

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA



GIẢI TÍCH 2 (MT1005)

Báo cáo bài tập lớn

Tổng Riemann trong tích phân kép Lý thuyết, ứng dụng và lập trình

Giáo viên hướng dẫn: Trần Ngọc Diễm
Sinh viên thực hiện: Nhóm BTL L07_19
Nguyễn Phúc Nhân - 2312438
Nguyễn Thiện Toàn - 2313493
Nguyễn Thiên Văn - 2313862
Nguyễn Tô Quốc Việt - 2313898
Phạm Đình Dược - 2310761
Vinh Tài - 2313017

THÀNH PHỐ HỒ CHÍ MINH, THÁNG 4 2024



Danh sách thành viên

| STT | Họ và tên | Mã số sinh viên | Ghi chú |
|-----|---------------------|-----------------|----------------|
| 1 | Nguyễn Phúc Nhân | 2312438 | |
| 2 | Nguyễn Thiện Toàn | 2313493 | |
| 3 | Nguyễn Thiên Văn | 2313862 | |
| 4 | Nguyễn Tô Quốc Việt | 2313898 | |
| 5 | Phạm Đình Dược | 2310761 | |
| 6 | Vĩnh Tài | 2313017 | Không tham gia |

Yêu cầu

- 1) Giới thiệu tổng Riemann của hàm số $f(x, y)$ trên hình chữ nhật $R = [a, b] \times [c, d]$.
- 2) Viết một đoạn code tính tổng Riemann trung tâm cho hàm số $f(x, y)$ trên miền hình chữ nhật R .

Input: $f(x, y)$, a , b , c , d để xác định R , m , n là số đoạn chia của $[a, b]$ và $[c, d]$.

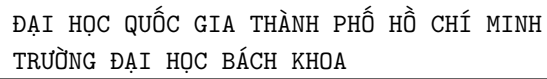
Output: Tổng Riemann.

- 3) Viết một đoạn code tính tổng Riemann cho $f(x, y)$ là dạng bảng số.

Input: $f(x, y)$ dạng bảng số.

Output: Tổng Riemann.

Yêu cầu: Kiểm soát dữ liệu có dùng được tổng trung tâm hay không, nếu được, output sẽ là tổng trung tâm, ngược lại, output có thể dùng cách chọn điểm trên hình chữ nhật con gần điểm $(0, 0)$ nhất.



This image shows a full page of white paper with horizontal dotted lines. The lines are evenly spaced and run across the width of the page, providing a guide for handwriting practice. There are no margins, text, or other markings on the page.



Mục lục

| | | |
|----------|--|-----------|
| 1 | Giới thiệu tổng Reimann trong tích phân kép | 4 |
| 1.1 | Ví dụ | 6 |
| 2 | Công thức và phân tích thuật toán để lập trình | 7 |
| 2.1 | Phân tích thuật toán chia miền để tính tích phân | 7 |
| 2.2 | Thuật toán kiểm soát dữ liệu dạng bảng | 8 |
| 2.3 | Mã giả biểu diễn thiết kế thuật toán | 9 |
| 3 | Dùng Python để tính tổng Riemann, trực quan hoá dữ liệu và thiết lập giao diện người dùng | 11 |
| 3.1 | Giới thiệu về Python | 11 |
| 3.1.1 | Thư viện NumPy xử lý toán học | 12 |
| 3.1.2 | Thư viện Tkinter biểu diễn đồ hoạ | 13 |
| 3.1.3 | Module trong Python | 14 |
| 3.2 | Hiện thực thuật toán | 15 |
| 3.2.1 | Hàm tính tổng Reimann trung tâm | 15 |
| 3.2.2 | Xử lý tính tổng Reimann trong miền hình chữ nhật từ hàm đã cho trước | 16 |
| 3.2.3 | Lấy dữ liệu dạng bảng | 18 |
| 3.2.4 | Xử lý dữ liệu và thực hiện tính toán dạng bảng | 18 |
| 4 | Kết luận | 22 |
| | Tài liệu tham khảo | 22 |
| 5 | Tổng kết | 23 |

1 Giới thiệu tổng Reimann trong tích phân kép

Tổng Riemann là một phương pháp quan trọng trong tính toán và phân tích hàm số hai biến trên các miền xác định. Giả sử $z = f(x, y)$ là một hàm liên tục được xác định trên một miền hình chữ nhật

$$R = [a, b] \times [c, d] = \{(x, y) : a \leq x \leq b, c \leq y \leq d\}$$

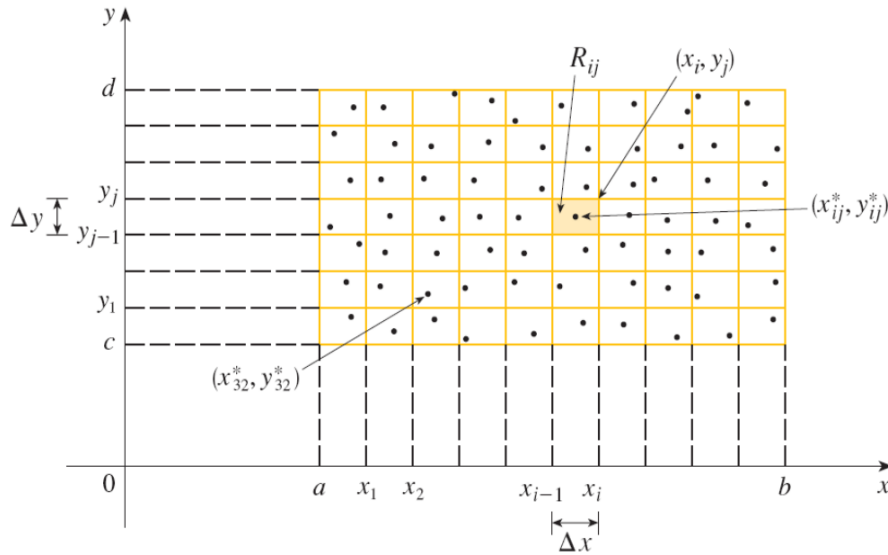
và giả sử rằng $f(x, y) \geq 0$ trên R . Cho

