Лабораторна робота 13. Побудова графіка функції

Зміст заняття: робота з програмою gnuplot, графічне представлення функціональних залежностей і табличних даних

Загальні відомості

Результати роботи якоїсь C++- програми часто виводяться не тільки на екран, але і в файли на диску. Завжди варто зберегти результати обрахунків для їх подальшого уважного аналізу або введення в якості вихідних даних іншої програми. Прикладом такої іншої програми може служити `gnuplot`, якщо є бажання побачити результати у вигляді графіків.

`Gnuplot` – потужна утиліта для побудови графіків, заданих аналітичними залежностями, таблично (експериментальні дані та дані чисельного моделювання), підтримує командний режим роботи та написання сценаріїв (скриптів).

Дистрибутиви для різних ОС, а також приклади побудови графіків можна завантажити за наступними посиланнями:

http://sourceforge.net/projects/gnuplot/files/

http://gnuplot.sourceforge.net/demo/

Робота в gnuplot здійснюється в наступних режимах:

• пакетний. Користувач готує файл, який містить послідовний набір команд (нова команда починається з нового рядка). При роботі в такому режимі командний файл повинен знаходитися в одній директорії з виконуваним файлом gnuplot.

• інтерактивний. Користувач спілкується з програмою за допомогою командного рядка в режимі реального часу.

Розглянемо приклади того, яким чином можна зображувати дані на координатній площині. Нехай є деякий набір експериментальних даних. Для визначеності розглянемо дані експерименту по вимірюванню відхилення маятника від положення рівноваги в залежності від часу. Будемо вважати, що ці дані записані в текстовий файл 'data.txt'. Розглянемо різні способи візуалізації цих даних.

Для початку зобразимо ці дані на координатній площині. Для цього в програмі gnuplot потрібно задати команду:

plot ’data.txt’

Будуть намальовані точки, що відповідають даним з файлу 'data.txt'.

При цьому перша колонка даних (момент часу вимірювання) буде відображена по осі *x*, а друга колонка (відхилення від положення рівноваги) - по осі *y*.

Для побудови графіка функції на площині також використовується команда plot:

Як приклад побудуємо графік синусоїди, на проміжку від -2π до 2π:

plot [-2*pi:2*pi] sin(x)

Область зміни значень аргументу / функції можна задати до побудови, для цього потрібно ввести:

set xrange [<поч. значення>: <кінц. значення>]

set yrange [<поч. значення>: <кінц. значення>]

Для більшої наочності можна намалювати сітку на графіку. Це робиться командою:

set grid

Після задання сітки треба заново перемалювати графік. Зробити це можна, або задавши знову команду на малювання графіка, або задавши команду **replot**. Команда replot без аргументів діє просто: вона перемальовує тt, що було намальовано до цього, тобто виконує останню команду plot.

З'єднання окремих точок відрізками

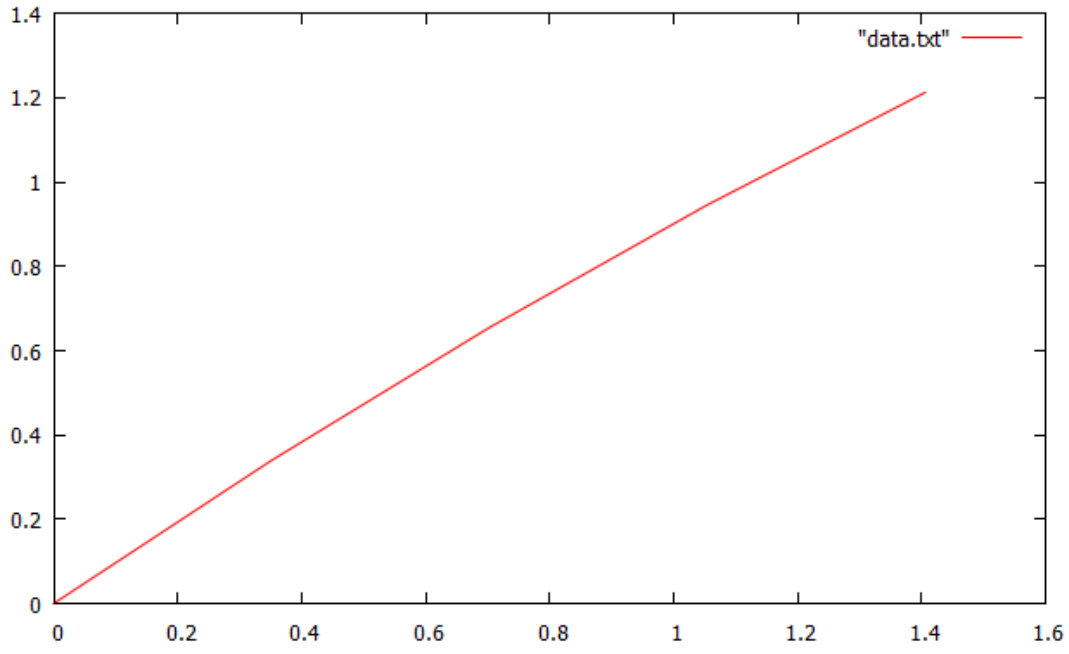
Ламана лінія, що з'єднує експериментальні точки, в програмі gnuplot будується наступним чином:

