Wstęp do gnuplota

Dominik Perykasza

Spis treści

1 Wstęp

Strona gnuplota z dokumentacją i przykładami: `http://www.gnuplot.info`

Na ćwiczeniach korzystamy z dwóch trybów pracy z gnuplotem:

- W linii poleceń po wpisaniu komendy `gnuplot`
- Poprzez tworzenie skryptów i wywoływanie ich przez `gnuplot nazwa_skryptu.pg`

Z lini poleceń możemy wywołać skrypt poleceniem `load ''nazwa_skryptu.pg''`. Podane rozszerzenie pliku nie jest obowiązkowe - ma nas jedynie informować o tym, że zawarliśmy w nim komendy gnuplota.

Po uruchomieniu gnuplota możemy skorzystać z pomocy. Wpisanie komendy `help` spowoduje wyświetlenie podstawowych informacji o programie i listę sugerowanych tematów. Wywołanie komendy `help nazwa_polecenia` pozwala na uzyskanie szczegółowych informacji na temat interesującej nas instrukcji.

2 Rysowanie wykresów

Wykresy w programie gnuplot tworzymy za pomocą komend `plot` i `gnuplot`. Pierwsza służy do rysowania wykresów dwu wymiarowych, druga zaś do rysowania dwuwymiarowych rzutów powierzchni 3d. Składnie obu poleceń są zbliżone, parametry w nawiasach {} są opcjonalne:

```
plot {<ranges>}
     {<function> | {"<datafile>" {datafile-modifiers}}}
     {axes <axes>} {<title-spec>} {with <style>}
     {, {definitions,} <function> ...}
```

`<function> | ''<datafile>'' datafile-modifiers` Parametrem wjeściowym dla komendy `plot` jest funkcja lub plik z danymi. Szerszy opis dostępnych tu opcji znajduje się w dalszych częściach skryptu.

`<ranges>` W tym parametrze definiujemy zakresy wartości x,y dla których chcemy wygenerować rysunek. Dla komedy plot w trybie domyślnym(jeśli nie podamy tego parametru) zakres dla x to [-10:10], zaś zakres dla y jest automatycznie dobierany(skalowany) podczas generowania rysunku z funkcją.

`axes <axes>` Gdy gdy używamy komendy `plot` istnieją cztery zestawy osi, do których możemy wyskalować rysowaną funkcję. Używając tego określenia wybieramy je za pomocą jednego z kluczy ¡axes¿: `x1y1` - dolna i lewa oś, `x2y2` - prawa i górna oś, `x1y2` - dolna i prawa oś, `x2y1` - górna i lewa oś. Domyślnie wykres skalowany jest oczywiście do osi `x1y1`, co więcej zakres ustawiany w parametrze `<ranges>` odnosi się wyłącznie do tych osi.

`<title-spec>` Definicja nazwy funkcji

`with <style>` Wybór sposobu w jaki ma być rysowana funkcja np. za pomocą linii, punktów, itp.

`definitions` Definicje zmiennych i funkcji.

Przykład 1:
Polecenie

```
plot sin(x)
```

wykreśli funkcje sin(x) w domyślnym dla x przedziale $< -10, 10 >$ i wyskaluje oś y

Przykład 2:

```
plot [-pi pi] [-6.5 6.5] x**2*sin(x)
```

wykreśli funkcje $x^2 sin(x)$ dla x zmieniającego się w przedziale $< -10, 10 >$, y zmieniającego się w przedziale $< -6.5, 6.5 >$

Przykład 3:

```
plot sin(x) title "funkcja sinus", cos(x) title "funkcja cosinus"
```

ten przykład demonstruje sposób rysowania na jednym wykresie kilku funkcji. Funkcjom zostaną nadane nazwy "funkcja sinus" oraz "funkcja cosinus"

Przykład 4:

```
splot sin(x*y)
```

Tak działa komenda splot

2.1 Modyfikacja parametrów wykresu

Tworząc rysunek w gnuplocie możemy określić jego parametry(np. tytuł, podpisy osi) poprzez użycie komendy `set`. Powrót do domyślnych ustawien gnuplota osiąga się poprzez wydanie komendy `reset`. Wyświetlenie wszystkich parametrów nastąpi po wydaniu komendy `show all`, aby wyświetlić wartość konkretnego parametru używamy komendy `show nazwa_parametru`

