Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Фізико-технічний інститут

Комп'ютерний практикум № 2

Побудова графіка функції за алгоритмом Брезенхейма

предмет «Комп'ютерна графіка»

Роботу виконав:
Студент 3 курсу ФТІ, групи ФІ-91
Цибульник Антон Владиславович

Приймав:
Професор кафедри ІБ
Півень Олег Борисович

2022

Мета

Навчитися будувати за алгоритмом Брезенхейма графіки функцій, заданих рівняннями в параметричному вигляді та рівняннями у полярних координатах.

Завдання

Побудувати за алгоритмом Брезенхейма графіки функцій, заданих рівняннями в параметричному вигляді та рівняннями у полярних координатах.

Теоретичні відомості

Загальна постановка алгоритму Брезенхейма

Агналогічно до методу у Лабораторній роботі №1 (цифровий диференціальний аналізатор), алгоритм Брезенхейма теж обирає оптимальні растрові координати для представлення відрізку між двома точками. В процесі роботи спираємося на кутовий коефіцієнт нахилу. Одна з координат фіксується, а зміна іншої координати залежить від відстані між дійсним положенням відрізка та ближніми координатами сітки. Таку відстань називають помилкою й позначають як e. Величина помилки в наступній точці растру може бути обчислена таким чином: $e = e + m$, де m – кутовий коефіцієнт.

Отже, фіксуємо одну з координат, по іншій рухаємося. Задаємо певне від'ємне початкове значення помилки. Позначаємо точки поточної зафіксованої прямої на кординатній сітці доти, доки значення помилки не стане додатнім. А тоді корегуємо e (наприклад, відніманням одиниці), і одразу збільшуємо значення зафіксованої координати.

Цілочисельний алгоритм Брезенхейма

Оскільки важливий лише знак помилки, то просте перетворення $e_1 = 2ex$ перетворить алгоритм в цілочисельний і дозволить його ефективно організувати його на апаратному або мікропрограмному рівні.

Узагальнений алгоритм Брезенхейма

Щоб реалізація алгоритму була повною, необхідно обробляти відрізки у всіх квадрантах. Модифікацію можна провести, враховуючи в алгоритмі номер квадранту, в якому лежить відрізок та із врахуванням кутового коефіцієнту. Коли абсолютна величина кутового коефіцієнту більше 1, y постійно змінюється на 1, а критерій помилки Брезенхема використовується для прийняття рішення про зміну величини x. Вибір величини координати, що постійно міняється (на $+1$ чи -1) залежить, власне, від квадранту.

Варіант завдання

Спершу розглянемо криву, задану рівняннями **у параметричному вигляді**:

$$\begin{cases} x = at - b\sin(t) \\ y = a - b\cos(t) \end{cases} \quad t \in (0, 2\pi),\ a, b > 0 \tag{1}$$

Одразу знайдемо граничні значення змінних x та y, підставивши крайові величини параметра t у рівняння цієї системи:

$$t \in (0, 2\pi) \ \Rightarrow \begin{cases} x \in (0, 2\pi a) \\ y \in (a - b, a + b) \end{cases}$$

Алгоритм Брезенхейма так ніби підштовхує до використання кривої у декартовій системі координат, хоча це не є чимось обов'язковим: головне задати дякий набір значень – початки і кінці відрізків на графіку. Це можна зробити, використовуючи й параметричну форму рівняння. Проте, все ж поступово перетворимо систему (1) у декартові координати. Виразимо t із другого рівняння цієї системи:

$$t = \pm\arccos(\tfrac{a-y}{b}) + 2\pi n,\ n \in \mathbb{Z}$$

Виокремимо серед усіх розв'язків ті, які задовільняють обмеженню $t \in (0, 2\pi)$. Оскільки функція $\alpha = \arccos x$ приймає значення в проміжку від 0 до π, то нас влаштують лише такі розв'язки:

$$t = \begin{cases} \arccos(\frac{a-y}{b}), & t \in (0, \pi) \\ -\arccos(\frac{a-y}{b}) + 2\pi, & t \in (\pi, 2\pi) \end{cases}$$