$$a = x_0 < x_1 < \dots < x_{i-1} < x_i < \dots < x_m = b$$

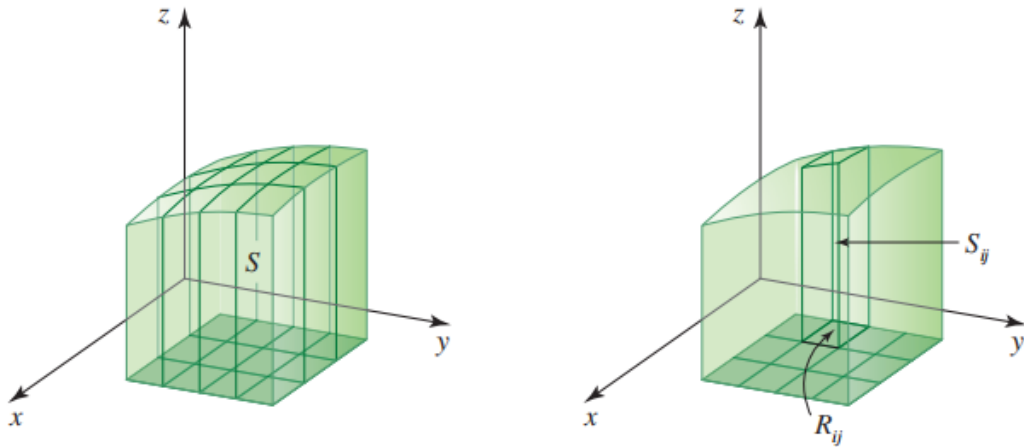
là một phân hoạch đều của khoảng $[a, b]$ thành m khoảng con có độ dài $\Delta x = \frac{b-a}{m}$ và cho

$$c = y_0 < y_1 < \dots < y_{j-1} < y_j < \dots < y_n = d$$

là một phân hoạch đều của khoảng $[c, d]$ thành n khoảng con có độ dài $\Delta y = \frac{d-c}{n}$. Khi đó, tấm lưới được tạo bởi các đoạn thẳng đứng x_i với $0 \leq i \leq m$ và các đoạn thẳng ngang y_j với $0 \leq j \leq n$ sẽ phân hoạch thành các hình chữ nhật R_{ij} ($0 \leq i \leq m, 0 \leq j \leq n$) với diện tích mỗi chữ nhật con là $\Delta A = \Delta x \Delta y$ như hình bên dưới.

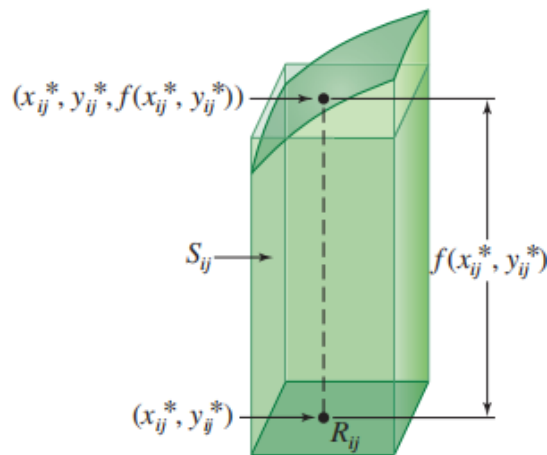


Lúc này, phần phân vùng sẽ chia đồ thị $z = f(x, y)$ thành các khối S_{ij} ($0 \leq i \leq m, 0 \leq j \leq n$) như hình minh họa bên dưới.



Ta xét (x_{ij}^*, y_{ij}^*) là các điểm trong vùng R_{ij} . Khi đó, $f(x_{ij}^*, y_{ij}^*)$ là chiều cao của khối hộp có đáy R_{ij} và thể tích của khối hộp là

$$f(x_{ij}^*, y_{ij}^*)\Delta A.$$



Hình trên cho ta thể tích xấp xỉ của khối S_{ij} . Chính vì thế, thể tích của cả khối S là tổng các thể tích của $N = nm$ khối hộp có đáy R_{ij} , ta có công thức:

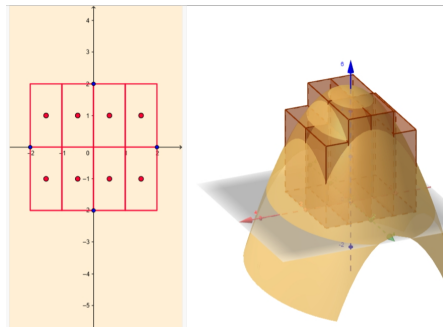
$$V \approx \sum_{i=0}^m \sum_{j=0}^n f(x_{ij}^*, y_{ij}^*)\Delta A$$

Vậy tổng Riemann trên hình chữ nhật là một phương pháp số học quan trọng, thường được dùng để ước lượng giá trị tích phân trong giải tích, đặc biệt là trong việc tìm diện tích dưới đường cong, tính thể tích và các vấn đề liên quan đến tích phân.