plot “data.txt” with lines

Приклад 13.1. Припустимо, є такий файл:

```
# data.txt
# x y
0  0
0.351506  0.339385
0.703012  0.654529
1.05452  0.94543
1.40602  1.21209
```

plot "data.dat" with lines # або: **p "data.txt" w l**



Якщо потрібно, щоб експериментальні точки, через які проходить ця лінія, також відображалися на графіку, задається трохи інша команда:

plot ’data. txt’ with linespoints

Попередні дві команди можуть бути записані більш компактно:

plot ’data. txt’ w l

plot ’data. txt’ w lp

При виведенні gnuplot дозволяє встановлювати різні візуальні параметри для графіка. Так, для відтворення самого графіка існує 8 простих і 14 розширених стилів. Для їх використання в команді plot після оголошення функції слід ввести:

with <стиль графіка> // linetype <тип, ціле число (комбінація стиль+колір)>

Деякі прості стилі:

lines (по замовч.) - лінії

points - точки

lines and points - лінії з точками

dots - дуже маленькі точки

impulses - дискретні прямі

steps - ламана під прямим кутом лінія

Для зміни кольору і товщини лінії графіка в команді plot потрібно вказати:

linestyle <колір, ціле число> // linewidth <товщина лінії, pt>

Ось деякі приклади:

Команда	Графік на екрані
plot ’data.txt’ with lines **plot ’data.txt’ with l** **plot ’data.txt’ w l** **plot ’data.txt’ with lines 1** **plot ’data.txt’ w l lt 1 lw 1** **plot ’data.txt’ w l lt 1 lw 5**	червона лінія те ж саме те ж саме те ж саме те ж саме лінія вп’ятеро товстіша
plot ’data.txt’ w points pt 5 ps 1 **plot ’data.txt’ w points pt 1 ps 3** **plot ’data.txt’ w p pt 1 ps 3**	червоні трикутники червоні ромбики те ж саме
plot ’data.txt’ with linespoint pt 1 ps 3 **plot ’data.txt’ with lp lt 2 lw 3 pt 1 ps 3**	те ж саме, тільки разом з лінією те ж саме, тільки змінили колір і товщину лінії

Згладжування ламаної лінії, що проходить через експериментальні точки

Для проведення плавної кривої необхідно використовувати додаткові алгоритми, що дозволяють проводити інтерполяцію, тобто знаходження значень, що лежать між експериментальними точками. Згладжування даних в програмі gnuplot можна робити різними способами - кривими Без'є або сплайнами.

Згладжування кривими Без'є здійснюється командою:

plot ’data.txt’ smooth bezier

Згладжування кубічними сплайнами здійснюється командою:

plot ’data.txt’ smooth csplines

Вивід на один рисунок графіків декількох функцій

Вивід графіків декількох функцій досягається розташуванням в рядку команди plot відповідних формул через кому. Наприклад, команда:

plot [-3:3] [-1.2:1.5] sin(x), cos(x), 0.5*sin(2*x), 0.5*cos(2*x)