Таким чином, підставивши наведені значення параметра t у друге рівняння системи (1) із урахуванням відповідних крайових величин, остаточно отримаємо задану криву у прямокутній системі координат:

$$x = \begin{cases} a\arccos(\frac{a-y}{b}) - b\sin(\arccos(\frac{a-y}{b})), & y \in (a-b, a+b),\ y\uparrow \\ -a(\arccos(\frac{a-y}{b}) - 2\pi) + b\sin(\arccos(\frac{a-y}{b}) - 2\pi), & y \in (a+b, a-b),\ y\downarrow \end{cases}$$

Дійшла черга до кривої, заданої **у полярних координатах**:

$$\rho^2 = 2a^2\cos(2\varphi),\ a \in \mathbb{R} \tag{2}$$

Це рівняння так званої лемніскати Бернуллі, яку згідно варіанту слід розглянути на двох різних проміжках: $\varphi \in (-\frac{\pi}{4}, \frac{\pi}{4})$ та $\varphi \in (-\frac{3\pi}{4}, \frac{5\pi}{4})$. У прямокутній системі координат ця крива виглядатиме так:

$$(x^2 + y^2)^2 = 2a^2(x^2 - y^2)$$

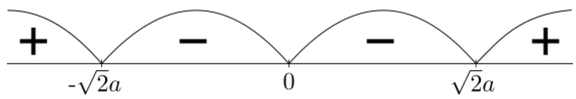
Покроково зведемо це рівняння до більш звичного вигляду $y = y(x)$. Спершу спростимо вираз, розкривши дужки. Опісля знайдемо дискримінант й розв'яжемо отримане біквадратне рівняння відносно y^2. Остаточно отримаємо такий результат:

$$y = \pm\sqrt{-x^2 - a^2 + \sqrt{4x^2a^2 + a^4}}$$

Знайдемо граничні значення x з умови невід'ємності підкореневого виразу:

$$-x^2 - a^2 + \sqrt{4x^2a^2 + a^4} \geqslant 0 \;\Rightarrow\; x^2(x^2 - 2a^2) \leqslant 0$$

Розв'яжемо отриману нерівність методом інтервалів. На малюнку праворуч зображені усі критичні точки. Отже, розв'язком є такий проміжок: $x \in$ [-$\sqrt{2}a, \sqrt{2}a$].



Оскільки значенням кута $\varphi \in$ (-$\frac{\pi}{4}, \frac{\pi}{4}$) відповідають додатні величини x, а значенням $\varphi \in$ (-$\frac{3\pi}{4}, \frac{5\pi}{4}$) – від'ємні, то завдання варіанту зведеться до розгляду таких двох кривих:

$$y_1 = \pm\sqrt{-x^2 - a^2 + \sqrt{4x^2a^2 + a^4}}, \qquad x \in [\text{-}\sqrt{2}a, 0]$$
$$y_2 = \pm\sqrt{-x^2 - a^2 + \sqrt{4x^2a^2 + a^4}}, \qquad x \in [0, \sqrt{2}a]$$

Вимоги до програми

1. Тестові значення кроку зміни аргументу t, параметрів функцій (a, b та інші) вибираються студентом самостійно і описуються в програмі як початкові значення, що використовуються для побудови графіків відразу після запуску програми.

2. Програма повинна:

   (а) дозволяти користувачеві вибирати за допомогою меню функцію та спосіб побудови графіка (з використанням стандартних вбудованих методів мови програмування та за алгоритмом Брезенхейма);

   (б) виводити на екран координатні осі та цифровану сітку (з точністю до округлених значень типу цілих, десятків тощо), підписи до вісей абсцис та ординат;

   (в) надавати можливість користувачеві змінювати параметри та крок побудови графіків, колір та товщину ліній графіка;

   (г) виконувати автоматичне масштабування побудови графіка при різних змінах параметрів.

