ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA VẬT LÝ - VẬT LÝ KỸ THUẬT
CHUYÊN NGÀNH VẬT LÝ TIN HỌC

——————oOo——————

KHOÁ LUẬN TỐT NGHIỆP

Đề tài:

GIẢI TÍCH SỐ TÍCH PHÂN MONTE CARLO NHIỀU LỚP BẰNG GIEO ĐIỂM QUAN TRỌNG

SVTH: Huỳnh Thị Hạ Vy

CBHD: TS. Nguyễn Chí Linh

TP. HỒ CHÍ MINH - 2019

ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA VẬT LÝ - VẬT LÝ KỸ THUẬT
CHUYÊN NGÀNH VẬT LÝ TIN HỌC

——————oOo——————

KHOÁ LUẬN TỐT NGHIỆP

Đề tài:

GIẢI TÍCH SỐ TÍCH PHÂN MONTE CARLO NHIỀU LỚP BẰNG GIEO ĐIỂM QUAN TRỌNG

SVTH: Huỳnh Thị Hạ Vy

CBHD: TS. Nguyễn Chí Linh

TP. HỒ CHÍ MINH - 2019

Lời cảm ơn

Đầu tiên, tôi xin gửi lời biết ơn đến gia đình đã hỗ trợ và tạo điều kiện tốt nhất cho tôi trong suốt quá trình học tập tại trường Đại học Khoa học Tự nhiên.

Em xin chân thành cảm ơn sâu sắc đến Thầy TS. Nguyễn Chí Linh đã tận tình hướng dẫn và chỉnh sửa khóa luận trong suốt quá trình thực. Bên cạnh đó tôi cũng muốn cảm ơn người bạn ở BM Vật Lý Lý Thuyết, đã dành thời gian trao đổi với tôi về những kiến thức liên quan đến đề tài của khóa luận nhờ vậy mà tôi hiểu hơn về kiến thức mình đang nghiên cứu.

Em xin cảm ơn các thầy cô khoa Vật Lý - Vật Lý Kĩ Thuật, đã truyền đạt kiến thức cho em trong những năm đầu đại học. Em xin chân thành cảm ơn thầy cô của Bộ môn Vật Lý Tin Học, đã tạo điệu kiện tối đa về cơ sở vật chất cũng như tạo dựng một môi trường học tập hết sức thân thiện. Từ đó mà em có thể thoải mái học tập và nghiên cứu.

Và cuối cùng, tôi xin cảm ơn những người bạn trong lớp Vật lý Tin học khóa 2015, những người em khóa 2016 đã cùng đồng hành với tôi trong suốt quá trình học tập tại Bộ môn Vật lý Tin học.

TP. Hồ Chí Minh, tháng 7 năm 2019.

Huỳnh Thị Hạ Vy

Mục lục

Danh sách hình vẽ

Danh sách bảng

Lời giới thiệu

Tích phân là phép tính quan trọng trong lĩnh vực giải tích. Được sử dụng thường xuyên trong các ngành Vật lý và Toán học. Để giải bài toán tích phần, người ta thường sử dụng các công thức giải tích để giải quyết. Nhưng đối với các tích phân đa chiều thì giải tích khó xử lý được.Vì thế phương pháp tích phân Monte Carlo được áp dụng vào để ước tính giá trị của tích phân. Phương pháp tích phân Monte Carlo là phương pháp sử dụng số ngẫu nhiên để tính toán kết quả gần đúng của tích phân. Phương pháp Monte Carlo bao gồm phương pháp cơ bản, phương pháp lấy mẫu có trọng tâm, phương pháp lấy mẫu phân tầng ... Độ chính xác của các phương pháp này đều dựa trên quan điểm thống kê. Nên các phương pháp này đều cho biết ước lượng sai số thống kê của phép tính.

Trong phạm vi đề tài này, tôi sử dụng thuật toán Vegas để xử lý các bài toán tích phân đa chiều. Thuật toán Vegas sử dụng hai phương pháp lấy mẫu có trọng tâm và phương pháp phân tầng để giảm phương sai tăng độ chính xác cho phép tính. .

Báo cáo đề tài gồm bốn chương chính như sau:

- **Chương 1: Kiến thức cơ sở xác suất và thống kê.** Giới các kiến thức cơ sở về lý thuyết xác suất và thống kê.
- **Chương 2: Tích phân Monte Carlo.** Trình bày kiến thức về tích phân Monte Carlo .
- **Chương 3: Thuật toán Vegas.** Trình bày về thuật toán Vegas được áp dụng trong tích phân Monte Carlo .
- **Chương 4: Kết luận và hướng phát triển.** Dưa ra kết luận về kết quả thu được của đề tài này và đánh giá hướng phát triển.

CHƯƠNG 1

Kiến thức cơ sở xác suất và thống kê

1.1 Biến ngẫu nhiên và xác suất

Biến ngẫu nhiên là một hàm ánh xạ với đặc điểm nó gán một giá trị bằng số cho kết quả đầu ra của một phép thử ngẫu nhiên.

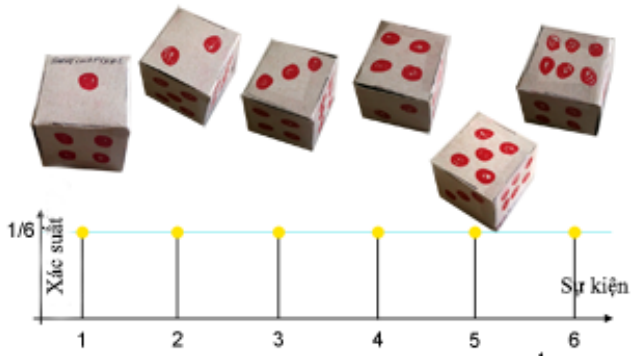
$$\mathbf{x}(\omega) = x \tag{1.1}$$

với ω là đại diện cho đầu ra của một thực nghiệm, x là một số thực (hay sự kiện), $\mathbf{x}$ là hàm ánh xạ (hay biến ngẫu nhiên). Tính ngẫu nhiên được thể hiện ở tham số đầu vào ω. Điều này dẫn đến đầu ra của hàm là ngẫu nhiên.
Đây chưa phải là định nghĩa đầy đủ của một biến ngẫu nhiên. Khái niệm khác liên quan đến định nghĩa của một biến ngẫu nhiên là khái niệm xác suất.
Xét một số thí dụ sau:

- Gieo một đồng xu trên mặt phẳng, đây là một phép thử. Kết quả có thể xảy ra là *Xuất hiện mặt sấp* hoặc *Xuất hiện mặt ngửa*. Như vậy xác suất cho các khả năng *Xuất hiện mặt sấp* và *Xuất hiện mặt ngửa* lần lượt là $\frac{1}{2}$ và $\frac{1}{2}$

Hình 1.1: Các sự kiện có thể xuất hiện khi gieo đồng xu

- Gieo một con súc sắc, đây là một phép thử. Kết quả có thể xảy ra là "Xuất hiện mặt k chấm"tương ứng với k = 1,2,3,..,6. Xác suất cho mỗi sự kiện "Xuất hiện k chấm"đều là $\frac{1}{6}$.



Hình 1.2: Các sự kiện có thể xuất hiện khi gieo súc sắc

Tổng quát, nếu một phép thử tạo ra n sự kiện khác nhau và khả năng xảy ra như nhau thì xác suất của mỗi sự kiện là $\frac{1}{n}$ chúng ta có thể nói rằng nếu kết quả của một quá trình nào đó phải là một trong n kết quả khác nhau.

1.2 Phân phối xác suất

Phân phối xác suất xác định xác suất cho từng sự kiện có thể xảy ra của một phép thử ngẫu nhiên. Hàm phân phối xác suất của biến ngẫu nhiên $\mathbf{x}$ được xác định như sau:

$$\mathbf{F}(x) = \mathbf{P}(\mathbf{x} \leq x) = \mathbf{P}\left\{\omega \in \Omega : \mathbf{x}(\Omega) \leq x\right\} \tag{1.2}$$

với mọi $x \in \mathbf{R}$ được gọi là hàm phân phối xác suất của biến ngẫu nhiên $\mathbf{x}$.

Đặc trưng của hàm phân phối xác suất hay gọi là hàm phân phối tích luỹ (CDF) là lấy xác suất các biến ngẫu nhiên bên trái của một giá trị $\mathbf{x}$ bất kì nào đó. Hàm này có đặc điểm là một hàm không giảm, tức là nếu $a < b$ thì $\mathbf{F}(a) \leq \mathbf{F}(b)$ vì sự kiện b đã bao gồm cả sự kiện a rồi.