1.1 Ví dụ

Xét hàm số $z = f(x, y) = 5 - \frac{x^2}{2} - \frac{y^2}{2}$ trên hình chữ nhật $R = [-2, 2] \times [-2, 2]$. Để tính tổng Riemann của $f(x, y)$ trên R , chúng ta chia R thành $m \times n$ hình chữ nhật con.

Giả sử chúng ta chia R thành 4×2 hình chữ nhật con. Kích thước của mỗi hình chữ nhật con là $\Delta x = \frac{2-(-2)}{4} = 1$ và $\Delta y = \frac{2-(-2)}{2} = 2$. Ta sẽ lấy điểm (x_{ij}^*, y_{ij}^*) với $(0 \leq i \leq 4, 0 \leq j \leq 2)$ là các điểm trung tâm của từng hình chữ nhật con. Khi đó, ta tính được các giá trị tương



Hình 1: Ví dụ phân tích tổng Reimann trong hình chữ nhật $[-2, 2] \times [-2, 2]$

ứng như sau:

- $f(-1.5, -1) = \frac{29}{8}$
- $f(-1.5, 1) = \frac{29}{8}$
- $f(-0.5, -1) = \frac{37}{8}$
- $f(-0.5, 1) = \frac{37}{8}$
- $f(0.5, -1) = \frac{37}{8}$
- $f(0.5, 1) = \frac{37}{8}$
- $f(1.5, -1) = \frac{29}{8}$
- $f(1.5, 1) = \frac{29}{8}$

Vậy tổng Riemann của $f(x, y)$ trên R là:

$$\sum_{i=1}^2 \sum_{j=1}^2 f(x_i, y_j) \cdot \Delta A_{ij} = \left(\frac{29}{8} + \frac{29}{8} + \frac{29}{8} + \frac{29}{8} + \frac{37}{8} + \frac{37}{8} + \frac{37}{8} + \frac{37}{8} \right) \cdot 1 \cdot 2 = 66$$

2 Công thức và phân tích thuật toán để lập trình

Như đã giới thiệu, Tổng Riemann là một phương pháp xấp xỉ để tính toán diện tích hoặc tổng tích phân của một hàm trên một miền xác định trong không gian hai hoặc nhiều chiều. Quy trình này bắt đầu bằng cách phân chia miền tích phân thành các hình chữ nhật hoặc các hình dạng khác nhau, sau đó tính toán giá trị của hàm tại một điểm thể hiện cho từng hình dạng.

Chúng ta thường sử dụng biểu thức sau để tính Tổng Riemann:

$$S = \sum_{i=1}^n f(x_i^*, y_i^*) \cdot \Delta A_i$$

Trong đó:

- $f(x_i^*, y_i^*)$ là giá trị của hàm tại một điểm (x_i^*, y_i^*) trong mỗi hình dạng của phân chia.
- ΔA_i là diện tích hoặc thể tích của hình dạng thứ i .
- n là số lượng hình dạng trong phân chia.

Khi kích thước của các hình dạng tiến dần đến không, tổng Riemann tiến dần đến giá trị của tích phân của hàm trên miền tích phân.

2.1 Phân tích thuật toán chia miền để tính tích phân

1. Xác định Miền Tích Phân:

- Xác định miền tích phân trên mặt phẳng hai chiều hoặc miền tích phân trong không gian ba chiều.
- Chia miền tích phân thành các phần nhỏ hơn, ví dụ: hình chữ nhật, hình vuông, ...

2. Chọn Điểm Biểu Diễn:

- Đối với mỗi phần nhỏ trong phân chia, chọn một điểm biểu diễn.
- Điểm này thường được chọn ở góc trên bên trái của hình dạng, nhưng cũng có thể chọn ở các vị trí khác, chẳng hạn như điểm trung tâm hoặc góc dưới bên phải. Trong bài toán này, chúng ta sẽ chọn điểm biểu diễn là trung tâm

3. Tính Giá Trị Hàm:

- Tại mỗi điểm biểu diễn đã chọn, tính giá trị của hàm.

- Đây là bước quan trọng và thường là bước tốn nhiều thời gian nhất trong quá trình tính toán. Đối với hàm phức tạp, có thể cần sử dụng các phương pháp xấp xỉ hoặc số học để tính giá trị của hàm.

4. Tính Diện Tích:

- Tính diện tích của mỗi phần nhỏ trong phân chia.
- Đối với hình chữ nhật, diện tích được tính bằng cách nhân độ dài và chiều rộng.

5. Tính Tổng Riemann:

- Nhân giá trị của hàm tại mỗi điểm biểu diễn với diện tích (hoặc thể tích) tương ứng của phần nhỏ đó.
- Cộng tất cả các kết quả lại để tính tổng Riemann.

7. Kiểm Tra Kết Quả và Điều Chính:

- Kiểm tra kết quả của tính toán và đảm bảo rằng nó đáng tin cậy và chính xác.

2.2 Thuật toán kiểm soát dữ liệu dạng bảng

Dựa trên một tập hợp dữ liệu đầu vào, được biểu diễn dưới dạng từ điển, thuật toán duyệt qua các điểm trên một lưới hai chiều, sử dụng tổng Riemann trung tâm để tính toán diện tích dưới đồ thị của hàm số.