Код програми й скріншоти результатів

Алгоритм Брезенхейма

```
import numpy as np
import matplotlib.pyplot as plt

def built_in(a,b,paint,width,equation,alpha):
    if (equation == 'param'):
        t = np.arange(0, 2*np.pi, 0.01)
        x = a*t - b*np.sin(t)
        y = a - b*np.cos(t)

        plt.grid()
        plt.plot(x, y, lw=width, color=paint)

        plt.xlabel('x values')
        plt.ylabel('y values')

        left, right = plt.xlim()
        down, up = plt.ylim()
        plt.xlim(left*alpha, right*alpha)
        plt.ylim(down*alpha, up*alpha)

        return plt.show()

    if (equation == 'polar'):
        phi1 = np.arange(-np.pi/4, np.pi/4, 0.0001)
        phi2 = np.arange(-3*np.pi/4, 5*np.pi/4, 0.0001)
        phi = np.append(phi1,phi2) # whole lemniscate of Bernoulli
        rho = np.sqrt(2*a*a*np.cos(2*phi1)) # or phi2

        plt.polar(phi1, rho, lw=width, color=paint)

        plt.yticks([40, 80, 120])

        left, right = plt.xlim()
        down, up = plt.ylim()
        plt.xlim(left*alpha, right*alpha)
        plt.ylim(down*alpha, up*alpha)

        return plt.show()

def Bresenham(a,b,paint,width,x,y,length,alpha):
    for i in range(length-1):
        x1 = x[i]
        y1 = y[i]
        x2 = x[i+1]
        y2 = y[i+1]

        x_draw = x1
        y_draw = y1

```

```python
        dx = abs(x2 - x1)
        dy = abs(y2 - y1)
        s1 = np.sign(x2 - x1)
        s2 = np.sign(y2 - y1)

        if dy > dx:
            dx, dy = dy, dx
            exchange = 1
        else:
            exchange = 0

        e = 2*dy - dx

        for j in range(int(dx)):
            plt.plot(int(x_draw), int(y_draw), 's', color=paint, markersize=width)

            while (e >= 0):
                if (exchange == 1):
                    x_draw = x_draw + s1
                else:
                    y_draw = y_draw + s2

                e = e - 2*dx

            if (exchange == 1):
                y_draw = y_draw + s2
            else:
                x_draw = x_draw + s1

            e = e + 2*dy

    plt.grid()

    plt.xlabel('x values')
    plt.ylabel('y values')

    left, right = plt.xlim()
    down, up = plt.ylim()
    plt.xlim(left*alpha, right*alpha)
    plt.ylim(down*alpha, up*alpha)

    return plt.show()
```