1.2.1 Hàm khối xác suất của biến rời rạc

Hàm xác xuất để xác định xác suất tại mỗi giá trị $\mathbf{x}$ nào đó trong miền giá trị là bao nhiêu. Hàm xác xuất như vậy được gọi là hàm khối xác suất (PMF) Đối với các

biến ngẫu nhiên rời rạc. Giả sử miền giá trị của $\mathbf{x}$ là $\mathbf{D}$, tức $\mathbf{x}$:$\mathbf{\Omega} \mapsto \mathbf{D}$ thì hàm khối xác suất được xác định như sau:

$$p(x) = pX(x) = \begin{cases} \mathbf{P}(\mathbf{x} = x) & if x \in \mathbf{D} \\ 0 & if x \notin \mathbf{D} \end{cases} \tag{1.3}$$

Như vậy ta có thể thấy rằng hàm khối xác suất thực chất cũng là một xác suất nên nó mang đầy đủ tất cả các tính chất của xác suất như:

- $0 \leq p(x) \leq 1$
- $\sum_{x_i \in \mathbf{D}} p(x_i) = 1$

1.2.2 Hàm mật độ xác suất của biến liên tục

Hàm mật độ xác suất (PDF - Probability Density Function) đối với các biến ngẫu nhiên liên tục, dùng để ước lượng độ tập trung xác suất tại lân cận điểm nào đó. Hàm mật độ xác suất $f(x)$ tại điểm x được xác định bằng cách lấy đạo hàm của hàm phân phối tích lũy $\mathbf{F}(x)$ tại điểm đó:

$$f(x) = \mathbf{F}'(x) \tag{1.4}$$

Như vậy thì nơi nào $f(x)$ càng lớn thì ở đó mức độ tập xác suất càng cao. Từ đây ta cũng có thể biểu diễn hàm phân phối tích luỹ như sau:

$$\mathbf{F}(x) = \int_{-\infty}^{x} f(t)dt \tag{1.5}$$

Xác suất trong 1 khoảng (α,β) cũng có thể được tính bằng hàm mật độ xác suất:

$$\mathbf{F}(x) = \int_{\alpha}^{\beta} f(x)dx \tag{1.6}$$

Hàm mật độ xác suất cũng có 2 tính chất như xác suất như sau:

- Không âm: $f(x) \geq 0$, $\forall (x) \in \mathbf{R}$
- Tổng toàn miền bằng 1: $\int_{-\infty}^{\infty} f(x)dx = 1$

1.3 Kỳ vọng và phương sai

Giá trị kỳ vọng của X là tổng "sự kiện" nhân với xác suất của sự kiện đó. Như đã nhắc đến phần 1.2, xác suất được tính theo hàm PMF cho biến ngẫu nhiên rời rạc và hàm PDF cho biến ngẫu nhiên liên tục. Hay nói cách khác giá trị kỳ vọng là giá trị trung bình có trọng số.

$$\mathbf{E}[\mathbf{x}] = \sum_{i=0}^{N-1} X_i pmf(X_i) \tag{1.7}$$

Trong đó N là số "*sự kiện*" có thể xảy ra của một biến ngẫu nhiên rời rạc $\mathbf{x}$. Để mở rộng cho biến ngẫu nhiên liên tục $\mathbf{x}$:

$$\mathbf{E}[\mathbf{x}] = \int_{-\infty}^{\infty} \mathbf{x} pdf(\mathbf{x}) \tag{1.8}$$

Giả sử $\mathbf{x}$ là một biến ngẫu nhiên có tập giá trị sau 1, 2, 3, 4, 5, 6 và $\mathbf{F}[\mathbf{x}] = (\mathbf{x} - 3)^2$. Chú ý rằng $\mathbf{x}$ là biến ngẫu nhiên thì $\mathbf{E}[\mathbf{x}]$ cũng là một biến ngẫu nhiên. Như vậy

- $\mathbf{x} = 1, \mathbf{F}(1) = (1-3)^2 = 4$
- $\mathbf{x} = 2, \mathbf{F}(2) = (2-3)^2 = 1$
- $\mathbf{x} = 3, \mathbf{F}(3) = (3-3)^2 = 0$
- $\mathbf{x} = 4, \mathbf{F}(4) = (4-3)^2 = 1$
- $\mathbf{x} = 5, \mathbf{F}(5) = (5-3)^2 = 4$
- $\mathbf{x} = 6, \mathbf{F}(6) = (6-3)^2 = 9$

Nhận thấy, xác suất của một kết quả $\mathbf{Y}$ từ $\mathbf{F}[\mathbf{x}]$ bằng tổng xác suất của bất kỳ X nào thỏa $\mathbf{F}[\mathbf{x}] = \mathbf{Y}$.

- $\mathbf{P}(\mathbf{F}(0)) = \frac{1}{6}$
- $\mathbf{P}(\mathbf{F}(1)) = \frac{1}{6} + \frac{1}{6} = \frac{2}{6}$
- $\mathbf{P}(\mathbf{F}(4)) = \frac{1}{6} + \frac{1}{6} = \frac{2}{6}$
- $\mathbf{P}(\mathbf{F}(9)) = \frac{1}{6}$

Trong toán học có hai phương pháp để tính giá trị kỳ vọng của $\mathbf{F}[\mathbf{x}]$

- Phương pháp 1: Nếu phân phối xác suất của $\mathbf{F}[\mathbf{x}]$ đã biết trước, thì giá trị kỳ vọng được tính như sau:

$$\mathbf{E}[\mathbf{F}[\mathbf{x}]] = \mathbf{E}[\mathbf{Y}] = \sum Y_i \mathbf{P}(Y_i) \tag{1.9}$$

- Phương pháp 2: Nếu phân phối xác suất của $\mathbf{F}[\mathbf{x}]$ chưa biết trước, thì giá trị kỳ vọng được tính như sau: Đối với biến ngẫu nhiên rời rạc

$$\mathbf{E}[\mathbf{F}(\mathbf{x})] = \sum \mathbf{F}(\mathbf{x})\mathbf{P}(\mathbf{x}) \tag{1.10}$$

  Đối với biến ngẫu nhiên liên tục

$$\mathbf{E}[\mathbf{F}(\mathbf{x})] = \int \mathbf{F}(\mathbf{x})\mathbf{P}(\mathbf{x})d\mathbf{x} \tag{1.11}$$

Phương pháp 2 đóng vai trò quan trọng trong phương pháp Monte Carlo. Bởi vì hàm phân phối xác xuất $\mathbf{F}(\mathbf{x})$ có thể không được biết trước. Phương sai được xác định như sau

$$\sigma^2[\mathbf{Y}] = E[(\mathbf{Y} - \mathbf{E}[\mathbf{Y}])^2] \tag{1.12}$$

trong đó σ, độ lệch chuẩn, là căn bậc hai của phương sai. Từ những định nghĩa này, nó là dễ dàng cho thấy rằng với bất kỳ hằng số a

$$\mathbf{E}[a\mathbf{Y}] = a\mathbf{E}[\mathbf{Y}] \tag{1.13}$$

$$\sigma^2[a\mathbf{Y}] = a^2\sigma^2[\mathbf{Y}] \tag{1.14}$$

Hơn nữa, giá trị kỳ vọng của một tổng các biến ngẫu nhiên Yi là tổng giá trị kỳ vọng của Yi:

$$E\left[\sum_i Y_i\right] = \sum_i E[Y_i] \tag{1.15}$$

Từ các tính chất này, có thể rút ra biểu thức đơn giản hơn cho phương sai:

$$\sigma^2[\mathbf{Y}] = \mathbf{E}[\mathbf{Y}^2] - \mathbf{E}[\mathbf{Y}]^2 \tag{1.16}$$

Ngoài ra, nếu các biến ngẫu nhiên độc lập thì:

$$\sigma^2\left[\sum_i Y_i\right] = \sum_i \sigma^2[Y_i] \tag{1.17}$$

CHƯƠNG 2

Tích phân Monte Carlo

2.1 Tổng quát

Giá trị kỳ vọng của một hàm biến ngẫu nhiên f (X).

$$\mathbf{E}[f(\mathbf{x})] = \int f(\mathbf{x})P(\mathbf{x})d\mathbf{x} \tag{2.1}$$

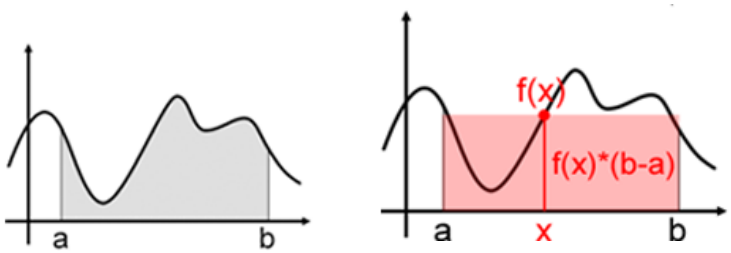
với $P(\mathbf{x})$ là là phân phối xác suất của biến ngẫu nhiên $\mathbf{x}$
Phương sai của biến ngẫu nhiên $\mathbf{x}$.

