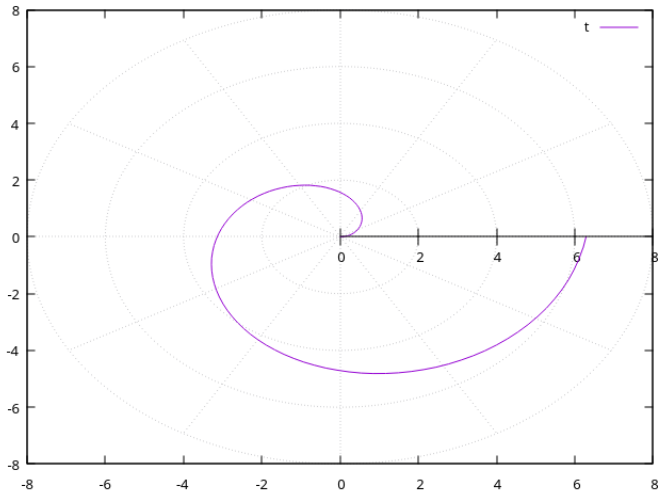
Pierwsza komenda zwiększa liczbę próbek (by powierzchnie nie były „kanciaste”).



Gnuplot oferuje także możliwość rysowania wykresów we współrzędnych biegunowych:

```
gnuplot> set polar
gnuplot> set grid polar
```

```
gnuplot> plot t
```



Ostatnią funkcją Gnuplota, o jakiej chciałbym wspomnieć, jest możliwość dopasowywania funkcji do danych („fitowania"). Spróbujmy np. dopasować parabolę do podanych na początku danych:

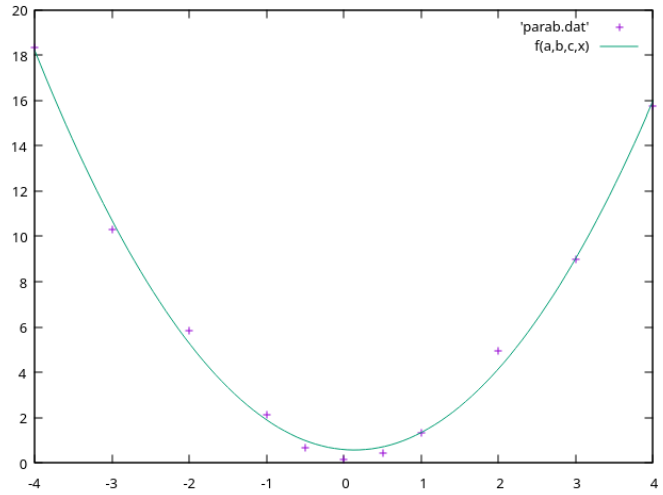
```
gnuplot> f(x)=a*x**2+b*x+c
gnuplot> fit f(x) 'parab.dat' via a,b,c
```

Powinniśmy dostać:

```
iter      chisq       delta/lim  lambda   a             b             c
   0 1.0201040000e+02   0.00e+00  4.86e+00    1.000000e+00   1.000000e+00   1.000000e+00
   1 1.0051708032e+01  -9.15e+05  4.86e-01    1.003279e+00   7.791976e-02   9.203348e-01
   2 1.5972121421e+00  -5.29e+05  4.86e-02    1.029968e+00  -2.807482e-01   6.091313e-01
   3 1.5961527625e+00  -6.64e+01  4.86e-03    1.031062e+00  -2.821487e-01   5.964354e-01
   4 1.5961527624e+00  -9.82e-06  4.86e-04    1.031062e+00  -2.821488e-01   5.964302e-01
iter      chisq       delta/lim  lambda   a             b             c

After 4 iterations the fit converged.
final sum of squares of residuals : 1.59615
rel. change during last iteration : -9.82079e-11

degrees of freedom    (FIT_NDF)                        : 8
rms of residuals      (FIT_STDFIT) = sqrt(WSSR/ndf)    : 0.446676
variance of residuals (reduced chisquare) = WSSR/ndf   : 0.199519

Final set of parameters            Asymptotic Standard Error
=======================            ==========================
a               = 1.03106          +/- 0.02305      (2.236%)
b               = -0.282149        +/- 0.05743      (20.35%)
c               = 0.59643          +/- 0.185        (31.01%)

correlation matrix of the fit parameters:
                a      b      c
a               1.000
b              -0.000  1.000
c              -0.685  0.000  1.000
```

Dostaliśmy poszukiwane wartości współczynników, wraz z garścią informacji statystycznych. Możemy spróbować narysować otrzymaną funkcję:

```
gnuplot> plot 'parab.dat', f(x)
```



Jeśli chcemy wykorzystać zawarte w pliku błędy do nadania punktom różnych wag przy dopasowywaniu funkcji, wystarczy odrobinę zmodyfikować polecenie `fit`:

```
gnuplot> fit f(x) 'parab.dat' yerror via a,b,c
```

Jak widać, Gnuplot nie jest ograniczony do regresji liniowej.

Pytania kontrolne

1. Które polecenie spowoduje narysowanie wykresu punktów przechowywanych w pierwszej i czwartej kolumnie pliku `plik.dat`, na którym punkty będą zarówno zaznaczone krzyżykami oraz połączone liniami?

   (a) `plot 'plik.dat' with lines`
   (b) `plot [1:4] 'plik.dat' w lp`
   (c) `plot 'plik.dat' u 1:4 w lp`
   (d) `plot 'plik.dat' via 1:4 w lp`
2. Jakiego fragmentu brakuje w poleceniu rysującym funkcję sinus w zakresie od $x = -1$ do $x = 4$?
   `plot sin(x)`
3. Jakiego fragmentu brakuje w poleceniu rysującym stosunek wartości w czwartej kolumnie do wartości w kolumnie trzeciej jako funkcji wartości z kolumny pierwszej?
   `plot 'plik.dat' w lp`