Інтерфейс для користувача

```
print('''Hi! Please, choose a curve:
1. x = at-b*sin(x)
   y = a-b*cos(x)

2. np.sqrt(2*a*a*np.cos(2*phi))''')

variant = int(input('\nYour answer: '))

if (variant == 1):
    equation = 'param'
if (variant == 2):
    equation = 'polar'

if (equation == 'param'):
    t = np.arange(0, 2*np.pi, 0.3)

    def X(t):
        return a*t - b*np.sin(t)
    def Y(t):
        return a - b*np.cos(t)

    x,y = [],[]
    for i in range(len(t)):
        x.append(X(t[i]))
        y.append(Y(t[i]))

    length = len(t)

if (equation == 'polar'):
    def Up(x):
        return np.sqrt(np.sqrt(pow(a,4) + 4*x**2*a**2) - x**2 - a**2)
    def Down(x):
        return -np.sqrt(np.sqrt(pow(a,4) + 4*x**2*a**2) - x**2 - a**2)

    R_up = np.arange(0, np.sqrt(2)*a, 5)
    R_down = np.arange(np.sqrt(2)*a, 0, -5)
    L_up = np.arange(0, -np.sqrt(2)*a, -5)
    L_down = np.arange(-np.sqrt(2)*a, 0, 5)

    x_phi1 = np.append(R_up, R_down)
    x_phi2 = np.append(L_up, L_down)
    x_whole = np.append(x_phi1, x_phi2) # whole lemniscate of Bernoulli
    x = x_phi1 # or x_phi2

    y = []
    for i in range(0, len(x)//2):
        y.append(Up(x[i]))
    for i in range(len(x)//2, len(x)):
        y.append(Down(x[i]))

    length = len(x)
```

```
print('''\nNow choose a way to draw this equation:
1. To draw via built-in features of Python
2. To draw via Bresenham algorithm''')

draw = int(input('\nYour answer: '))

# default values (can be changed)
paint = 'blue'
width = 3
alpha = 1

print('\nAnd finally set some necessary values:')
if (equation == 'param'):
    a = int(input('Set a parameter a: '))
    b = int(input('Set a parameter b: '))
if (equation == 'polar'):
    a = int(input('Set a parameter a: '))
    b = 0

if (draw == 1): built_in(a,b,paint,width,equation,alpha)
if (draw == 2): Bresenham(a,b,paint,width,x,y,length,alpha)

def switcher(answer,a,b,draw,paint,width,x,y,length,alpha):
    if (answer == '1'):
        a = int(input('Parameter a: '))
        if (draw == 1): return built_in(a,b,paint,width,equation,alpha)
        if (draw == 2): return Bresenham(a,b,paint,width,x,y,length,alpha)
    if (answer == '2'):
        b = int(input('Parameter b: '))
        if (draw == 1): return built_in(a,b,paint,width,equation,alpha)
        if (draw == 2): return Bresenham(a,b,paint,width,x,y,length,alpha)
    if (answer == '3'):
        paint = input('Color: ')
        if (draw == 1): return built_in(a,b,paint,width,equation,alpha)
        if (draw == 2): return Bresenham(a,b,paint,width,x,y,length,alpha)
    if (answer == '4'):
        width = int(input('Width: '))
        if (draw == 1): return built_in(a,b,paint,width,equation,alpha)
        if (draw == 2): return Bresenham(a,b,paint,width,x,y,length,alpha)
    if (answer == '5'):
        draw = int(input('''A way to draw:
1. To draw via built-in features of Python
2. To draw via Bresenham algorithm

Your answer: '''))
        if (draw == 1): return built_in(a,b,paint,width,equation,alpha)
        if (draw == 2): return Bresenham(a,b,paint,width,x,y,length,alpha)
    if (answer == '6'):
        alpha = float(input('Scaling coefficient: '))
        if (draw == 1): return built_in(a,b,paint,width,equation,alpha)
        if (draw == 2): return Bresenham(a,b,paint,width,x,y,length,alpha)
```

```
stop = False

while (stop != True):
    print('''\nYou can change some values:
1. Change parameter a:
2. Change parameter b:
3. Change a color of a curve
4. Change a width of a curve
5. A way to draw a line
6. Scaling
7. Quit ''')
    answer = input('\nYour answer: ')
    if (answer == '7'): stop = True
    else: switcher(answer,a,b,draw,paint,width,x,y,length,alpha)
```

Скріншоти кривої у параметричному виді (циклоїда)