$$Var(\mathbf{x}) = \mathbf{E}[(\mathbf{x} - \mathbf{E}[X])^2] = \mathbf{E}[\mathbf{x}^2] - \mathbf{E}[\mathbf{x}]^2 \tag{2.2}$$

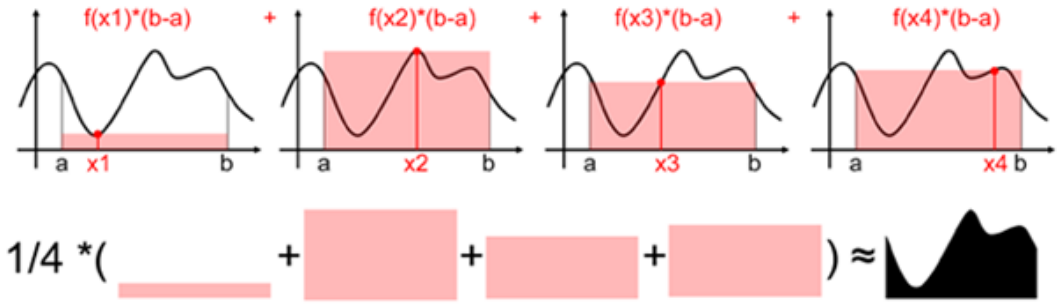
Xét tích phân Monte Carlo cơ bản, với hàm tích phân một chiều $f(x)$ từ a đến b.

$$\mathbf{F} = \int f(x)dx \tag{2.3}$$

Như hình 2.1 tích phân của hàm $f(x)$ có giá trị là diện tích bên dưới đường cong của hàm. Giả sử chọn một điểm ngẫu nhiên x nằm trong đoạn $[a, b]$, tính hàm $f(x)$ tại x và quả với $(b - a)$. Hình 2.1 cho thấy kết quả là một hình chữ nhật, trong đó $f(x)$ là chiều cao và $(b - a)$ chiều rộng của nó.



Hình 2.1: Tích phân trên miền [a,b] là diện tích dưới đường cong.



Hình 2.2: Giá trị tích phân gần được khi chọn các điểm ngẫu nhiên x

Nếu tính giá trị hàm tại $x1$, kết quả tính toán sẽ khá thấp, tính giá trị hàm tại $x2$ thì kết quả diện tích khá lớn. Nhưng khi tiếp tục tính toán hàm tại các điểm ngẫu nhiên khác nhau giữa a và b, cộng diện tích của các hình chữ nhật và tính trung bình, kết quả sẽ gần đúng với kết quả chính xác của tích phân. Và cách tính trung bình các hình chữ nhật này sẽ hội tụ đến diện tích của hàm tích phân khi số lượng mẫu được sử dụng trong phép tính tăng lên. Như minh họa minh họa trong hình 2.2. Giả sử tính tích phân tại bốn điểm ngẫu nhiên. Lấy kết quả của hàm f(x) tại 4 điểm ngẫu nhiên của x nhân với (b-a) và tính trung bình. Như vậy kết quả sẽ được xấp xỉ với kết quả chính xác.

2.2 Tích phân Monte Carlo với hàm mật độ xác suất đều

Giá trị trung bính của $\langle F^N \rangle$ khi biến ngẫu nhiên X_i là phân bố đều

$$\langle F^N \rangle = (b-a)\frac{1}{N}\sum_{i=0}^{N-1} f(x_i) \tag{2.4}$$

Trong đó N là số lượng mẫu, X_i là biến ngẫu nhiên phân bố đều trong khoảng $[a, b]$. Và xác suất của mỗi Xi đều bằng $\frac{1}{b-a}$ Theo định luật số lớn thì số mẫu N càng tăng đến vô cùng thì phép tính tích phân Monte Carlo sẽ hội tụ đến kết quả chính xác.

$$P(lim_{N\to\infty}\langle F^N \rangle = F) = 1 \tag{2.5}$$

$\langle F^N \rangle$ là một biến ngẫu nhiên, với $pdf(x_i) = \frac{1}{b-a}$. Giá trị kỳ vọng được tính như sau:

$$\begin{aligned} E[\langle F^N \rangle] &= E\left[(b-a)\frac{1}{N}\sum_{i=0}^{N-1} f(X_i)\right] \\ &= (b-a)\frac{1}{N}E[f(x)] \\ &= (b-a)\frac{1}{N}\sum_{i=0}^{N-1}\int_a^b f(x)pdf(x)dx \\ &= \frac{1}{N}\sum_{i=0}^{N-1}\int_a^b f(x)dx \\ &= \int_a^b f(x)dx \\ &= F \end{aligned} \tag{2.6}$$

2.3 Tích phân Monte Carlo với hàm mật độ xác suất bất kỳ

Để mở rộng tích phân Monte Carlo với hàm phân bố bất kỳ. Trong đó với $pdf(X_i)$ là hàm mật độ xác suất của biến ngẫu nhiên X_i thì $\langle F^N \rangle$ được xác định như sau.

$$\langle F^N \rangle = frac1N \sum_{i=0}^{N-1} \frac{f(X_i)}{pdf(X_i)} \tag{2.7}$$

Giá trị kỳ vọng được tính như sau:

$$\begin{aligned} E[\langle F^N \rangle] &= E\left[\frac{1}{N}\sum_{i=0}^{N-1}\frac{f(X_i)}{pdf(X_i)}\right] \\ &= \frac{1}{N}\sum_{i=0}^{N-1} E\left[\frac{f(X_i)}{pdf(X_i)}\right] \\ &= \frac{1}{N}\sum_{i=0}^{N-1}\int_\Omega \frac{f(X_i)}{pdf(X_i)}pdf(x)dx \\ &= \frac{1}{N}\sum_{i=0}^{N-1}\int_\Omega f(x)dx \\ &= \int_\Omega f(x)dx \\ &= \int_a^b f(x)dx \\ &= F \end{aligned} \tag{2.8}$$

2.4 Giảm phương sai

Giảm phương sai thì kết quả tích phân sẽ trở nên chính xác hơn. Do không phụ thuộc vào số lượng mẫu như đã đề cập tại phương trình 1.17 thì phương sai của $\langle F^N \rangle$ như sau.

$$\begin{aligned}
\sigma^2\left[\langle F^N\rangle\right] &= \sigma^2\left[\frac{1}{N}\sum_{i=0}^{N-1}\frac{f(X_i)}{pdf(X_i)}\right] \\
&= \frac{1}{N^2}\sum_{i=0}^{N-1}\sigma^2\left[\frac{f(X_i)}{pdf(X_i)}\right] \\
&= \frac{1}{N^2}\sum_{i=0}^{N-1}\sigma^2\left[Y_i\right] \\
&= \frac{1}{N}\sigma^2\left[Y\right]
\end{aligned} \tag{2.9}$$

Và độ lệch chuẩn với $Y_i = \frac{f(X_i)}{pdf(X_i)}$

$$\left[\langle F^N\rangle\right] = \frac{1}{\sqrt{N}}\sigma[Y] \tag{2.10}$$

Phương trình 2.10 đã chứng minh rằng lệch chuẩn hội tụ với $O(\sqrt{N})$. Phương trình này cũng cho thấy bằng cách giảm phương sai của mỗi Y_i, chúng ta có thể giảm phương sai của $\sigma\left[\langle F^N\rangle\right]$

CHƯƠNG 3

Thuật toán Vegas

3.1 Giới thiệu

Phương pháp Monte Carlo sử dụng các thuật toán để giải quyết các bài toán trên máy tính bằng cách lấy mẫu ngẫu nhiên. Kết quả của phương pháp Monte Carlo này càng chính xác khi số lượng mẫu càng tăng. Theo quy luật số lớn khi kích thước mẫu càng lớn thì tốc độ hội tụ của tích phân càng lớn, nhưng khi mở rộng số chiều của tích phân thì số lượng mẫu cũng phải tăng theo. Điều này cũng làm ảnh hưởng đến tốc độ tính toán của máy tính. Nếu cố định số lượng mẫu thì tốc độ hội tụ của tích phân sẽ giảm khi số chiều tăng lên. Nhìn chung, tốc độ hội tụ của phương pháp Monte Carlo là khá thấp.Vì vậy, phương pháp tích phân Monte-Carlo sử dụng thuật toán Vegas sẽ giải quyết hiệu quả với các tích phân nhiều chiều.

Đặc điểm thuật toán:[**vegas**]

1. Phương sai của tích phân thì được tính toán dễ dàng.
2. Hàm tính tích phân không cần liên tục và các hàm bước nhảy cũng không gặp khó khăn trong tính toán khi sử dụng thuật toán mới. Vì thế để tính tích phân nhiều chiều là việc đơn giản.
3. Tốc độ hội tụ không phụ thuộc vào số chiều của tích phân.
4. Đây là thuật toán thích nghi, nó tự động tập trung tính toán tích phân tại những vùng có tích phân quan trọng nhất.

