Monte-Carlo integrálás

Korszerű számítási módszerek a fizikában I.

Medveczky Attila

2021.05.07.

1. Bevezetés

Az élet számos területén, legyen az fizika, biológia, tőzsde, stb., előfordulnak olyan egy vagy többdimenziós integrálok, amelyeknek nincs analitikus megoldásuk.

$$I = \int_{\Omega} f(\mathbf{x}) d\mathbf{x} \tag{1}$$

Ebben az esetben fordulunk a numerikus számításokhoz. Egy ilyen algoritmustól a következőket várhatjuk:

- Közelítse meg a lehető legjobban az integrál értékét,
- Legyen hibabecslés,
- Belátható időn belül legyen meg az eredmény.

Az idők során több ilyen eljárás született. Nem létezik egy mindenre jól használható, "tökéletes" metódus; az adott problémához a megfelelő eszközt kell kiválasztani. Jelen projekt keretében a Monte-Carlo módszert fogom részletesebben bemutatni.

2. Klasszikus módszerek

A klasszikus módszereket alapvetően 2 fő csoportba sorolhatjuk (illetve van több típus is, de talán ez a 2 a legelterjedtebb):

- Newton-Cotes módszerek: az integrált egyenletesen kiválasztott helyeken értékeljük ki.
- Gauss-kvadratúrák: Specifikusan választjuk ki, hogy hol értékeljük ki az integrált, ez általában jobb eredményre vezet, megfelelő használat esetén.

2.1. Newton-Cotes módszerek

Ezen képletek általános alakja (1 dimenzióban):

$$\int_{a}^{b} f(x) d(x) \approx \sum_{i=0}^{n} w_i f(x_i) \tag{2}$$

Ahol $x_i = x_0 + hi$, azaz a lépésköz, w_i pedig a megfelelő súlyozás. Ezek a módszerek akkor hatékonyak, ha a függvényértékek meghatározottak az egymástól egyenlő távolságra lévő pontokban. Ha az n alappontszám nagyon nagy, előfordulhat a Runge-jelenség: a hibatag exponenciálisan növekedhet. Ide tartozik pl. a trapézszabály, Simpson-szabály, téglalapszabály.

2.2. Gauss-kvadratúrák

Ezek általános alakja megegyezik (2) -vel, azonban nem csak a súlyozást választhatjuk szabadon, hanem a helyeket is, ahol a függvényt kiértékeljük. Míg a Newton-Cotes formulák n kiértékelési pont esetén legfeljebb csak n-ed (ha n páratlan) vagy $n-1$-ed (ha n páros) fokú polionomok esetén ad megfelelő eredményt, a Gauss-kvadratúrák viszont $2n-1$-ed fokú polinomig működnek jól.

2.3. Többdimenziós integrálok

Az előbbi módszereket általánosíthatjuk d dimenzióra:

$$\int_\Omega d^d u f(u_1, \ldots, u_d) = \frac{1}{n^d} \sum_{j_1=0}^{n} \ldots \sum_{j_d=0}^{n} w_{j_1} \ldots w_{j_d} f\left(\frac{j_1}{n}, \ldots, \frac{j_d}{n}\right) + O\left(\frac{1}{n^2}\right) \quad (3)$$

A függvényt összesen $N = n^d$-szer kell kiértékelni. A számítási idő arányos N-el, és megfigyelhető, hogy a hiba $N^{-2/d}$ módon változik. A hibakorlát $O(N^{-2/d})$ d növekedésével jelentősen romlik.

3. Monte-Carlo módszer

Vegyük az alábbi integrált:

$$I = \int_\Omega d^d u f(u_1, \ldots, u_d) \quad (4)$$

A Monte-Carlo közelítése:

$$E = \frac{V}{N} \sum_{n=1}^{N} f(x_n) \quad (5)$$

Ahol V az integrálási tartomány bennfoglaló d dimenziós téglatest térfogata, N a kiértékelések száma, $x_n = (u_{1_n}, \ldots, u_{d_n})$ egy pont $[-L, L]^d$ tartományon belül ($-L, L$ a bennfoglaló test oldalainak határa).

A nagy számok törvénye alapján E az integrál valódi értékéhez tart:

$$\lim_{N \to \infty} \frac{V}{N} \sum_{n=1}^{N} f(x_n) = I \quad (6)$$

A hiba tárgyalásához vezessük be a szórásnégyzetet:

$$\sigma^2(f) = \int dx (f(x) - I)^2 \quad (7)$$

Belátható, hogy

$$\int dx_1 \dots \int dx_N \left(\frac{1}{N} \sum_{n=1}^{N} f(x_n) - I \right)^2 = \frac{\sigma(f)^2}{N} \tag{8}$$

Ezt úgy is értelmezhetjük, hogy a hiba a Monte-Carlo módszer esetén $\frac{\sigma(f)}{\sqrt{N}}$.