Щоб побудувати одночасно кілька графіків з даних, записаних в колонках одного файлу, потрібно кілька разів зчитати ці дані з файлу в одній і тій же команді plot

plot "xyz.dat" using 1:2 , "xyz.dat" using 1:3 , "xyz.dat" using 2:3

або

plot "xyz.dat" u 1:2 , "xyz.dat" u 1:3 , "xyz.dat" u 2:3

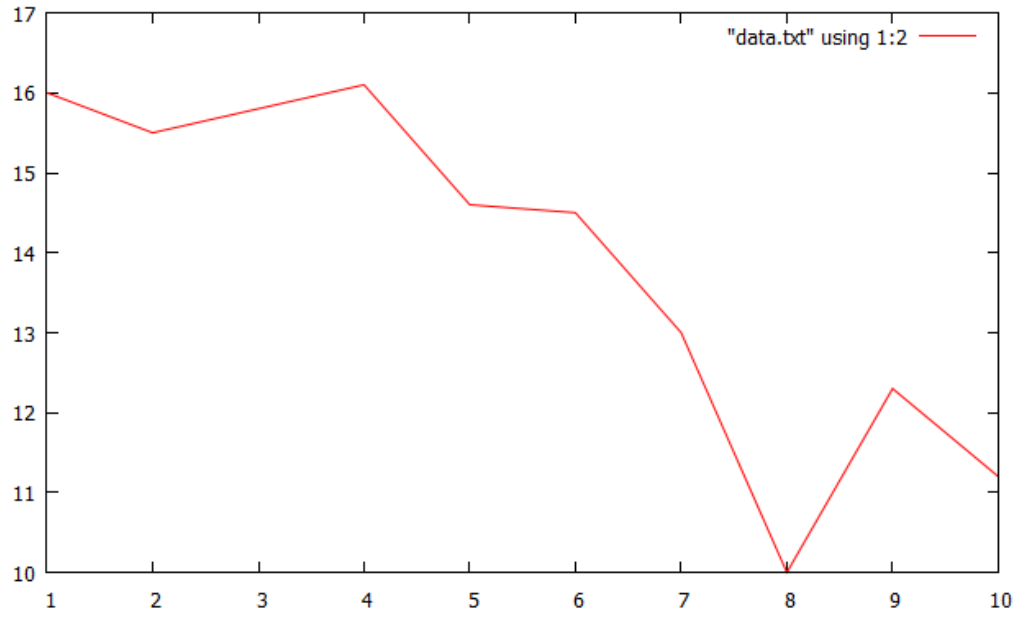
Тут команда plot в порівнянні з попередніми варіантами її використання містить опцію using, після якої пишеться номер стовпця аргументу і через двокрапку номер стовпчика, що відповідає поточному значенню функції. Відсутність опції using за замовчуванням еквівалентно using 1: 2, а при наявності в вихідному файлі лише одного стовпчика в якості аргументу береться номер точки.

Приклад 13.2. Маніпуляції з даними покажемо на прикладі файлу:

```
# data.txt
# x  y  z
1   16.0   40
2   15.5   30
3   15.8   35
4   16.1   38
5   14.6   39
6   14.5   40
7   13.0   38
8   10.0   39
9   12.3   39
10  11.2   37
```

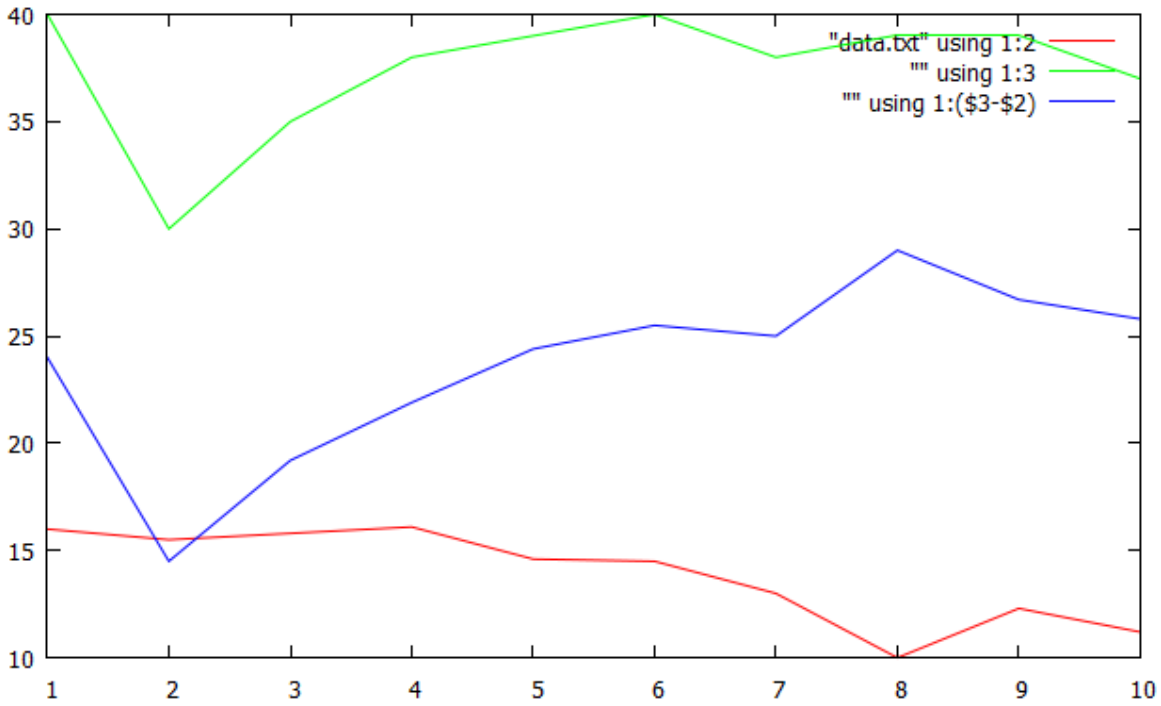
Можна побудувати графік за першими двома колонками:

plot "data.txt" using 1:2 with lines



Можна також побудувати графік за першою і третьою колонками та обчислити різницю між графіками

```
plot "data.txt" using 1:2 with lines,\
   "" using 1:3 with lines,\
   "" using 1:($3-$2) with lines
```



Додавання $ до номера колонки дозволяє використовувати останню в якості змінної, виконувати з колонками математичні операції і застосовувати до них функції.

Анімація створюється за заданими функціями та файлами даних. В другому випадку дані, що становлять кадр анімації, повинні відділятися від даних іншого кадру двома порожніми рядками. Подивимося (рис. 3), як будується анімація за даними з файлу projectile.dat.

Приклад 13.3.

```
set terminal gif animate delay 100
set output "projectile.gif"
set yrange [0:3]
set xrange [0:12]
do for [i=0:29] {
    plot "projectile.dat" index i  u 1:2 w p lt 7 lc rgb "black" ps 3
}
```

Дані виводяться в формат GIF (set terminal gif) з інтервалом між кадрами (delay) 100 мс. do for [i = 0: 29] {...} - цикл за кадрах анімації. Звернення до кожного кадру відбувається за його номером - index i.

Апроксимація дозволяє отримати рівняння кривої, яка найкраще відповідає точкам даних. Розглянемо дані з файлу projectile.dat і знайдемо криву, яка описує ці дані, тобто опишемо параболу

```
f(x) = a + b*x + c*x**2
```

Задамо її в командному рядку і вкажемо gnuplot знайти коефіцієнти a, b і c:

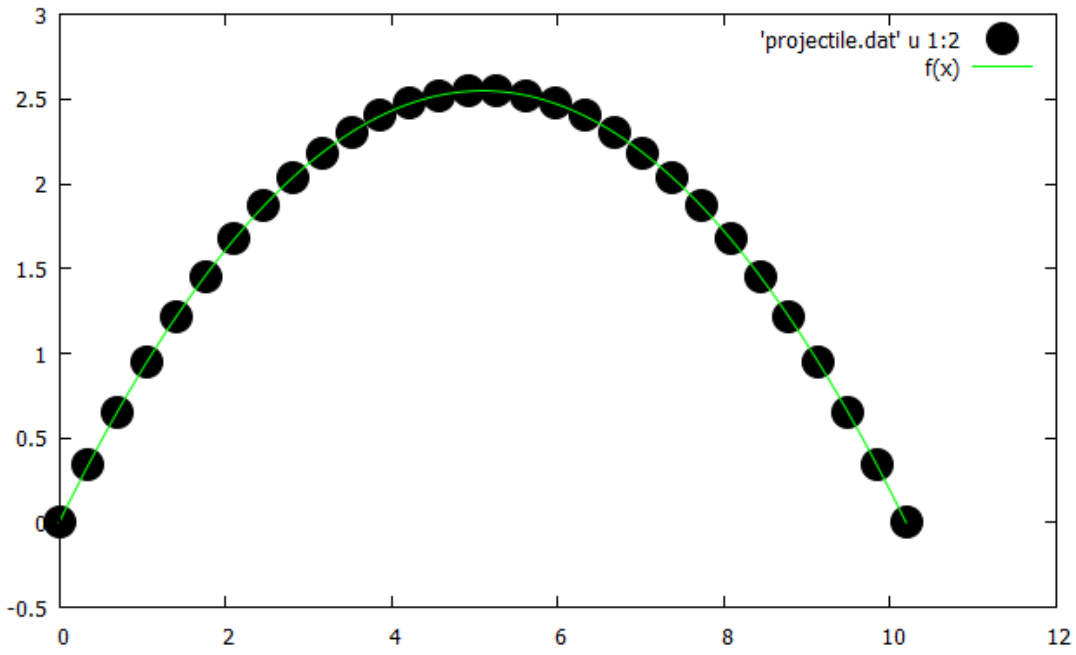
```
fit f(x) 'projectile.dat' via a,b,c
```