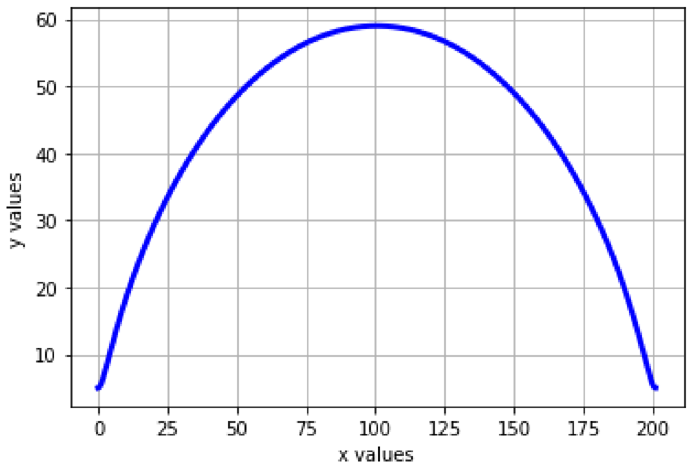


Рис. 1: Вбудовані методи креслення

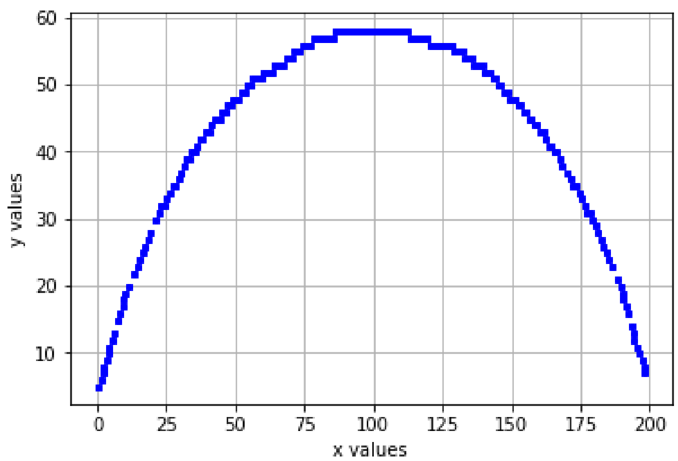


Рис. 2: Алгоритм Брехенхейма

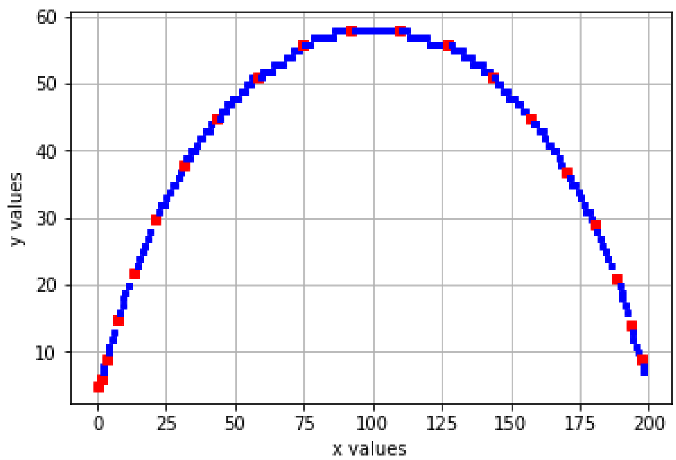


Рис. 3: Алгоритм Брехенхейма (мітки)

Скріншоти кривої у полярному виді (лемініската Бернуллі)