Đặc điểm (1) và (3) là phổ biến trong tất cả phương pháp Monte Carlo. Với (4) là một tính năng quan trọng nhất trong thuật toán này. Vấn đề chính trong tính toán các tích phân nhiều chiều là việc tăng số lượng mẫu theo cấp số nhân khi tăng số chiều. Do đó, mục đích chung của thuật toán để tính toán tích phân nhiều chiều là thích nghi sau mỗi vòng lặp.

3.2 Tích phân Monte Carlo sử dụng thuật toán Vegas

Monte Carlo là phương pháp sử dụng số ngẫu nhiên để tính toán, vì vậy để có ước tính tích phân chính xác thì vấn đề giảm phương sai cần được quan tâm. Để giảm phương sai, tất cả những gì có thể làm là tăng số lượng mẫu M. Muốn giảm phương sai xuống hai lần thì phải tăng gấp bốn lần số mẫu. Đối với các hàm tích phân nhiều chiều, số lượng mẫu tăng theo cấp số nhân khi số chiều tăng lên. Do đó, có nhiều phương pháp đã được nghiên cứu để giảm phương sai mà không cần tăng số lượng mẫu M. Như vậy, phương pháp lấy mẫu trọng và phương pháp lấy mẫu phân tầng được sử dụng trong thuật toán Vegas để giảm phương sai.

3.2.1 Lấy mẫu trọng

Lấy mẫu trọng là phương pháp làm giảm phương sai bằng cách chọn hàm mật độ xác suất bất kỳ. Nhưng nếu chọn hàm mật độ xác suất tương tự với hình dạng của hàm f(x) thì kết quả tích phân càng chính xác. Tuy nhiên, nếu đặt nhiều mẫu tại những nơi mà tích phân có đóng góp lớn, thì phương sai của tích phân Monte Carlo giảm đáng kể. Ở đây hàm mật độ $p(x)$ được thay đổi để giảm phương sai. Như đã biết, phương sai được tối ưu khi:

$$p(x) = \frac{|f(x)|}{\int_\Omega dx|f(x)|} \tag{3.1}$$

Do đó, khi sử dụng gieo điểm trọng tích phân sẽ được tính tập trung tại nơi tích phân có độ lớn lớn nhất.

3.2.2 Lấy mẫu phân tầng

Để giảm phương sai, thể tích tích phân có thể được chia thành N khối thể tích nhỏ với các kích thước khác nhau. Sau đó tích phân Monte Carlo được thực hiên trong mỗi khối thể tích nhỏ. Với mỗi khối thể tích nhỏ này sử dụng M/N điểm ngẫu nhiên. Khi kích thước của khối thể tích nhỏ bị thay đổi thì phương sai sẽ được tính toán lại và được tối ưu khi phương sai của mỗi khối thể tích nhỏ này là như nhau ($=\sigma^2/N$). Do đó, khi sử dụng gieo điểm phân tầng, tích phân sẽ được tập trung tại nơi có sai số lớn nhất, tức là nơi tích phân vừa lớn và vừa dao động mạnh.

Bước đầu tiên trong thuật toán sử dụng phương pháp lấy mẫu phân tầng, là chia siêu khối thể tích của tích phân n chiều thành N^n siêu khối nhỏ như nhau. Bằng cách sử dụng một lưới hình chữ nhật đồng nhất. Mỗi siêu khối nhỏ sử dụng hai điểm để đóng góp vào việc tính tích phân Monte Carlo và phương sai. Các phương sai từ các siêu khối nhỏ sau đó được sử dụng để định nghĩa lại không gian lưới mới dọc theo mỗi trục và không gian lưới mới này sẽ được sử dụng cho lần lặp tiếp theo. Và phải giữ cho tổng số khối nhỏ không đổi. Do vậy qua mỗi vòng lặp thì các khối nhỏ có thể tập trung dần về nơi có phương sai lớn nhất trước đó, như vậy phương sai được giảm.
Thuật toán lấy mẫu phân tầng được sử dụng rộng rãi trong ngành Vật lý Lý thuyết và rất thành công cho nhiều ứng dụng tính toán hai chiều trở lên. Ưu điểm lớn nhất của phương pháp là khả năng thích nghi của tích phân đang được tính toán. Tuy nhiên khả năng thích nghi của nó được quyết định bởi số lượng đoạn được chia dọc theo mỗi trục N, và N bị giới hạn bởi tổng số tích phân M được cho phép ở mỗi vòng lặp.

$$M = 2N^n \tag{3.2}$$

Hạn chế này cho thấy một khuyết điểm khi số chiều lớn.

3.2.3 Thuật toán Vegas

Hạn chế từ phương pháp lấy mẫu phân tầng có thể tránh được bằng cách sử dụng phương pháp lấy mẫu trọng để giảm phương sai. Phương pháp lấy mẫu trọng dường như cho thấy kém hiệu quả hơn phương pháp lấy mẫu phân tầng. Tuy nhiên, khả năng thích nghi của phương pháp vượt trội hơn khi được thể hiện ở bài toàn có số chiều lớn. Vì vậy thuật toán Veags sẽ kết hợp cả hai phương pháp trên để xử lý bài toán.

Xét tích phân của một biến $\mathbf{x}(x1, ..., xn)$ trên không gian mẫu Ω.

$$I = \int_\Omega dx f(x) \tag{3.3}$$

Chon ngẫu nhiên M biến $\mathbf{x}$ từ một phân bố trong không gian mẫu Ω với xác suất $p(x)$. Cho thấy được tích phân xấp xĩ bằng

$$S^{(1)} = \frac{1}{M} \sum_{\mathbf{x}} \frac{f(\mathbf{x})}{p(\mathbf{x})} \tag{3.4}$$

Trong đó hàm mật độ xác suất được chuẩn hóa thành

$$\int_\Omega d\mathbf{x} p(\mathbf{x}) = 1 \tag{3.5}$$

Đại lượng $S^{(1)}$ được kì vọng sẽ dao động về giá trị đúng của tích phân. Tương ứng với mỗi tập hợp các điểm M ngẫu nhiên là 1 đại lượng ngẫu nhiên $S^{(1)}$ khác nhau. Phương sai của dao động này là:

$$\sigma^2 = \left\{ \int_\Omega d\mathbf{x} \frac{f^2(\mathbf{x})}{p(\mathbf{x})} - \left[\int_\Omega d\mathbf{x} f(\mathbf{x}) \right]^2 \right\} M^{-1} \tag{3.6}$$

Đối với tập hợp điểm M lớn thì phương sai là

$$\sigma^2 \cong \frac{S^{(2)} - S^{(1)}}{M-1} \tag{3.7}$$

tại:

$$S^{(2)} = \frac{1}{M} \sum_{\mathbf{x}} \left(\frac{f(\mathbf{x})}{p(\mathbf{x})} \right)^2 \tag{3.8}$$

Xem xét tích phân một chiều

$$I = \int_0^1 dx f(x) \tag{3.9}$$

Khởi tạo M điểm với mật độ xác suất đều. Ngoài việc ước tính giá trị tích phân 3.4 và phương sai 3.7, các điểm M dùng để tính tích phân cũng có thể được sử dụng để tính toán lại mật độ xác suất tốt hơn cho vòng lặp tiếp theo. Trong phương pháp này phương sai sẽ được giảm dần qua các vòng lặp.

Có một số kỹ thuật để tạo ra dãy số giả ngẫu nhiên với phân bố đều. Tuy nhiên, khó khăn hơn khi muốn tạo ra dãy số ngẫu nhiên với một phân bố bất kỳ có mật độ $p(x)$. Trong bài toán hiện tại sẽ đơn giản hơn nếu ta coi $p(x)$ là một hàm bước nhảy với Nbước. Xác suất của một số ngẫu nhiên được chọn từ một bước nhảy bất kỳ thì được xác định là một hằng số $1/N$ cho tất cả các bước $(0 = x_0 < \ldots X_N = 1, \Delta x_i = x_i - x_{i-1})$.