Отримаємо наступні значення:

```
a = 1.71135e-006
b = 0.999998
c = -0.0980998
```

Тепер подивимося, наскільки добре у нас вийшла апроксимація:

```
plot 'projectile.dat' u 1:2 w p lt 7 ps 3,\
     f(x)
```

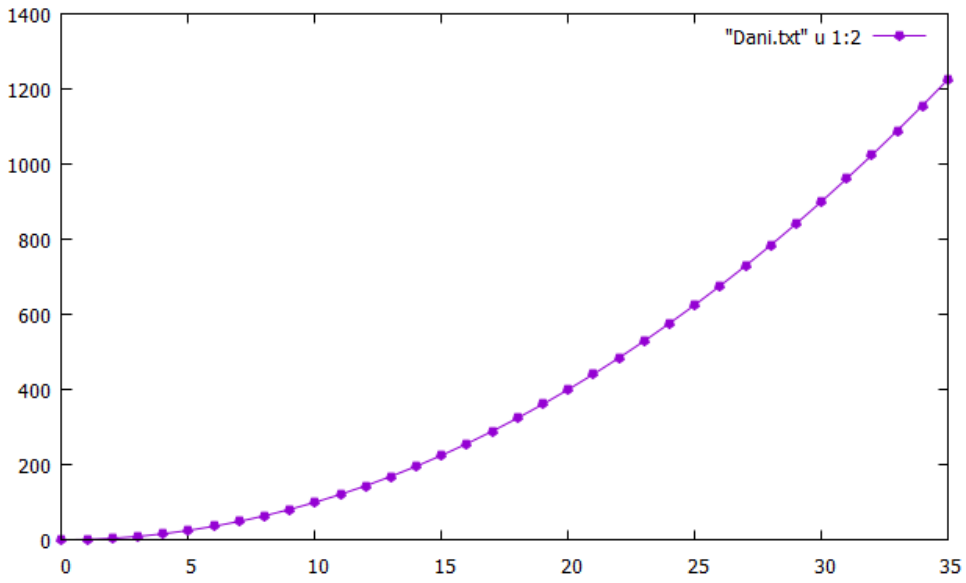


Виклик gnuplot із програми C++

```
#include <iostream>
#include <fstream>
#include <stdlib.h>

using namespace std;

int main()
{

    string sctName = "plotscript.gps"; // файл скрипта для gnuplot
    string fName = "Dani.txt";  //файл з данимим
    string gpCommand; // команда виклику gnuplot
    fstream fOut; // файлова змінна

    //створення файлу-сценарію для gnuplot
    fOut.open(sctName.c_str(), ios::out);
        // скрипт для одного графіка
    fOut<<"plot \""<<fName<<"\" u 1:2 ps 1 pt 15 with lp \n";
        fOut.close();

    // створення файлу з даними
```

```
    fOut.open(fName.c_str(), ios::out);
    for(int x=0;x<36;x++)
          fOut<<x<<"\t"<<x*x<<"\n";
    fOut.close();

        // команди виклику gnuplot
        gpCommand = "C:/gnuplot/bin/gnuplot.exe -persist "+sctName;
        system(gpCommand.c_str());
    return 0;
}
```