Przykład 5:

```
set title "Wykres funkcji sinus"
set xlabel "kat"
set ylabel "y=sin(x)"
set xrange [0: 2*pi]
set xtics ("0" 0, "" pi/4, "90" pi/2, "" 3*pi/4, "180" pi, \
"" 5*pi/4, "270" 3*pi/2, "" 7*pi/4, "360" 2*pi)
set grid
plot sin(x) title "sinus"
set xtics 0, pi/4
replot
```

W powyższym przykładzie komendą `title` dodaliśmy do wykresu nazwę, która pojawi się centralnie w jego górnej części oraz zdefinowaliśmy podpisy osi `x`, `y` i określiliśmy przedział zmiennej `x`. Komendą `set xtics (''0'' 0, ...` ustawiliśmy napisy na osi `x`, tak aby podane były w stopniach. Komendą `set grids` dodaliśmy do rysunku siatkę pomocniczą. Wykres narysowaliśmy komendą `plot` nadając funkcji nazwę "sinus"(**uwaga** komenda `set title ''Wykres funkcji sinus''` i `plot ... title ''sinus''` odnosi się do różnych "części" wykresu) Ostatecznie stosując ponownie komendę `xtics` ustawiliśmy napisy na osi `x` tak aby podane były w radianach(napisy pojawią się począwszy od 0, co $\frac{\pi}{4}$). Komendą `replot` ponownie wygenerowaliśmy rysunek.

2.2 Modyfikowanie styli

W gnuplocie mozemy oczywiście zmieniać zarówno styl linii z jakim rysujemy funkcję lub krzywą, kolor linii, szerokość, typ punktów oraz ich rozmiar. Bardzo pomocną komendą jest komenda `test`, która wyświetla dostępne w danym terminalu kolory, typy punktów i przykładowe grubości linii. Styl linii zmieniamy stosując określenie `with` w komendzie `plot`.

Przykład 6:
Komenda

```
plot sin(x) with linespoints
```

wyrysuje nam funkcje w postaci krzyżyków połączonych linią.
Inne możliwe stle linii to np.`lines`(domyślny), `impulses`, `boxes`, `dots`, `points`, `steps`

Aby zmienić kolor linii(typ linii w terminalach nie zapewniających koloru), szerokość linii, typ punktów, rozmiar punktów odwołujemy się do następujących określeń: `linetype`, `linewidth`, `pointtype`, `pointsize` lub ich skrótów: `lt`, `lw`, `pt`, `ps`

Przykład 7:
Komenda

```
plot sin(x) with linespoints lt 2 lw 2 pt 12 ps 3, cos(x)
```

lub jej odpowiednik

```
plot sin(x) with linespoints linetype 2  \
   linewidth 2 pointtype 12 pointsize 3, cos(x)
```

wyrysuje nam funkcję $sin(x)$ za pomocą punktów połączonych linią(`linespoints`), kolorem zielonym(`lt 2`) i o szerokości równej 2 jednostkom(`lw 2`). Wybranym typem punktów są romby(`pt 12`) zaś ich rozmiar to 3 jednostki(`ps 3`)

Możliwe jest definiowanie własnych stylów linii i odwoływanie się później do nich za pomocą indeksów. Stosujemy w tym celu komendę `set linestyle numer_indeksu...` i określenia `lt`, `lw`, `pt`, `ps`

Przykład 8:
Komendy

```
set linestyle 1 lt 2 lw 2 pt 12 ps 3
set linestyle 2 lt 3 lw 2 pt 3 ps 3
plot sin(x) with linespoints ls 1, cos(x) ls 2
```

definiują style linii o indeksach 1 i 2. Do tak zdefiniowanych stylów odwołujemy się poprzez indeks w komendzie `plot` za pomocą określenia `ls numer_indeksu` lub `linestyle numer_indeksu`

3 Definiowanie funkcji i zmiennych

Podczas definiowania zmiennych istotne jest to, czy przypisując wartość zmiennej w momencie jej deklarowania użyjemy kropki czy nie. W pierwszym przypadku zdefinijemy zmienną rzeczywistą, w drugim całkowitą. Wpłynie to oczywiście na wyniki operacji na zmiennych. Zdefiniowane zmienne możemy w każdej chwili wyświetlić przy pomocy komendy `show variables`