$$p(x) = \frac{1}{N \Delta x_i} \tag{3.10}$$

trong đó $x_i - \Delta x_i \leq x \leq x_i$, $i = 1, ..., N$ Phân phối xác suất được điều chỉnh theo các tích phân từng phần cụ thể bằng cách thay đổi kích thước của các đoạn Δx_i. Nếu N bị giới hạn bởi bộ nhớ máy tính thì N là một hằng số ($N = 50$đến 100). Cho các M điểm để tính tích phân và mật độ xác suất, kích thước các đoạn Δx_i được điều chỉnh lại bằng cách chia nhỏ mỗi đoạn thành$m_i + 1$ đoạn nhỏ hơn, tại

$$m_i = K \left\{ \left[\frac{\overline{f_i} \Delta x_i}{\sum_j f_j \Delta x_j} - 1 \right] \frac{1}{log \left[\frac{\overline{f_i} \Delta x_i}{\sum_j \overline{f_j} \Delta x_j} \right]} \right\} \tag{3.11}$$

với

$$\overline{f_i} \equiv \sum_{x \in x_i - \Delta x_i}^{x_i} f(x) \tag{3.12}$$

Toàn miền lấy tích phân sẽ bị chia nhỏ thành $k+1$ đoạn con ($k = 1000$). Số lượng đoạn con trong mỗi đoạn được xác định trong biểu thức 3.11, mỗi đoạn con của mỗi đoạn Δx_i là như nhau. Sự đóng góp của mỗi đoạn cho hàm trọng sẽ tăng tỉ lệ với sự đóng góp của nó cho tích phân $|f(x)|$, như yêu cầu trong phương trình 3.1. Vì tổng số đoạn Δx_i phải giữ nguyên bằng N, các đoạn con sẽ được gom lại thành các đoạn Δx_i mới sao cho số đoạn con trong mỗi đoạn Δx_i là như nhau. Kết quả, kích thước của các đoạn Δx_i được thay đổi nhưng tổng số đoạn Δx_i không đổi. Vì vậy, đoạn Δx_i có kích thước nhỏ nhất là nơi $|f(x)|$ lớn nhất. Lưới mới này sẽ được dùng và sẽ được điều chỉnh trong mỗi lần lặp cho tới khi lưới đạt tối ưu (tức $m_i = m_j$,i,j=1,. . . ,N). Mỗi tích phân và sai số được sử dụng để tính toán cho tổng tích phân.

Các giá trị ước tính của tích phân và phương sai của vòng lặp sẽ được sử dụng cho phần tính toán tích phân cuối cùng sau các vòng lặp.

$$I = \sigma_I^2 \sum_{i=1}^{m} \frac{I_i}{{\sigma_i}^2} \tag{3.13}$$

$$\sigma = \left(\sum_{i=1}^{m} \frac{1}{{\sigma_i}^2} \right)^{1/2} \tag{3.14}$$

Xem xét tích phân n chiều

Thuật toán mô tả bên dưới được tổng quát hóa với mục đích xử lý tích phân nhiều chiều. Minh họa cho sự thay đổi

$$\int_0^1 dx \int_0^1 dy f(x,y) \tag{3.15}$$

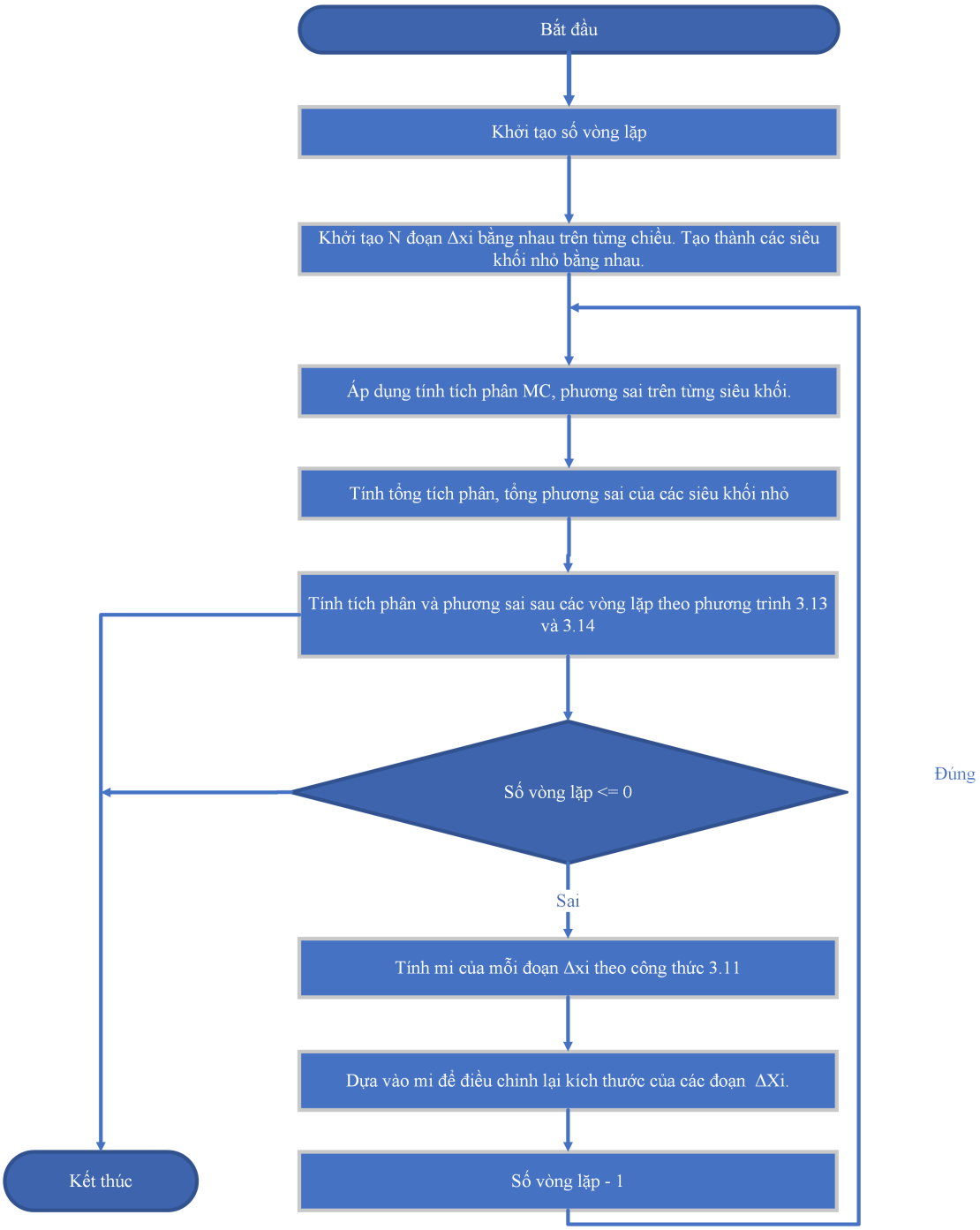
Lúc này, hàm mật độ xác suất được viết lại như sau:

$$p(x,y) = p_x(x) p_y(y) \tag{3.16}$$

Do đó thuật toán một chiều có thể áp dụng dọc theo mỗi trục. Phương trình 3.12 được viết lại

$$\overline{f_i} \equiv \sum_{x \in x_i - \Delta x_i}^{x_i} \sum_{y} \frac{f^2(x,y)}{P_y^2(y)} \tag{3.17}$$

Hình 3.1 mô tả lại các bước thực hiện của thuật toán Vegas.



Hình 3.1: Các bước thực hiện thuật toán Vegas

Các bước thực hiện

- Khởi tạo khối siêu thể tích nhỏ bằng nhau.

  Giả sử chia đoạn [0,1] thành N đoạn Δx_i. Với $i = 0, 1, ..., N-1$. Sử dụng hàm *rand()* có sẵn trong thư viện của C/C++ để tạo biến ngẫu nhiên đều trong đoạn [0,1]. Xác suất $p(x)$ của biến x trong đoạn Δx_i bằng $\frac{1}{N\Delta x_i}$
  Xét:

$$x_0 = 0$$
$$x_1 = x_0 + \Delta x_0$$
$$x_2 = x_1 + \Delta x_1$$
$$...$$
$$x_N = x_{N-1} + \Delta x_{N-1} = 1$$

  Sau đó gán $rc = x_i$ thì giá trị của rc dao động từ 0 đến 1, vậy biến x ngẫu nhiên trên đoạn $[a, b]$ có xác suất $p(x)$ bằng

$$x = a + rc(b - a) \tag{3.18}$$

  Theo cách giải thích trên, để tạo số ngẫu nhiên đa chiều $\mathbf{x}(x_1, x_2, x_3, ..., x_n)$. Xác suất $p(\mathbf{x})$ bằng:

$$p(\mathbf{x}) = p(x_1)p(x_2)...p(x_n) \tag{3.19}$$

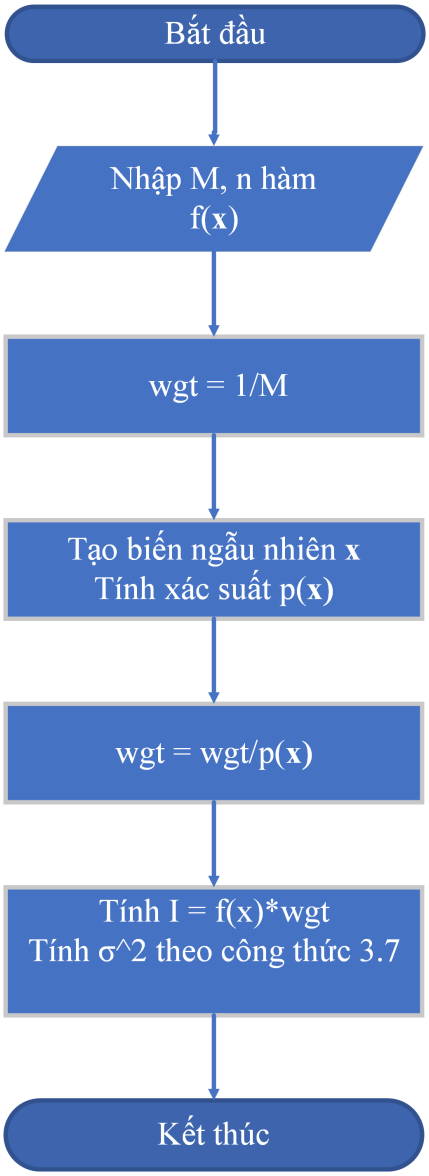
  Xét tích phân trong miền n chiều, khối siêu thể tích cần được chia thành các siêu khối nhỏ. Sau đó, tích phân MC sẽ được tính trong các khối nhỏ này. Để tạo ra các khối nhỏ bằng nhau, cần chia N đoạn Δx_i bằng nhau theo mỗi chiều.