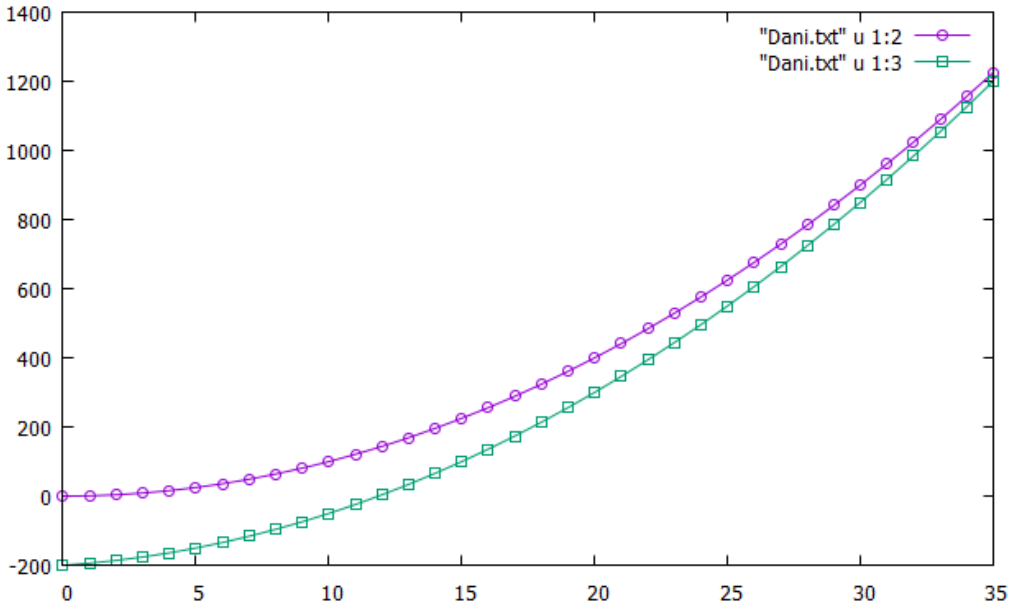
Результат виконання програми:




```
// скрипт для двох графіків
    fOut<<"plot \""<<fName<<"\" u 1:2 ps 1 pt 6 w lp,"<<"\""<<fName<<"\" u 1:3 ps 1 pt 4
w lp\n";
```

```
// створення файлу з даними
    fOut.open(fName.c_str(), ios::out);
    for(int x=0;x<36;x++)
          fOut<<x<<"\t"<<x*x<<"\n";
    fOut.close();
```

Результат виконання програми:



Завдання до лабораторної роботи

13.А.

Закон Максвелла описується деякою функцією $f(v)$, що називається функцією розподілу молекул за швидкостями руху:

$$f(\upsilon)=4\pi\left(\frac{m_0}{2\pi kT}\right)^{\frac{3}{2}}\upsilon^2 e^{-\frac{m_0\upsilon^2}{2kT}}$$

Конкретний вигляд функції залежить від роду газу (m_0) і від параметра стану (T).

13.1А. Графічно зобразити функцію розподілу молекул за швидкостями для 15 гр. водню за нормальних умов.

13.2А. Графічно зобразити функцію розподілу молекул за імпульсом для 1 кг молекул азоту за температури 273 К.

13.3А. Графічно зобразити функцію розподілу молекул за кінетичною енергією для 100 г молекул кисню за температури 20°С.

13.Б. Побудувати графіки систем нелінійних рівнянь .

Варіант №	Система
13.1	$\begin{cases} \sin(x+1)-y=1.2; \\ 2x+\cos y=2. \end{cases}$
13.2	$\begin{cases} \cos(x-1)+y=0.5; \\ x-\cos y=3. \end{cases}$
13.3	$\begin{cases} \sin x+2y=2; \\ x+\cos(y-1)=0.7. \end{cases}$
13.4	$\begin{cases} \cos x+y=1.5; \\ 2x-\sin(y-0.5)=1. \end{cases}$
13.5	$\begin{cases} \sin(x+0.5)-y=1; \\ \cos(y-2)+x=0. \end{cases}$
13.6	$\begin{cases} \cos(x+0.5)+y=0.8; \\ \sin y-2x=1.6. \end{cases}$
13.7	$\begin{cases} \sin(x-1)=1.3-y; \\ x-\sin(y+1)=0.8. \end{cases}$
13.8	$\begin{cases} 2y-\cos(x+1)=0; \\ x+\sin y=-0.4. \end{cases}$

13.9	$\begin{cases} \cos(x+0.5)-y=2; \\ \sin y-2x=1. \end{cases}$
13.10	$\begin{cases} \sin(x+2)-y=1.5; \\ x+\cos(y-2)=0.5. \end{cases}$
13.11	$\begin{cases} \cos(y-1)+x=0.5; \\ y-\cos x=3. \end{cases}$
13.12	$\begin{cases} \sin y+2x=2; \\ \cos(x-1)+y=0.7. \end{cases}$
13.13	$\begin{cases} \cos y+x=1.5; \\ 2y-\sin(x-0.5)=1. \end{cases}$
13.14	$\begin{cases} \cos(x+0.5)-y=2; \\ \sin y-2x=1. \end{cases}$