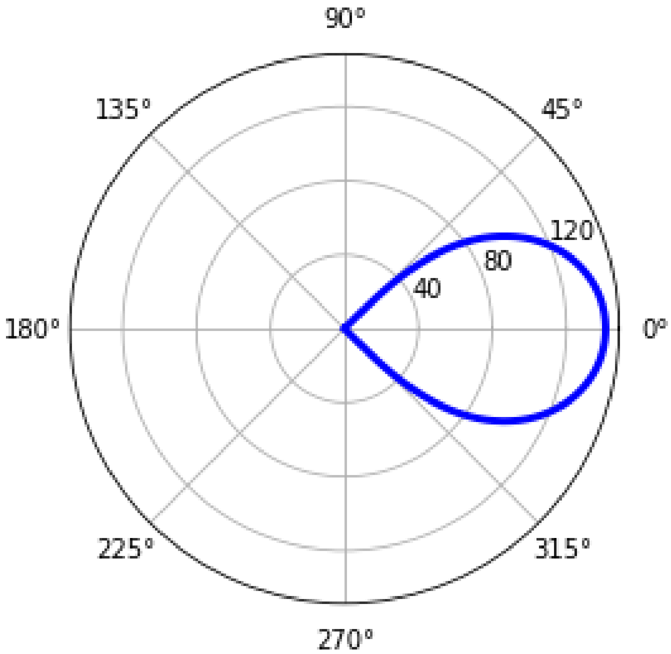


Рис. 4: Вбудовані методи креслення

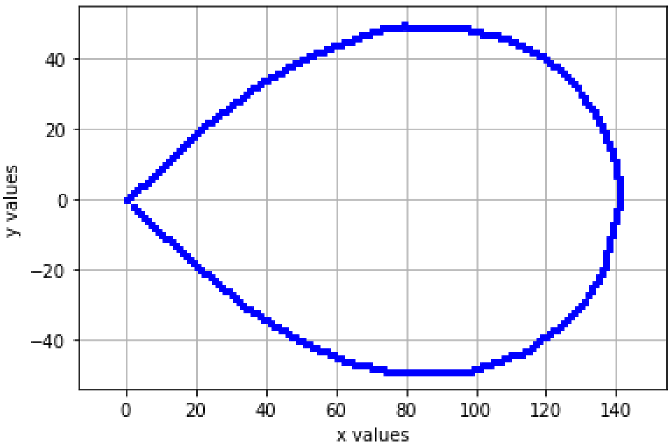


Рис. 5: Алгоритм Брехенхейма

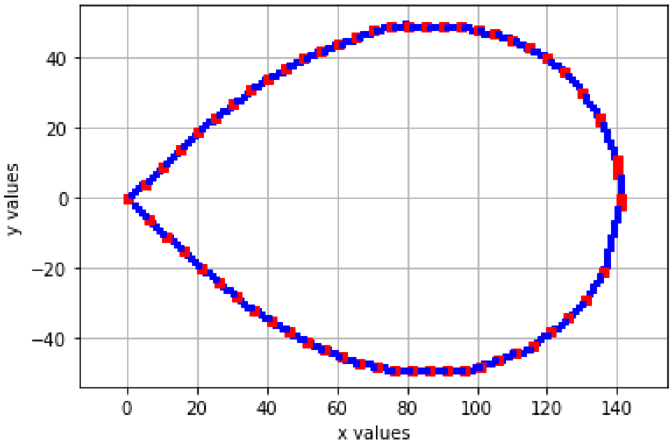


Рис. 6: Алгоритм Брехенхейма (мітки)

Висновки

У лабораторному практикумі я навчився будувати криві лінії як сукупність відрізків, побудованих за допомогою алгоритму Брезенхема. Здобув практичні навички кодування алгоритму на мові python. Намалював графіки функцій, заданих рівняннями в параметричному вигляді та рівняннями у полярних координатах. Розглянув декілька варіантів побудови прямих в залежності від заданих початкових та кінцевих точок, розташування у різних квадрантах.

Контрольні питання

1. *Для чого використовують алгоритм Брезенхейма?*

   Як і алгоритм цифрового диференціального аналізатора, алгоритм Брезенхейма теж використовують для пошуку оптимальних растрових координат для представлення відрізку між двома точками.

2. *Опишіть технологію побудови відрізка прямої по алгоритму Брезенхейма.*

   Фіксуємо одну з координат, по іншій рухаємося. Задаємо певне від'ємне початкове значення помилки. Позначаємо точки поточної зафіксованої прямої на кординатній сітці доти, доки значення помилки не стане додатнім. А тоді корегуємо e (наприклад, відніманням одиниці), і одразу збільшуємо значення зафіксованої координати.

3. *Який вид має рівняння функції у параметричному вигляді та у полярних координатах?*

   У загальному випадку рівняння функції у параметричному вигляді має вид:

   $$\begin{cases} x = x(t) \\ y = y(t) \end{cases}, \forall t \in \mathbb{R}$$

   У полярних же координатах рівняння виглядає так:

   $$\rho = \rho(\varphi),\ \varphi \in (0, 2\pi),\ \rho \geqslant 0$$

4. *Як перетворити рівняння у полярних координатах до рівняння у параметричному вигляді та навпаки?*

   (а) полярні координати $\rightarrow$ параметричний вигляд:
   застосувавши для заданого рівняння $\rho(\varphi)$ нищенаведені заміни, якраз і отримаємо систему у параметричному виді:

   $$\begin{cases} x = \rho\cos\varphi = \rho(\varphi)\cos\varphi = x(\varphi) \\ y = \rho\sin\varphi = \rho(\varphi)\sin\varphi = y(\varphi) \end{cases}, \ \varphi \in (0, 2\pi)$$

   (б) параметричний вигляд $\rightarrow$ полярні координати:
   із заданого у параметричному вигляді системи $x = x(\varphi),\ y = y(\varphi)$ підставляємо значення x та y у рівняння $\rho^2 = x^2 + y^2$. Таким чином:

   $$\rho^2 = x^2 + y^2 = x^2(\varphi) + y^2(\varphi) \Rightarrow \rho = \rho(\varphi)$$

5. *Як виконати масштабування при побудові графічних об'єктів?*

   Перш за все, слід віднайти поточні граничні значення вісей:

```
left, right = plt.xlim()
down, up = plt.ylim()
```

   Надалі, скажімо, за допомогою наведених нижче вбудованих функцій масштабувати графік на деякий коефіцієнт `alpha`:

```
plt.xlim(left*alpha, right*alpha)
plt.ylim(down*alpha, up*alpha)
```