Przykład 1:

```
a = 1.0 # Definicja zmiennej rzeczywistej
b = a/2 = 0.5
```

Przykład 2:

```
a = 1 # Definicja zmiennej całkowitej
b = a/2 = 0
```

W gnuplocie oprócz zmiennych możemy definiować własne funkcje a następnie odwoływać się do nich na przykład podczas tworzenia rysunków(`plot`) lub dopasowywania funkcji(`fit`) do danych pomiarowych. W zależności od trybu pracy(nieparametryczny/parametryczny) funkcje definiujemy przy pomocy zmiennych x,y lub parametrów t, u, v. Zdefiniowane funkcje możemy w każdej chwili wyświetlić przy pomocy komendy `show functions`

Przykład: 3(tryb nieparametryczny)
`f(x) = x**3+x+1` oznacza $f(x, y) = x^3 + x + 1$

Przykład: 4(tryb nieparametryczny)
`g(x,y) = (x-1)**2+y(-2)**2` oznacza $g(x, y) = (x - 1)^2 + (y - 2)^2$

Po zdefiniowaniu powyższych funkcji możemy je na przykład narysować:

```
plot f(x)
splot g(x,y)
```

Przykład: 5(tryb parametryczny)
`h(t) = cos(t)`

Przyklad: 6(tryb parametryczny)
`k(u,v) = cos(u)*sin(v)`

Również tymi funkcjami możemy się np. posiłkować przy tworzeniu wykresów:

```
plot h(t), sin(t)
splot k(u,v), u, v
```

Podczas definiowania funkcji i pracy z gnuplotem przydatne jest następujące wyrażenie znane m.in. z języka C:

`a?b:c`

Jego znaczenie jest następujące: jeśli `a` jest prawdą wykonaj `b`, jeśli nie wykonaj `c`. Zalety tego wyrażenia widoczne są w poniższym przykładzie.
Przykład: 7(tryb nieparametryczny)
`h(x) = (x<0) ? -x : x` oznacza $h(x) = \begin{cases} -x & \text{dla } x < 0 \\ x & \text{dla } x > 0 \end{cases}$

Aby poznać operatory i funkcje dostępne w gnuplocie wystarczy odwołać się do następujących tematów pomocy:

```
help expressions operators
help expressions function
```

4 Wstawianie rysunków do LaTeXa

Jedną z podstawowych zalet gnuplota jest możliwość zapisania rysunków w wielu różnych formatach. Format pliku w jakim zostanie zapisany rysunek ustawiamy komendą `set terminal nazwa_terminala`. W linuxie domyślnym "terminalem" jest X11. Oznacza to, że wydanie komendy `plot` spowoduje po prostu wyświetlenie rysunku w środowisku graficznym, w którym pracujemy. Aby wyświetlić listę wszystkich obsługiwanych terminali należy wydać komendę `set terminal`
Możemy na conajmniej dwa sposoby wygenerować rysunek gnuplotem i wstawić go do LaTeXa

Pierwszy sposób:

W gnuplocie generujemy rysunek w formacie `eps`

```
set terminal postscript eps color # Format postscript eps, z opcją color
set output "wykres.eps"   # Wybieramy nazwę pliku i otwieramy plik
plot sin(x)    # Wydanie komendy plot powoduje zapis rysunku do pliku
set output    # Zamykamy plik
set terminal X11   # Przywracamy terminal domyślny np. do dalszej pracy
```

W LaTeXu korzystamy ze środowiska `graphicx` i komendy `\includegraphics`
i generujemy następujący dokument:

```
\documentclass[12pt,a4paper]{article}

% Dodajemy pakiety do właściwej obsługi kodowania wprowadzanych
% znaków, fontów i dzielenia wyrazów
\usepackage[T1]{fontenc}
\usepackage[latin2]{inputenc}
\usepackage[polish]{babel}

% Dodajemy pakiet umożliwiający wstawianie plików
% graficznych do dokumentu
\usepackage{graphicx}

\begin{document}
% \listoffigures to nieobowiązkowa komenda tworząca w miejscu
% jej wpisania wykaz rysunków zamieszczonych w wdokumencie
\listoffigures
\begin{figure}[hb]
\begin{center}
\includegraphics[width=0.5\textwidth]{wykres}
\caption{To jest podpis pod rysunkiem}\label{fig:sinus}
\end{center}
\end{figure}

Na rysunku (\ref{fig:sinus}) widzimy wykres funkcji $y=\sin(x)$
```

Drugi sposób:

W gnuplocie generujemy plik LaTeXa

```
set terminal late # Format generowanego pliku to dokument LaTeXowy
set output "wykres.tex" # Wybieramy nazwę pliku i otwieramy plik
plot sin(x)  # Wydanie komendy plot powoduje zapis rysunku do pliku
set output  # Zamykamy plik
set terminal X11 # Przywracamy terminal domyślny np. do dalszej pracy
```

W LaTeXu dodajemy tak wygenerowany plik poleceniem `include`

```
\documentclass[12pt,a4paper]{article}

% Dodajemy pakiety do właściwej obsługi kodowania wprowadzanych
% znaków, fontów i dzielenia wyrazów
\usepackage[T1]{fontenc}
```

```
\usepackage[latin2]{inputenc}
\usepackage[polish]{babel}

\begin{document}
% \listoffigures to nieobowiązkowa komenda tworząca w miejscu
% jej wpisania wykaz rysunków zamieszczonych w wdokumencie
\listoffigures
\begin{figure}[hb]
\input{wykres.tex}
\caption{To jest podpis pod rysunkiem}
\end{figure}
\end{document}
```

Oprócz wspomnianego środowiska i polecenia `\includegraphics` korzystaliśmy dodatkowo ze środowiska `figure`, które służy do tworzenia wstawek. Wywołanie środowiska z argumentem [hb] instruuje LaTeXa co do sposobu umieszczania wstawki w dokumencie. W tym przypadku LaTeXw pierwszej kolejności ma próbować wstawić rysunek w miejscu, w którym wywołanie środowiska pojawia się w pliku źródłowym(h) a następnie na dole strony(b). Więcej o środowisku `figure` i mechanizmach zarządzania wstawkami przez LaTeXa należy doczytać z "Nie za krótkiego wprowadzenia do systemu LaTeX"

5 Rysowanie na podstawie danych z pliku

Plik z danymi powinien być w określonym formacie. Kolumny powinny być oddzielone spacjami. Standardowo pierwsza kolumna zwiera współrzędne x zaś druga y. Możliwe jest podanie tylko jednej kolumny, w tym przypadku dla współrzędnej x gnuplot generuje wartości całkowite zaczynając od 0. Miejsca dziesiętne odzielone są kropkami a nie przecinkami. Możliwe jest stosowanie zapisu w formacie 4.1e-10, 1e+2 itp. Komentarz w pliku oznaczany jest symbolem #.
Przykładowy plik:

```
#
# W komentarzu mozemy na przykład umieścić informacje o tym
# jakiego rodzaju dane zawiera nasz plik
#
857.7 855.5 2.3 2.0
```

```
848.7 851.8 5.5 5.7
855.1 851.5 1.1 0.3
854.0 852.6 1.8 1.6
850.2 854.6 2.0 3.8
845.6 844.5 4.1 3.4
847.9 846.8 4.8 4.5
```

Przykład 1:

```
plot 'nazwa_pliku'
```

wyrysuje nam zestaw punktów biorąc za współrzędne x wartości z pierwszej kolumny a za współrzędne y wartości z 2 kolumny.

Przykład 2:

```
plot 'nazwa_pliku' with yerrorbars
```

wyrysuje nam zestaw punktów biorąc za współrzędne x wartości z pierwszej kolumny a za współrzędne y wartości z 2 kolumny. Z trzeciej kolumny pobrane zostaną wartości Δy, które posłużą do wyrysowania błędów y w postaci pionowych linii od $(x, y - \Delta y)$ do $(x, y + \Delta y)$

Jeżeli w pliku z danymi mamy wiele kolumn i chcemy wskazać, które z nich gnuplot ma przetwarzać, powinniśmy użyć określenia `using`.

Przykład 3:

```
plot 'nazwa_pliku' using 2:3:5 with yerrorbars
```

wyrysuje nam zestaw punktów biorąc za współrzędne x wartości z drugiej kolumny a za współrzędne y wartości z 3 kolumny. Z piątej kolumny pobrane zostaną wartości Δy, które posłużą do wyrysowania błędów y, również w postaci pionowych linii od $(x, y - \Delta y)$ do $(x, y + \Delta y)$

6 Dopasowywanie funkcji do danych

Polecenie `fit` umożliwia dopasowanie zdefiniowanej przez nas funkcji do zbioru danych pomiarowych. Jego działanie oparte jest na metodzie najmniejszych kwadratów.