Thuật toán bắt đầu từ điểm có tọa độ x nhỏ nhất và y tương ứng, sau đó duyệt qua các ô hình chữ nhật trên lưới, kiểm tra xem các giá trị hàm số có tồn tại tại các điểm đó hay không trong từ điển dữ liệu. Nếu có ít nhất một điểm không tồn tại, thuật toán sẽ báo hiệu không thể sử dụng tổng Riemann trung tâm.

Nếu tất cả các giá trị hàm số tồn tại, thuật toán tính tổng các giá trị này.

Trong trường hợp, miền được phân hoạch theo hình chữ nhật nhưng không tìm thấy được điểm trung tâm. Ta sẽ lấy điểm gần với gốc tọa độ $(0,0)$, với ý tưởng như sau:

- Ta xác định hình chữ nhật sẽ có 2 tọa độ x và 2 tọa độ y , hàm dùng để xác định khoảng cách 1 tọa độ với điểm 0 là hàm trị tuyệt đối (`abs()` hoặc `| |`). Vậy điểm gần gốc tọa độ là điểm có trị tuyệt đối của tọa độ x và y nhỏ nhất.
- ta sẽ chọn tọa độ x dựa theo $\min(|x_1|, |x_2|)$ và tọa độ y tương tự

Ta có mã giả như sau:

- choose(x_1, x_2, y_1, y_2):
- x, y
- if $\text{abs}(x_1) > \text{abs}(x_2)$ $x = x_2$ else $x = x_1$
endif
- if $\text{abs}(y_1) > \text{abs}(y_2)$ $y = y_2$ else $y = y_1$
endif
- return x, y

2.3 Mã giả biểu diễn thiết kế thuật toán

Dưới đây là thiết kế thuật toán cho tổng Riemann với điểm biểu diễn trung tâm từ miền hình chữ nhật và các phân hoạch theo x, y

FUNCTION RiemannSum(f, a, b, c, d, m, n):

- **Input:** Hàm hai biến $f(x, y)$, biên độ (a, b) và (c, d) của miền tích phân theo hai biến x và y , số lượng cột m và hàng n để chia mỗi biến.
- **Output:** Tổng Riemann của hàm $f(x, y)$ trên miền tích phân.
- $\Delta x \leftarrow \frac{b-a}{m}$
- $\Delta y \leftarrow \frac{d-c}{n}$
- **sum** $\leftarrow 0$
- FOR i FROM 0 TO $m - 1$ DO:
 - FOR j FROM 0 TO $n - 1$ DO:
 - * $x \leftarrow a + (i + 0.5) \times \Delta x$
 - * $y \leftarrow c + (j + 0.5) \times \Delta y$
 - * **sum** \leftarrow **sum** $+ f(x, y) \times \Delta x \times \Delta y$
- RETURN **sum**

Khi chọn điểm biểu diễn ở góc trên bên phải của mỗi phần nhỏ:

- Cập nhật x và y để $x \leftarrow a + (i + 1) \times \Delta x$ và $y \leftarrow c + (j + 1) \times \Delta y$.

Khi chọn điểm biểu diễn ở góc dưới bên trái của mỗi phần nhỏ:

- Cập nhật x và y để $x \leftarrow a + i \times \Delta x$ và $y \leftarrow c + j \times \Delta y$.

Đối với thuật toán tính tổng Reimann từ dữ liệu nhập dạng bảng số, sau khi lấy dữ liệu và lưu trữ, ta thiết lập hàm kiểm soát dữ liệu được thiết kế theo mã giả sau:

FUNCTION `checkRectangle`($start_x, start_y, \Delta x, \Delta y, f_val, num_x, num_y$):

- **Input:**

- $start_x, start_y$: điểm x và y bắt đầu xét
- $\Delta x, \Delta y$: là khoảng cách giữa các tọa độ trên trục Ox và Oy
- f_val : là "tờ điển" lưu trữ giá trị hàm $f(x,y)$ tại tọa độ (x,y)
- num_x, num_y : là số lượng đỉnh xét theo trục Ox và Oy

- **Output:** Tổng Riemann của hàm $f(x,y)$ trên miền tích phân dựa theo bảng số (nếu tính được). Nếu không thì trả về "False"

- $sum \leftarrow 0$

- FOR i FROM 0 TO num_x DO:

- FOR j FROM 0 TO num_y DO:

- * $x \leftarrow start_x + 2i \times \Delta x$
- * $y \leftarrow start_y + 2j \times \Delta y$
- * if (x not in f_val) or (y not in $f_val[x]$): return false

- $start_x \leftarrow start_x + \Delta x$

- $start_y \leftarrow start_y + \Delta y$

- FOR i FROM 0 TO $num_x - 1$ DO:

- FOR j FROM 0 TO $num_y - 1$ DO:

- * $x \leftarrow start_x + 2i \times \Delta x$
- * $y \leftarrow start_y + 2j \times \Delta y$
- * if (x not in f_val) or (y not in $f_val[x]$): return false
- * else $sum \leftarrow sum + f_val[x][y]$

- RETURN sum

3 Dùng Python để tính tổng Riemann, trực quan hoá dữ liệu và thiết lập giao diện người dùng

3.1 Giới thiệu về Python

Python là một ngôn ngữ lập trình tạo ra bởi Guido van Rossum và được phát hành năm 1991

Python là một ngôn ngữ mạnh và được ứng dụng rộng rãi:

- Phát triển web/app (đặc biệt trong xử lý ngoại biên)
- Công nghệ phần mềm
- Khoa học dữ liệu và Trí tuệ nhân tạo
- Toán học và xử lý số liệu
- Thiết kế và quản lý hệ thống, hệ điều hành

Cú pháp của Python so với các ngôn ngữ khác:

- Python là ngôn ngữ bậc cao được thiết kế gần với ngôn ngữ tự nhiên giúp dễ dàng hiện thực các thuật toán
- Python sử dụng các dòng mới để hoàn thành một lệnh thay vì dùng dấu ';' hoặc '()'
- Python giúp cho lập trình viên có thể viết code ngắn hơn các ngôn ngữ khác

Làm sao để sử dụng Python

Thông thường, một vài loại PC và Mac đã có cài đặt sẵn Python, hoặc có thể tải [tại đây](#). Để kiểm tra xem Python đã được cài đặt trên Window PC hay chưa, bạn có thể tìm kiếm trong Start hoặc gõ lệnh trong Command Line (cmd.exe):

```
python --version
```

Hàm trong Python dùng để định nghĩa một chức năng bằng một khối lệnh được biểu diễn như sau:

```
def <Ten_ham>(<Danh_sach_cac_tham_so>):  
    Khoi_lenh
```



Vòng lặp **for** trong Python cho phép thực hiện một khối mã nhiều lần

```
for <biến> in <danh sách hoặc dãy số>:  
    Khởi lệnh
```

Ngoài ra Python còn hỗ trợ các kiểu dữ liệu mạnh (list, dictionary, tuple, int, float, string ...) và các câu lệnh hỗ trợ như (format(), chr(), append(), join(), transpose(), ...) và còn nhiều hơn nữa các hàm, các lệnh xử lý rất mạnh của Python ([tại đây](#))

3.1.1 Thư viện NumPy xử lý toán học

NumPy là một thư viện toán học và mảng số trong Python (Numerical Python). NumPy hỗ trợ các phương thức, hàm hỗ trợ trong Đại số tuyến tính, biến đổi Fourier, và ma trận. Thư viện NumPy là một dự án mã nguồn mở được tạo ra năm 2005 bởi Travis Oliphant và bạn có thể dùng nó miễn phí.

Để cài đặt NumPy, bạn có thể sử dụng lệnh pip của Python trong Command Line (đối với Windows) và terminal (Đối với Macs và các hệ điều hành Unix):

```
pip install numpy
```

Sau khi cài đặt xong thư viện, bạn có thể đưa thư viện vào chương trình bằng từ khoá import:

```
import numpy
```

Có thể sử dụng numpy bằng tên thay thế np

```
import numpy as np
```

Ta tạo mảng số 1, 2 chiều hoặc nhiều chiều bằng NumPy:

```
# Python program for create arrays  
import numpy as np
```

```
# Creating a array 1D  
arr = np.array([1, 2, 3])  
print("Array 1D: \n",arr)
```

```
# Creating a array 2D (Matrix)
```

```
arr = np.array([[1, 2, 3],
                [4, 5, 6]])
print("Array 2D: \n",arr)

# Creating an array from tuple
arr = np.array((1, 3, 2))
print("\nArray created using tuple:\n", arr)
```

Kết quả:

Array 1D:

[1 2 3]

Array 2D:

[[1 2 3]

[4 5 6]]

Array created using tuple:

[1 3 2]

Tìm hiểu thêm về NumPy [tại đây](#)

NumPy là một thư viện mạnh ứng dụng rộng rãi trong các lĩnh vực đặc biệt của Toán Ứng Dụng và Toán - tin như Khoa học Dữ liệu (Data Science), Học Máy (Machine Learning), Học Sâu (Deep Learning), Trí tuệ nhân tạo (Artificial Intelligence), Phân tích dữ liệu (Data Analyst), Thị giác máy tính (Computer Vision), ...

3.1.2 Thư viện Tkinter biểu diễn đồ hoạ

Tkinter được định nghĩa là Giao diện đồ hoạ người dùng trong Python (Python GUI), nó cung cấp các bộ công cụ (tools) và các thành phần đồ hoạ (widgets) để tạo ra ứng dụng máy tính với giao diện đồ hoạ. Thư viện này được cài đặt sẵn cùng với Python, nhằm giúp lập trình viên dễ dàng xây dựng đồ hoạ mà không cần thư viện bổ sung.

Những ứng dụng của Tkinter trong Python:

- **Tạo cửa sổ và hộp thoại:** Tkinter có thể được dùng để tạo cửa sổ hộp thoại. Thông qua đó có thể hiển thị thông tin, thu thập đầu vào hay trình bày tùy chọn cho người dùng

- **Xây dựng giao diện người dùng (GUI) cho máy tính:** Tkinter có thể tạo giao diện để dễ dàng tương tác bao gồm menu, các nút, ...
- Tạo các thành phần đồ họa (widget) tùy chỉnh: Tkinter có tích hợp các widget đa dạng, tuy nhiên bạn có thể tự định nghĩa các widget của riêng mình

Đây là một ví dụ để tạo một chương trình đồ họa:

```
from tkinter import *  
root = Tk()  
root.title("Hello World")  
root.geometry('350x200')  
root.mainloop()
```

Kết quả chương trình



Hình 2: Chương trình đồ họa Hello World

Tóm lại, Tkinter là một công cụ hữu ích để tạo nhiều giao diện người dùng đồ họa, bao gồm cửa sổ, hộp thoại và tiện ích tùy chỉnh. Nó đặc biệt phù hợp để xây dựng các ứng dụng máy tính để bàn và thêm GUI vào các chương trình dòng lệnh.