- Ước tính tích phân của từng khối siêu thể tích nhỏ với trọng số wgt.

  Như vậy ta sử dụng các đoạn [0,1] và ban đầu chia đoạn này thành N đoạn nhỏ Δx_i bằng nhau. Với $i = 0, 1, ..., N-1$. Với công thức $M = 2N^n$. Khởi tạo số mẫu M dùng để tính tích phân, số chiều n. Số điểm ngẫu nhiên cho mỗi khối bằng 2. Suy ra số lượng các khối siêu thể tích nhỏ. Khởi tạo các miền xác định của từng chiều của biến ngẫu nhiên $x(x_1, x_2, x_3, ..., x_n)$. Mỗi biến giá trị x_1, x_2, x_3,... đều được tạo ra thông qua việc lấy ngẫu nhiên từ các đoạn nhỏ Δx_i từ đoạn [0,1]. Như vậy ban đầu Δx_i đều bằng nhau thì mật độ các điểm lấy ngẫu nhiên trên miền xác định đều như nhau. Khi đã tạo được biến ngẫu nhiên $\mathbf{x}$ với xác suất p($\mathbf{x}$) bên trong mỗi khối siêu thể tích nhỏ thì sau cùng tích phân MC bên trong mỗi khối siêu thể tích nhỏ sẽ được tính với trọng số wgt theo công thức sau

$$I = \frac{1}{M}\frac{f(\mathbf{x})}{p(\mathbf{x})} \tag{3.20}$$

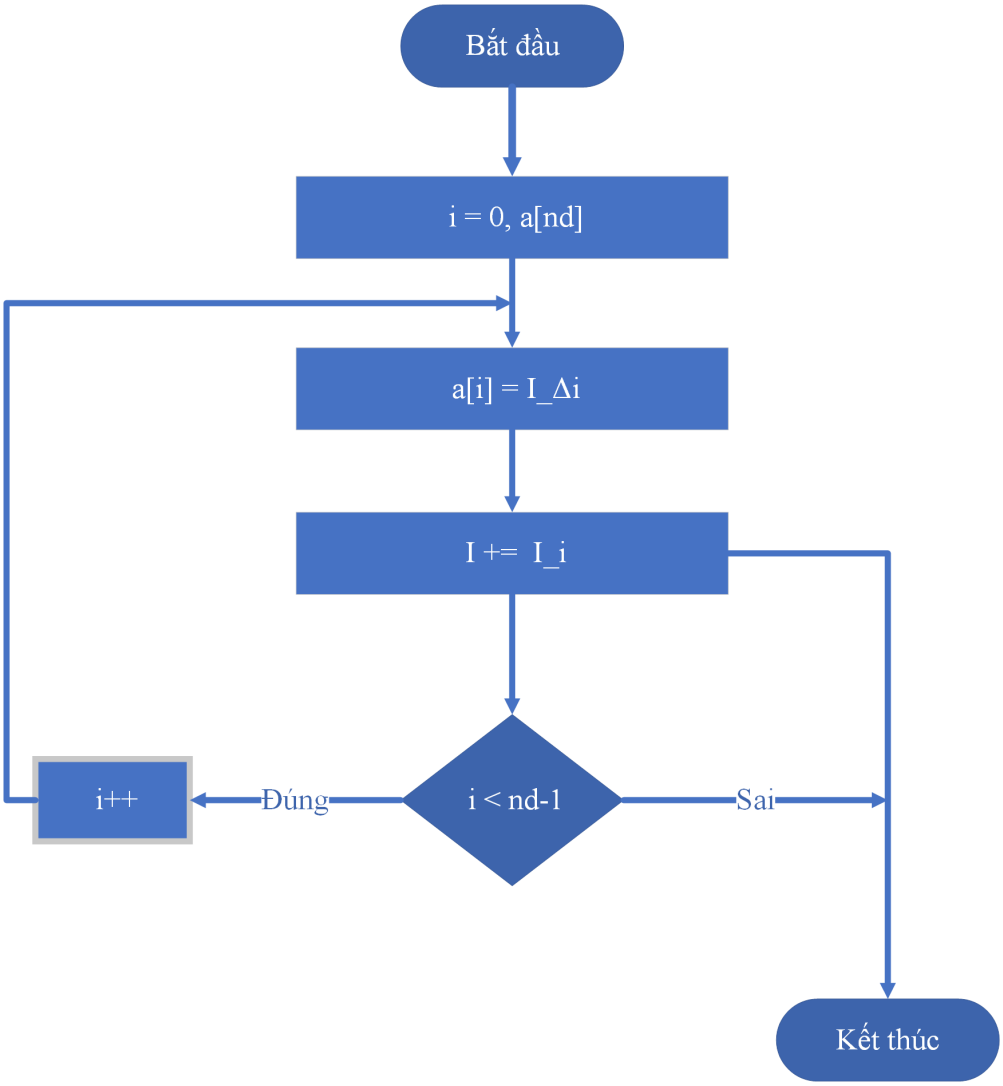
Sơ đồ khối trong hình 3.2 mô tả các bước tính I tại **x**.



Hình 3.2: Tích phân bên trong khối siêu thể tích

- Sau khi đã tính các khối tích phân nhỏ thì dễ dàng tính được tổng tích phân của nó dựa vào các thông tin tích phân và phương sai của vòng lặp trước để tính giá trị m_i trong phương trình 3.11

  Sơ đồ khối trong hình 3.3 thể hiện thuật toán ước tính tổng tích phân của các đoạn Δx_i.



Hình 3.3: Tính tổng tích phân trên các đoạn Δx_i

Trong đó:

- i=1, 0, ..., nd (nd là số đoạn Δx_i)
- Mảng a[nd] để lưu giá trị ước tính tích phân trên mỗi đoạn Δx_i.
- I là tổng ước tính tích phân của các đoạn Δx_i.