Składnia polecenia, parametry w nawiasach {} są opcjonalne.:

```
fit {[xrange] {[yrange]}} <function> '<datafile>'
    {datafile-modifiers}
    via '<parameter file>' | <var1>{,<var2>,...}
```

`[xrange]` `[yrange]` Zakresy zmiennych, w obrębie których chcemy dopasowywać funkcję. Wszystkie zmienne spoza zakresu zostaną zignorowane.

`<function>` Wyrażenie definiujące dopasowywaną funkcję lub odwołanie do wcześniej zdefiniowanej funkcji np. f(x) lub f(x,y).

`<datafile>` Plik z danymi, do których dopasowywana jest funkcja. Określenia służące sprecyzowania sposobu odczytu danych z pliku są takie same jak w poleceniu `plot`(z wyjątkiem `smooth` i `thru`, które nie mają tu zastosowania).

textttvia Określenie precyzujące, które z parametrów funkcji mają być optymalizowane. Jak widać w składni polecenia można je podać albo bezpośrednio oddzielając przecinkami albo poprzez odwołanie do pliku z definicjami parametrów. Jeśli podajemy je bezpośrednio, wówczas wszystkie zmienne, które nie zostały wcześniej zdefiniowane zostają utworzone i przypisana im zostaje wartość 1.0. W przypadku gdy odwołujemy się do pliku z definicjami parametrów powinniśmy zadbać o to by miał on następującą postać:

```
nazwa_parametru = wartość_początkowa
```

przy czym definicja każdego z parametrów powinna być podana w osobnej linii.

Domyślny format danych do dopasowywania funkcji jednej zmiennej, to plik z dwoma kolumnami x:y lub plik z trzema kolumnami x:y:s. W drugim przypadku trzecia kolumna powinna zawierać wartości określające odchylenie standardowe y, które są używane jako wagi dla danego punktu pomiarowego $\left(\frac{1}{s^2}\right)$. Jeśli nie podamy trzeciej kolumny wagi wszystkich punktów ustawione są na 1. Jeśli nie skorzystamy z polecenia `using` odchylenia standardowe dla y nie zostaną odczytane nawet jeśli plik zawiera trzecią kolumnę.

Przykład 1:

```
f(x) = a*x**2 +b*x + c
FIT_LIMIT = 1e-6
fit f(x) 'measured.dat' via 'start.par'
```

W powyższym skrypcie kolejno: definujemy funkcję, ustawiamy wartość parametru `FIT_LIMIT` na `1-e6`, nakazujemy dopasowanie funkcji `f(x)` do danych zlokalizowanych w pliku `measured.dat` z optymalizacją parametrów zdefiniowanych w pliku `start.par`

Ponieważ dopasowanie polega na numerycznej minimalizacji wartości funkcji χ^2 dla różnych kombinacji zadeklarowanych parametrów (w tym przypadku `a,b,c`), konieczne jest określenie momentu do którego algorytm będzie próbował zminimalizować tą wartość. Można to zrobić poprzez ustawienie parametru `FIT_LIMIT`, wówczas jeśli pomiędzy dwoma krokami iteracji algorytm wykryje, że wyliczana wartość zmieniła się mniej niż `FIT_LIMIT` proces dopasowania zostanie uznany za zakończony. Alternatywną metodą jest ustawienie parametru `FIT_MAXITER` określającego maksymalną liczbę iteracji.

Przykład 2:

```
f(x) = a*x**2 +b*x + c
FIT_LIMIT = 1e-6
fit f(x) 'measured.dat' using 3:($7-5) via 'start.par'
```

Dopasowujemy funkcję do danych znajdujących się w pliku `'measured.dat'`, odczydując z niego kolumnę 3 dla wartości x zaś współrzędną y odpowiadającą danemu x odczytujemy z kolumny 7 pomniejszając ją o 5. Optymalizaowane parametry definiujemy w pliku `start.par`

Przykład 3:

```
f(x) = a*x**2 +b*x + c
a = 4.0
b = 3.0
c = 1.0
FIT_LIMIT = 1e-6
fit f(x) './data/trash.dat' using 1:2:3 via a, b, c
```

Dopasowujemy funkcję do danych znajdujących się w pliku `trash.dat`. Plik zawiera trzecią kolumnę z błędami, dlatego nakazujemy jej wykorzystanie określeniem `using 1:2:3`. Optymalizowane parametry podajemy bezpośrednio. W tym przykładzie optymalizowanym parametrom przypisano konkretne wartości przed wywołaniem komendy `fit` po to aby "wspomóc" algorytm dopasowujący zadaną funkcję.