3.1.3 Module trong Python

Module có thể coi như là một thư viện mà ta tự định nghĩa, nó chứa các hàm mà ta muốn đưa vào ứng dụng của mình

Cách tạo module đơn giản là định nghĩa các hàm sau đó lưu tệp bằng đuôi .py

Ví dụ chúng ta tạo 1 file có tên mymodule.py:

```
def sayHi(name):  
    print("Hello " + name)
```

Sau đó ở chương trình chính, ta gọi:



```
import mymodule  
mymodule.sayHi("thay Hiep")
```

hoặc

```
from module import sayHi  
sayHi("co Diem")
```

Như vậy, chương trình sẽ cho ra kết quả:

```
Hello co Diem
```

3.2 Hiện thực thuật toán

Như các định nghĩa và thuật toán đã nêu trước đó, ta hiện thực các thuật toán trong các hàm. Tuy nhiên bài báo cáo này sẽ không tập trung vào các câu lệnh và thuật toán thiết lập đồ hoạ, vì vậy các lệnh `print()` sẽ thay thế cho các hộp thoại thông báo của phần mềm.

3.2.1 Hàm tính tổng Reimann trung tâm

Thiết lập hàm tính tổng Reimann định nghĩa trong module `ReimannSumoff.py` nhận các tham số đầu vào:

- f : Hàm hai biến $f(x, y)$ được tính toán trên miền tích phân.
- a, b : Biên độ của miền tích phân trên trục x , trong đó a là giới hạn dưới và b là giới hạn trên.
- c, d : Biên độ của miền tích phân trên trục y , trong đó c là giới hạn dưới và d là giới hạn trên.
- m : Số lượng cột để chia miền tích phân theo trục x .
- n : Số lượng hàng để chia miền tích phân theo trục y .

```
# Function to calculate Riemann Sum  
def riemann_sum(f, a, b, c, d, m, n):  
    dx = (b - a) / m  
    dy = (d - c) / n
```

```
riemann_sum = 0
for i in range(m):
    for j in range(n):
        x = a + (i + 0.5) * dx
        y = c + (j + 0.5) * dy
        riemann_sum += f(x, y) * dx * dy
return riemann_sum
```

Bên cạnh đó, module ReimannSumofF.py cũng chứa các hàm thực hiện quá trình nhập dữ liệu cho chương trình và chuẩn bị dữ liệu cho mô phỏng tổng Reimann.

3.2.2 Xử lý tính tổng Reimann trong miền hình chữ nhật từ hàm đã cho trước

Sau khi khởi động màn hình đồ họa, ta nhấn chuột vào nút tính, chương trình sẽ gọi tới hàm `calculate_riemann_sum()`



Hình 3: Khởi động giao diện đồ họa cho phần mềm

```
def calculate_riemann_sum():
    try:
        f = input_function()
        a, b, c, d = input_intervals()
        m, n = input_partition()
        result = riemann_sum(f, a, b, c, d, m, n)
        print(result)
```

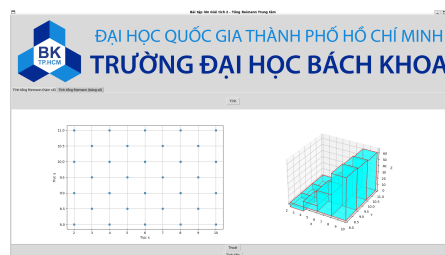
```
t_data, dx, dy, f_val = prepare_data(f,a,b,c,d,m,n)
fig1 = simulate2D(t_data)
simulate3D(a, c, dx/2, dy/2, f_val, m, n, fig1)
except Exception as e:
    print("Error: ", f'{e}')
```

Hàm này thực hiện chức năng lấy đầu vào bằng các hàm input định nghĩa trong module ReimannSumoff.py, sau đó dùng các lệnh mô phỏng trong module simulate.py để thể hiện các miền phân hoạch. Bên cạnh đó, hàm dùng kỹ thuật 'throw exception' để đảm bảo dữ liệu người dùng nhập vào là hợp hệ với ngôn ngữ và cách xử lý của phần mềm, ví dụ: nhập hàm $f(x,y)$ là $x^2 + 2y$ là sai mà phải nhập là $x**2+2*y$.

Như vậy chương trình tính tổng Riemann trên miền hình chữ nhật với hàm số cho trước đã được hiện thực xong. Sau đây là ví dụ khi chạy chương trình với dữ liệu đầu vào là hàm $f(x,y) = x**2 - 2*y + 1$, trong miền (x,y) là $[2 \ 10] \times [8 \ 11]$ với khoảng chia lần lượt là 4 và 3



Hình 4: Kết quả khi chạy chương trình từ dữ liệu như trên



Hình 5: Mô phỏng miền chia và các khối trong 3D

3.2.3 Lấy dữ liệu dạng bảng

Để lấy dữ liệu từ người dùng, ta hiện thực hàm `get_user_input()` trong module `inputtest`, sau đó lưu vào file `data.csv`. Ngoài ra chương trình có thiết lập hàm tạo bộ dữ liệu là miền hình chữ nhật được phân hoạch có thể tính tổng Reimann trung tâm và hàm tạo bộ dữ liệu ngẫu nhiên để kiểm tra được lưu lần lượt trong các module *rectangleTest.py*, *randomtest.py*