Như vậy các giá trị m_i được tính như sau

$$m_i = I_\Delta x_i / I_j$$

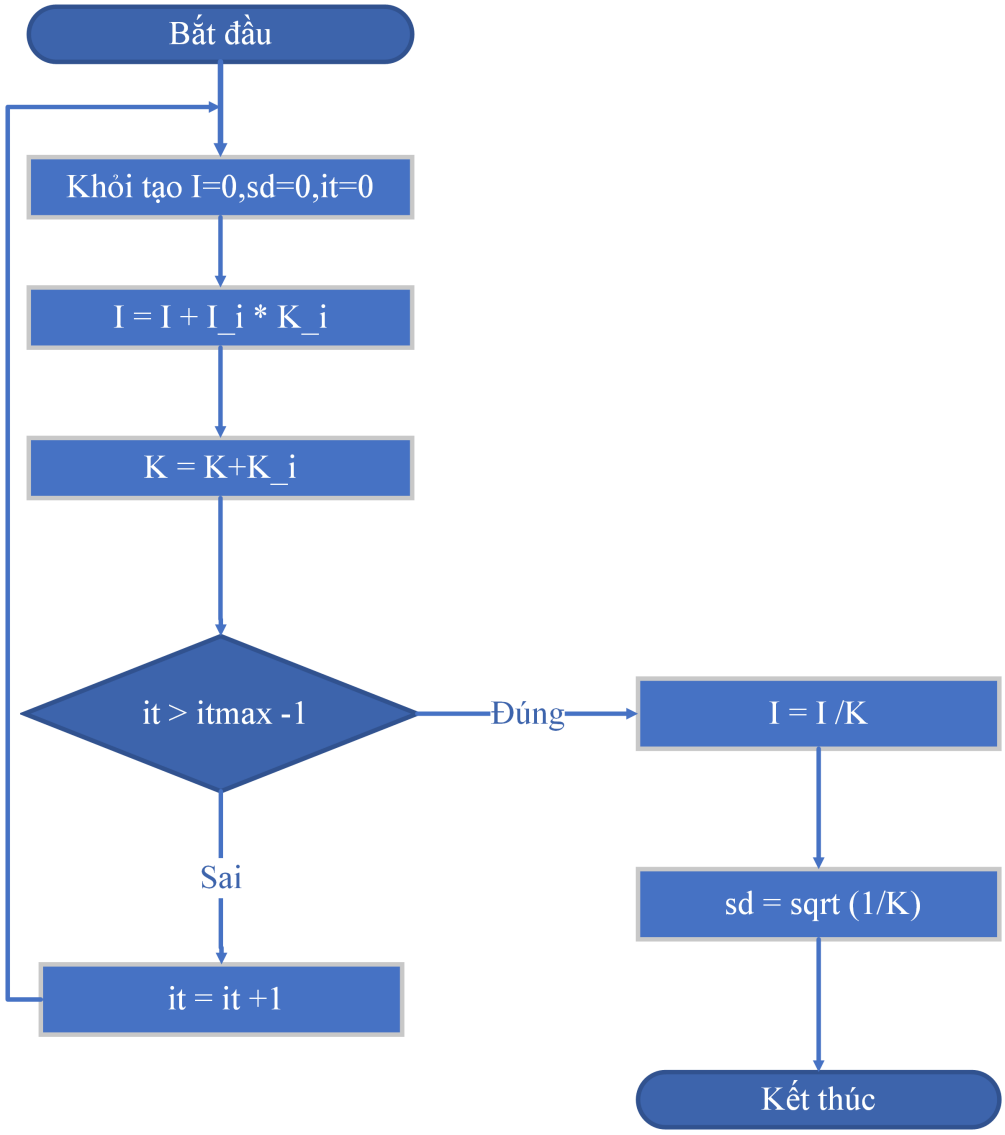
trong đó I_j là tổng tích phân thuộc vòng lặp trước đó. Thấy rằng giá trị m_i sau khi tính toán nó thể hiện cho tỉ lệ kích thước mà đoạn Δx_i có được trên tổng kích thước các đoạn Δx_i. như được trình bày ở phần 3.2.3. Sau đó lấy các giá

trị m_i để điều chỉnh lại các đoạn Δx_i. Với thuật toán Vegas, kích thướt các đoạn Δx_i sẽ được điều chỉnh theo từng chiều. Kết quả kích thước các khối siêu thể tích sẽ được điều chỉnh. Hay nói cách khác, mật độ điểm gieo sẽ tập trung tại nhưng nơi có tích phân lớn. Từ đó giảm sai số cho ước tính tích phân.

- Ước tổng tính tích phân và độ lệch chuẩn qua các vòng lặp.

  Khi đã có giá trị tích phân f(x) tại x và phương sai σ^2 bên trong từng khối siêu thể tích nhỏ. Sau đó ước tính giá trị tích phân và độ lệch chuẩn trong từng vòng lặp.

  Sơ đồ khối trong hình 3.4 thể hiện thuật toán ước tính tích phân và độ lệch chuẩn qua các vòng lặp.



Hình 3.4: Tính giá trị hàm f(x) tại x với trọng số

Trong đó

- it là biến đếm vòng lặp.
- itmax là số vòng lặp.
- I là ước tính tích phân khi kết thúc vòng lặp.
- I_i là Ước tính tích phân qua các vòng lặp.

Khi kết thúc vòng lặp đây sẽ là kết quả ước tính tích phân và sai số cuối cùng của hàm $f(x)$.

CHƯƠNG 4

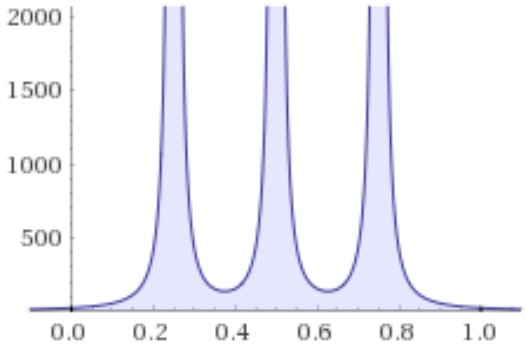
Kết luận và hướng phát triển

4.1 Kết quả đạt được

4.1.1 Xét các hàm 1 chiều

- Xét hàm f(x) có 3 đỉnh nhọn lấy tích phân trên miền xác định [0,1]

$$I = \int_0^1 \left(\frac{1}{(x-0.25)^2 + 10^{-6}} + \frac{1}{(x-0.5)^2 + 10^{-6}} + \frac{1}{(x-0.75)^2 + 10^{-6}} \right) dx \tag{4.1}$$



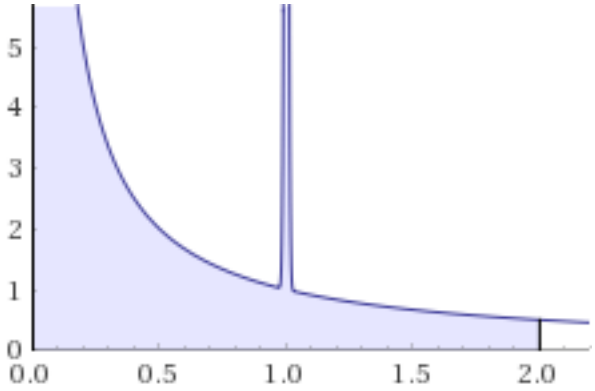
Hình 4.1: Dạng đồ thị của f(x)

Số mẫu	Số vòng lặp	Kết quả I	Sai số σ	Kết quả giải tích số
10000	1	9370.37	29.30	9410.11
10000	10	9410.12	0.07	9410.11
10000	100	9410.12	0.02	9410.11

Bảng 4.1: Bảng kết quả tích phân 1 chiều

- Xét hàm f(x) có đỉnh nhọn cao và hẹp, lấy tích phân trên miền xác định $[10^{-10}, 2]$

$$I = \int_{10^{-10}}^1 \left(\frac{1}{x} + 20exp(-10^{-4}(x-1)^2) \right) \tag{4.2}$$



Hình 4.2: Dạng đồ thị của f(x)

Số mẫu	Số vòng lặp	Kết quả I	Sai số σ	Kết quả giải tích số
10000	1	11.4698	0.6	$log(2.10^2) + 2.sqrt(\frac{\pi}{10})$
10000	10	24.0734	0.0002	$log(2.10^2) + 2.sqrt(\frac{\pi}{10})$
10000	100	24.0734	4.039e-05	$log(2.10^2) + 2.sqrt(\frac{\pi}{10})$

Bảng 4.2: Bảng kết quả tích phân 1 chiều

4.1.2 Xét các hàm 2 chiều

- Xét tích phân

$$I = \int_{-1}^{1}\int_{-1}^{1} exp(-20(x^2 + y^2))dxdy \tag{4.3}$$

Số mẫu	Số vòng lặp	Kết quả I	Sai số σ	Kết quả giải tích số
10000	1	0.157403	0.0003	≈ 0.15708
10000	10	0.157029	0.0002	≈ 0.15708
10000	100	0.156698	0.0001	≈ 0.15708

Bảng 4.3: Bảng kết quả tích phân 2 chiều

4.1.3 Xét các hàm 3 chiều

- Xét tích phân sau

$$I = \int_{0}^{1}\int_{0}^{1}\int_{0}^{1} \frac{1}{xyz + 10^{-6}}dxdydt \tag{4.4}$$

Số mẫu	Số vòng lặp	Kết quả I	Sai số σ	Kết quả giải tích số
10000	1	574.415	125.527	462.216
10000	10	461.803	0.425806	462.216
10000	100	462.31	0.0964302	462.216

Bảng 4.4: Bảng kết quả tích phân 3 chiều

4.1.4 Xét các hàm 4 chiều

- Xét hàm f(x) có 3 đỉnh nhọn lấy tích phân trên miền xác định [0,1]

$$I^n = \left(\frac{1}{a\pi^{\frac{1}{2}}}\right)^n \int_0^1 d^n x \exp\left(-\sum_{i=1}^{n} \frac{(x_i - 0.5)^2}{a^2}\right) \tag{4.5}$$

  Với $n = 4, a = 0.1$

Số mẫu	Số vòng lặp	Kết quả I	Sai số σ	Kết quả giải tích số
10000	1	1.18126	0.1271	1
10000	10	1.00069	0.00126	1
10000	100	0.999867	0.0004	1

Bảng 4.5: Bảng kết quả tích phân 4 chiều

4.2 Kết luận

Đề tài tập trung giải quyết các bài toán tích phân trong không gian 1 chiều đến 4 chiều. Kết quả ước tính tích phân từ báo cáo khóa luận so với kết quả chính xác từ phương pháp giải tích số gần như bằng nhau. Sự chênh lệch của ước tính tích phân thì nằm trong khoảng sai số mà phương pháp tính toán được. Vì thế, ước tính tích phân được xem như là chính xác so với kết quả thực tế. Ngoài ra sản phẩm của đề tài đã xử lý thành công các bài toán có hàm lấy tích phân có các đỉnh nhọn cao và hẹp trong không gian đa chiều (1 chiều đến 4 chiều).