A centrális határeloszlás tétele alapján a valószínűsége annak, hogy a közelítésünk a $I - \frac{a\sigma(f)}{\sqrt{N}}$ és $I + \frac{b\sigma(f)}{\sqrt{N}}$ határok között legyen:

$$\lim_{N\to\infty} P\left(-\frac{a\sigma(f)}{\sqrt{N}} \leq \frac{1}{N} \sum_{n=1}^{N} f(x_n) - I \leq \frac{b\sigma(f)}{\sqrt{N}} \right) = \frac{1}{\sqrt{2\pi}} \int_{-a}^{b} dt e^{-\frac{t^2}{2}} \tag{9}$$

Ez azt is jelenti, hogy a hiba az eljárás során $\frac{1}{\sqrt{N}}$-el változik, a d dimenziótól függetlenül. Ez az oka annak, hogy magasabb dimenziós integrálások numerikus számítása során népszerű választás a Monte-Carlo módszer.

3.1. Variancia-redukáló módszerek

Nem csak a hiba változik $\frac{1}{\sqrt{N}}$-el, hanem az integrál értékéhez való konvergálás is. Rendkívül sok módszer van ennek a gyorsítására, jelen összefoglalóban azonban csak egyet részletezek, amelyet a feladat megvalósítása során alkalmaztam.

3.1.1. Importance sampling

Matematikailag a módszer az integrálási változócserével analóg:

$$\int dx f(x) = \int \frac{f(x)}{p(x)} p(x) dx = \int \frac{f(x)}{p(x)} dP(x) \tag{10}$$

ahol

$$p(x) = \frac{\partial^d}{\partial x_1 \dots \partial x_d} P(x) \tag{11}$$

Ha úgy vesszük, hogy $p(x) \geq 0$, és 1-re normált, akkor tekinthetjük valószínűségi sűrűségfüggvénynek. Ha a $P(x)$ eloszlás alapján generálunk $x_1 \dots x_n$ random számot, akkor a Monte-Carlo közelítés:

$$E = \frac{1}{N} \sum_{n=1}^{N} \frac{f(x_n)}{p(x_n)} \tag{12}$$

Látjuk, hogy a legjobb eredményt akkor érjük el, ha $p(x)$ hasonlít $f(x)$-re, mivel akkor a mintavételezésünk olyan, hogy nagyrészt az integrál értéke szempontjából releváns pontokat kapunk.

4. Implementáció

A kurzuson tanultak szerint a számítást végző részét a programnak egy külön headerbe raktam, a main.cpp-ban csak a main() függvény található (Meg egy futási idő mérő rész, erről később).

A számítógép determinisztikusan működik, így csak pszeudorandom számokkal lehet dolgozni. A program először az egyszerű módszerrel (Crude method) határozza meg a feladatban megadott integrált. Ehhez az (x, y, z) koordinátákat egyenletes eloszlás alapján generáljuk, a megadott integrálási határokon belül. Mivel a $\sigma(f)$ értékét nem egyszerű meghatározni, azt a következőképpen közelítjük:

$$S^2 = \frac{V}{N} \sum_{n=1}^{N} (f(x_n))^2 - E^2 \tag{13}$$

Az Importance sampling esetén, az (x, y, z) koordinátákat normális eloszlásból vesszük, ez illeszkedik a legjobban a feladatban megadott függvényalakhoz. Ennél a módszernél a hiba közelítése:

$$S^2 = \frac{1}{N} \sum_{n=1}^{N} \left(\frac{f(x_n)}{p(x_n)} \right)^2 - E^2 \tag{14}$$

5. Diszkusszió

Ahogy vártuk, az egyszerű módszerrel a kapott érték igencsak nagy hibával rendelkezik, és nagyjából az [5, 6] tartományon belül mozog, míg az Importance sampling módszerrel a hiba értéke tizede a sima módszer hibájának, a kapott érték pedig az [5.5, 5.6] tartományon belül esik. Az eredményt a WolframAlpha 3D integrál számolójával ellenőriztem, az 5.56833-at adott eredményül.

Érdekességképpen a programot átírtam Python nyelvre, és megnézetm a futási időket:

```
Integral value: 5.49443 +/- 0.0309382 (Crude method)
Integral value: 5.56554 +/- 0.00408744 (Importance sampling method)
Time taken by function: 391583 microseconds
Program ended with exit code: 0
```

1. ábra.

```
Uniform sampling : s =  5.575111986749553  +/-  0.03128744507268846
Gaussian sampling: s =  5.569335777434436  +/-  0.004090397071678337
Runtime of the program is 125022263.76533508microseconds
```

2. ábra.

A különbség nagyon jelentős, Python környezetben ugyan az a program $n = 10^6$ futásszámmal kb. 125 másodpercig futott, míg C++-ban 0.39 másodpercig; ez több, mint 300-szoros különbség.

6. Felhasznált források

- Newton - Cotes formula - Wikipedia
- Gauss kvadratúra - Wikipedia
- Monte - Carlo method - Wikipedia
- The basics of Monte Carlo integration - Towards data science
- Introduction to Monte Carlo methods
- MONTE CARLO TECHNIQUES
- Monte Carlo theory and practice
- Importance sampling
- Introduction to numerical programming
- Stackoverflow