3.2.4 Xử lý dữ liệu và thực hiện tính toán dạng bảng

Sau khi đã lấy dữ liệu ta tiến hành xử lý và lưu vào *từ điển* (một kiểu dữ liệu trong python) bằng hàm `read_data_from_csv()` trong module *ReimannSumfromTable.py*

Cũng trong module này, ta định nghĩa hàm `checkRectangle()` tương tự thuật toán đã nêu:

```
def checkRectangle(start_x, start_y, \
    del_x, del_y, f_val, num_x, num_y):
    sum = 0
    for i in range(num_x):
        for j in range(num_y):
            x = start_x + i * del_x * 2
            y = start_y + j * del_y * 2
            if x not in f_val or y not in f_val[x]:
                return False
        start_x = start_x + del_x
        start_y = start_y + del_y
    for i in range(num_x - 1):
        for j in range(num_y - 1):
            x = start_x + i * del_x * 2
            y = start_y + j * del_y * 2
            if x not in f_val or y not in f_val[x]:
                return False
        else:
            sum = sum + f_val[x][y]

    return sum
```

Sau đó, ở chương trình chính, ta cũng thiết lập nút lệnh trở tới hàm xử lý tính tổng Reimann trung tâm từ bảng số gọi tên là `calculate_riemann_sum_from_table()`, đầu tiên ta thực hiện việc nhập và kiểm tra $\Delta x, \Delta y$ có bằng nhau tại mọi điểm hay không thông qua cấu trúc lệnh `diff` trong thư viện Numpy, sau đó gọi hàm `checkRectangle()` để kiểm tra xem có tính được tổng Reimann hay không, nếu có thì cho ra kết quả và mô phỏng 2D, 3D, nếu không thì cho biết không tính được và biểu diễn các điểm trên mặt phẳng tọa độ.

```
data, func_val = read_data_from_csv()
arr_x = sorted(set(np.array([x for x, _, _ in data])))
arr_y = sorted(set(np.array([y for _, y, _ in data])))
d_x = np.diff(arr_x).min()
d_y = np.diff(arr_y).min()
ReimannSum = False
if d_x and d_y:
    ReimannSum = checkRectangle(arr_x[0], arr_y[0], d_x, \
                                d_y, func_val, len(arr_x)//2 + 1, len(arr_y)//2 + 1)
    if ReimannSum:
        messagebox.showinfo("Result", f"Reimann Sum:\n\
{round(ReimannSum, 4)}")
        fig2 = simulate2D(data)
        simulate3D(arr_x[0], arr_y[0], d_x, d_y, func_val, \
                    len(arr_x)//2, len(arr_y)//2, fig2)
    else:
        messagebox.showerror("Error", "Can't use Reimann Sum")
        fig2 = simulate2D(data)
```

Sau đây là ví dụ khi chạy chương trình, từ bộ dữ liệu hình chữ nhật được phân hoạch đúng và bộ dữ liệu ngẫu nhiên



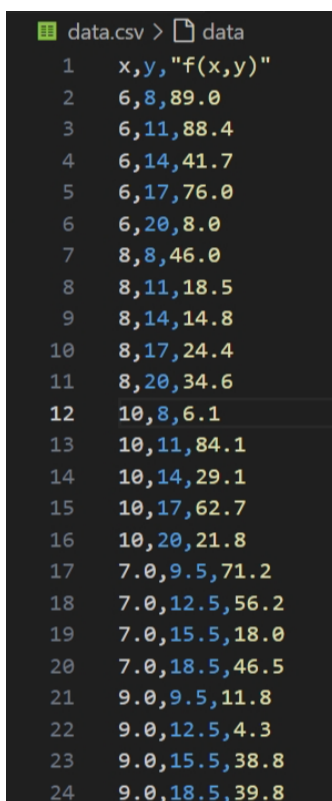
Hình 6: Khởi động chương trình

Trong chương trình chính, có 3 nút để tạo và lấy bộ dữ liệu

- **Lấy dữ liệu:** Nút này để lấy dữ liệu từ file *data.csv*, Nếu muốn lấy dữ liệu từ bàn phím thì trước tiên phải chạy file *inputtest.py* để nhập dữ liệu
- **Tạo dữ liệu:** Nút này để tạo bộ dữ liệu hình chữ nhật phân hoạch đúng
- **Tạo dữ liệu ngẫu nhiên:** Tạo bộ dữ liệu ngẫu nhiên

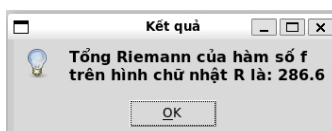
Lưu ý: sau khi tạo dữ liệu xong, ta mới lấy dữ liệu

Sau đây là kết quả từ quá trình tạo dữ liệu hình chữ nhật:

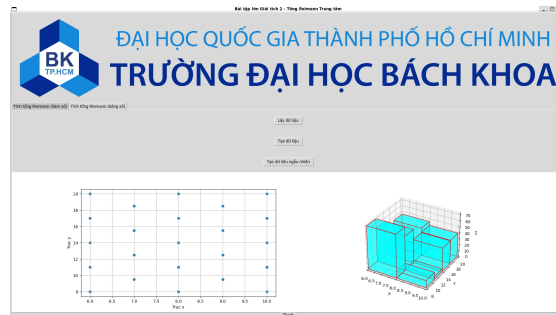


```
data.csv > data
1  x,y,"f(x,y)"
2  6,8,89.0
3  6,11,88.4
4  6,14,41.7
5  6,17,76.0
6  6,20,8.0
7  8,8,46.0
8  8,11,18.5
9  8,14,14.8
10 8,17,24.4
11 8,20,34.6
12 10,8,6.1
13 10,11,84.1
14 10,14,29.1
15 10,17,62.7
16 10,20,21.8
17 7.0,9.5,71.2
18 7.0,12.5,56.2
19 7.0,15.5,18.0
20 7.0,18.5,46.5
21 9.0,9.5,11.8
22 9.0,12.5,4.3
23 9.0,15.5,38.8
24 9.0,18.5,39.8
```