Bên cạnh việc xử lý được các vấn đề đã đề cập ở trên thì báo cáo khóa luận vẫn chưa xử lý chính xác các bài toán lớn hơn 4 chiều. Và đôi khi vẫn cho ra ước tính sai lệch khá lớn so với kết quả chính xác đối với các hàm tích phân đặc biệt trong không gian 4 chiều.

4.3 Hướng phát triển đề tài

Dựa vào kết quả và phân tích đạt được thì đề tài cần phát triển thêm khả năng tính toán các tích phân lớn hơn 4 chiều. Tìm hiểu và phân tích các kết quả từ các thuật toán tính tích phân khác của nhiều tác giả khác nhau, để từ đó có thể tối ưu hóa ước tính tích phân.

Phụ lục A

Phần code đầy đủ

Dưới đây là mã chương trình đầy đủ để giải các bài toán tích phân trong không gian 1 chiều đến 4 chiều.

```
#define ALPH 1.5
#define NDMX 100
#define MXDIM 6
#define TINY 1.0e-68
#define DIM 4

#include <iostream>
#include <ctime>
#include <cmath>
#include <string>
double pi=3.14159265359;

using namespace std;

class infoIteration
{
public:
    double Wgt;
    double sWgt;
    double sInt;
};
class infoBins
{
public:
    double ti;
    double tsi;
};
class infoBin
```

```
29 {
30 public:
31     double f2;
32     double fb;
33     double f2b;
34     unsigned long samples2;
35 };
36
37 void rebin(double wgt_avg, int nd, double wgt[], double xin[],
       double xi[])
38 {
39     int i;
40     int k = 0;
41     double residues = 0.0;
42     double xn = 0.0;
43     double xo = 0.0;
44
45     for (i = 0; i < nd - 1; i++)
46     {
47         while (wgt_avg > residues)
48         {
49             residues += wgt[k++];
50         }
51         if (k > 1)
52             xo = xi[k - 2];
53         xn = xi[k - 1];
54         residues -= wgt_avg;
55         xin[i] = xn - (xn - xo) * residues / wgt[k - 1];
56     }
57     for (i = 0; i < nd - 1; i++)
58     {
59         xi[i] = xin[i];
60     }
61     xi[nd - 1] = 1.0;
62 }
63
64 double randomNumber(double a, double b)
65 {
```

```
66      return a + (b - a) * rand() / RAND_MAX;
67  }

69  void vegas(double regn[], int ndim, void (*fxn)(double x[],
        double *f),unsigned long samples, int loops, double *tgralin
        , double *sdin)
70  {
71      static info_iteration Ai;
72      static info_bins Ab;
73      static info_bin Ax;
74      static int count, ndo, mds;
75      static double xi[MXDIM][NDMX];
76      int it,i, j, k,nd, ng;
77      unsigned int samples_2;
78      double tgral,sd,calls,dv2g,dxg,rc,wgt,xn,xnd,xo,xJac,f,
        xrand;
79      int index[MXDIM], kg[MXDIM];
80      double x[MXDIM],d[NDMX][MXDIM],dt[MXDIM],r[NDMX],xin[NDMX],
        dx[MXDIM];
81
82      mds = ndo = 1;
83      for (j = 0; j < ndim; j++)
84          xi[j][0] = 1.0;
85      Ai.sInt = 0.0;
86      Ai.sWgt = 0.0;
87      count = 1;
88      nd = NDMX;
89      ng = 1;
90      if (mds)
91      {
92          ng = (int)pow(samples / 2.0 + 0.25, 1.0 / ndim);
93          mds = 1;
94          if ((2 * ng - NDMX) >= 0)
95          {
96              mds = -1;
97              samples_2 = ng / NDMX + 1;
98              nd = ng / samples_2;
99              ng = samples_2 * nd;
100         }
```

```
101     }
102     for (k = 1, i = 0; i < ndim; i++)
103         k *= ng;
104     samples_2 = (samples / k > 2) ? (samples / k) : (2);
105     calls = (double)samples_2 * (double)k;
106     dxg = 1.0 / ng;
107     for (dv2g = 1, i = 0; i < ndim; i++)
108         dv2g *= dxg;
109     dv2g = 1.0 / (samples_2 - 1);
110     xnd = nd;
111     dxg *= xnd;
112     xJac = 1.0 / calls;
113     for (j = 0; j < ndim; j++)
114     {
115         dx[j] = regn[j + ndim] - regn[j];
116         xJac *= dx[j];
117     }
118     if (nd != ndo)
119     {
120         for (i = 0; i < (nd > ndo ? nd : ndo); i++)
121             r[i] = 1.0;
122         for (j = 0; j < ndim; j++)
123             rebin(ndo / xnd, nd, r, xin, xi[j]);
124         ndo = nd;
125     }
126     for (it = count; it <= loops + count - 1; it++)
127     {
128         Ab.ti = Ab.tsi = 0.0;
129         for (j = 0; j < ndim; j++)
130         {
131             kg[j] = 1;
132             for (i = 0; i < nd; i++)
133                 d[i][j] = 0.0;
134         }
135     while (1)
136     {
137         Ax.fb = 0.0;
138         Ax.f2b = 0.0;
```

```
139         Ax.samples_2 = 0;
140         for (k = 0; k < samples_2; k++)
141         {
142          wgt = xJac;
143          for (j = 0; j < ndim; j++)
144          {
145              xrand = gfsr_rand(0, 1);
146              xn = (kg[j] - xrand) * dxg + 1.0;
147              index[j] = ((int)xn < NDMX) ? ((int)xn) : (NDMX);
148              index[j] = (index[j] > 1) ? (index[j]) : (1);
149              if (index[j] > 1)
150              {
151                  xo = xi[j][index[j] - 1] - xi[j][index[j] -
    2];
152                  rc = xi[j][index[j] - 2] + (xn - index[j]) *
    xo;
153              }
154              else
155              {
156                  xo = xi[j][index[j] - 1];
157                  rc = (xn - index[j]) * xo;
158              }
159
160              x[j] = regn[j] + rc * dx[j];
161              wgt *= xo * xnd;
162          }
163          fxn(x, &f);
164          f *= wgt;
165          Ax.f2 = f * f;
166          Ax.fb += f;
167          Ax.f2b += Ax.f2;
168          for (j = 0; j < ndim; j++)
169          {
170              if (mds >= 0)
171              {
172                  d[index[j] - 1][j] += Ax.f2;
173              }
174          }
```

```
175         }
176         Ax.f2b = sqrt(Ax.f2b * samples_2);
177         Ax.f2b = (Ax.f2b - Ax.fb) * (Ax.f2b + Ax.fb);
178         if (Ax.f2b <= 0.0)
179             Ax.f2b = TINY;
180         Ab.ti += Ax.fb;
181         Ab.tsi += Ax.f2b;
182         if (mds < 0)
183         {
184             for (j = 0; j < ndim; j++)
185                 d[index[j] - 1][j] += Ax.f2b;
186         }
187         for (k = ndim - 1; k >= 0; k--)
188         {
189             kg[k] %= ng;
190             if (++kg[k] != 1)
191                 break;
192         }
193         if (k < 0)
194             break;
195     }
196     Ab.tsi *= dv2g;
197     Ai.Wgt = 1.0 / Ab.tsi;
198      Ai.sInt += Ai.Wgt * Ab.ti;
199     Ai.sWgt += Ai.Wgt;
200     tgral = Ai.sInt / Ai.sWgt;
201     sd = sqrt(1.0 / Ai.sWgt);
202     *sdin = sd;
203     *tgralin = tgral;
204     for (j = 0; j < ndim; j++)
205     {
206         xo = d[0][j];
207         xn = d[1][j];
208         d[0][j] = (xo + xn) / 2.0;
209         dt[j] = d[0][j];
210         for (i = 1; i < nd - 1; i++)
211         {
212             rc = xo + xn;
```

```
213                 xo = xn;
214                 xn = d[i + 1][j];
215                 d[i][j] = (rc + xn) / 3.0;
216                 dt[j] += d[i][j];
217             }
218             d[nd - 1][j] = (xo + xn) / 2.0;
219             dt[j] += d[nd - 1][j];
220         }
221         for (j = 0; j < ndim; j++)
222         {
223             rc = 0.0;
224             for (i = 0; i < nd; i++)
225             {
226                 if (d[i][j] < TINY)
227                     d[i][j] = TINY;
228                 r[i] = pow((1.0 - d[i][j] / dt[j]) / (log(dt[j]) -
    log(d[i][j])), ALPH);
229                 rc += r[i];
230             }
231             rebin(rc / xnd, nd, r, xin, xi[j]);
232         }
233 }
234 count += loops;
235 }
```