Hình 7: Dữ liệu từ file csv



Hình 8: Kết quả tính được Reimann Sum

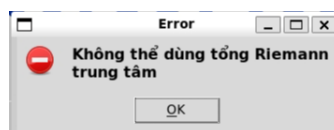


Hình 9: Mô phỏng từ bộ dữ liệu

Sau đây là kết quả từ quá trình tạo dữ liệu ngẫu nhiên:

| | x,y,"f(x,y)" |
|----|--------------|
| 1 | 2.8,4.4,10.6 |
| 2 | 3.2,1.6,5.9 |
| 3 | 1.1,2.1,8.7 |
| 4 | 2.5,4.9,10.3 |
| 5 | 2.2,3.5,6.9 |
| 6 | 1.6,2.5,10.3 |
| 7 | 4.3,4.2,6.6 |
| 8 | 4.4,4.4,4.6 |
| 9 | 3.9,2.6,10.6 |
| 10 | 4.8,2.3,6.7 |
| 11 | 1.3,2.0,11.7 |
| 12 | 2.6,0.8,12.1 |
| 13 | 3.3,2.9,10.0 |
| 14 | 3.3,0.7,7.2 |
| 15 | 1.3,2.6,8.2 |
| 16 | 1.8,1.6,7.5 |
| 17 | 3.5,3.3,10.2 |
| 18 | 4.3,2.4,7.2 |

Hình 10: Dữ liệu ngẫu nhiên



Hình 11: Kết quả từ bộ dữ liệu



Hình 12: Kết quả mô phỏng

Xem toàn bộ chương trình [tại đây](#)

4 Kết luận

Tổng Riemann trên hình chữ nhật là một công cụ mạnh mẽ và cơ bản trong việc tính toán tích phân kép của hàm số hai biến trên một miền xác định. Phương pháp này chia miền thành các hình chữ nhật nhỏ hơn, giúp xấp xỉ thể tích của khối dưới đồ thị hàm số bằng tổng các thể tích của các khối hộp. Việc chọn các điểm lấy mẫu x_{ij}^*, y_{ij}^* trong mỗi hình chữ nhật con là một bước quan trọng, ảnh hưởng đến độ chính xác của kết quả xấp xỉ. Tổng Riemann không chỉ có ý nghĩa trong lý thuyết mà còn được ứng dụng rộng rãi trong thực tiễn, từ tính toán các bài toán vật lý đến phân tích dữ liệu trong khoa học và kỹ thuật.

Tài liệu tham khảo

- [1] Giáo trình Giải tích 2 - Nguyễn Đình Huy (Chủ biên)
- [2] Soo T. Tan. (2009). Multivariable Calculus. Brooks Cole.
- [3] James Stewart. (2015). Multivariable Calculus (8th ed.). Cengage Learning.
- [4] Kaiser, U. Introduction to Multivariable Calculus. (n.d.).

5 Tổng kết

Nhóm chúng em đã thể hiện sự xuất sắc trong việc hợp tác và nghiên cứu đề tài, đặc biệt là trong việc phát triển sản phẩm code. Mỗi thành viên đã đóng góp không chỉ về kiến thức mà còn về nhiệt huyết và tinh thần trách nhiệm, giúp nhóm vượt qua mọi thử thách và đạt được những kết quả ấn tượng. Sự phối hợp nhịp nhàng và tinh thần làm việc đồng đội đã tạo nên một môi trường làm việc hiệu quả và sáng tạo, nơi mọi ý tưởng đều được lắng nghe và phát triển.

Quá trình nghiên cứu đề tài và phát triển sản phẩm code đã giúp chúng em không chỉ nắm vững hơn về chủ đề mà còn rèn luyện nhiều kỹ năng quan trọng như giao tiếp, quản lý thời gian và giải quyết vấn đề. Chúng em đã học hỏi lẫn nhau, chia sẻ kiến thức và cùng nhau tiến bộ, khẳng định rằng sự đoàn kết và hợp tác luôn là chìa khóa dẫn đến thành công.

Thành quả lớn nhất của chúng em là sản phẩm code hoàn chỉnh, thể hiện sự kết hợp hoàn hảo giữa kỹ thuật và ý tưởng sáng tạo. Sản phẩm này không chỉ là minh chứng cho khả năng chuyên môn của chúng em mà còn là kết quả của một quá trình làm việc chăm chỉ và kiên trì. Mỗi dòng code, mỗi tính năng và mỗi cải tiến đều là thành quả của sự hợp tác và nỗ lực không ngừng của toàn bộ nhóm.

Nhìn lại những gì đã đạt được, chúng em có thể tự hào về những nỗ lực và thành quả của mình. Những kinh nghiệm và kỹ năng mà chúng em đã tích lũy sẽ là nền tảng vững chắc cho các dự án trong tương lai. Với sự cam kết và làm việc nhóm, không có mục tiêu nào là không thể đạt được.