Projekt dokumentáció:

Koroknai Botond

4. félév

1. A Monte-Carlo integrálás:

Az első legfontosabb különbség a hagyományos numerikus intergráláshoz képest, hogy nem egy szabályos rácson futnánk végig a kiértékelendő pontokon, hanem véletlen választjuk ki őket, ezzel lényegesen gyorsítva a folyamatot. Miután lefutattuk megfelelő mennyiségű lépésre, a tartományon kivűl és belül eső pontok arányából megtippelhetjük az integrálás erdményét.

2. A függvény definiálása:

```
template <typename Func, typename Dom>
double MonteCarlo(Func integrand, Dom domain, double xmin, double xmax, double ymin, double ymax, double zmin, double zmax, int Samples = 1000)
```

Szokásos módon Template-el lett megírva a függvény, hogy tudjon kezelni különböző típusú változókat. A Func típusú változó reprezentálja az integrandust, és a Dom típusú a tartományt, melyen a függvény mintavételezni fog. Ezt követően a határokat double ként definiáltam az egyszerűség kedvéért és az alapértelmezett mintavételt 1 milliónak rögzítettem.

3. Random szám generálás:

```
std::random_device rd;
std::mt19937 gen(rd());
std::uniform_real_distribution<double> xDistribution(xmin, xmax);
std::uniform_real_distribution<double> yDistribution(ymin, ymax);
std::uniform_real_distribution<double> zDistribution(zmin, zmax);
```

Az első sor egy olyan objektumot hoz létre, amit c++-ban, szokás használni a véletlenszámok inicializálásához. A második sor azt mondja meg, hogy a szám generátor a Mersenne Twister algoritmus alapján fog működni. A mardék három sor a koordinátáknak megfelelő határok között fog egyenletes eloszlású random értékkel visszatérni, ezek segítségével fogjuk a koordinátákat generálni a Monte - Carlo integrálás során.

4. A pontok kiértékelése:

```
int indomain = 0;
for (int i = 0; i < samples; ++i)
{
    double x = distributionx(gen);
    double y = distributiony(gen);
    double z = distributionz(gen);

    if (domain(x, y, z))
    {
        double value = integrand(x, y, z);
        indomain++;
    }
}
```

Először is létrehoztam egy indomain nevű integer változót, ennek segítségével számoltam a tartományon belül található pontok számát. Ezt követően a for ciklussal elindítottam egy samples hosszúságó iterációt, melynek során, az előző részekben definiált eloszlás alapján generáltam egy véletlenszerű pontot. Ezek után a domain-nek megadott lambda függvénybe betáplálom ezen értékeket, és megvizsgálom, hogy kielégítik-e a megadott feltételt. Ha a bool True értékkel tér vissza akkor eggyel növelem az indomain számláló értékét, máskülönben csak továbbhaladok a következő pontra.

5. Az integrálás értéke:

```
double volume = (xmax - xmin) * (ymax - ymin) * (zmax - zmin);
double integral = static_cast<double>(indomain) / static_cast<double>(samples) * volume;
return integral;
```

Először az x,y,z tartományok összeszorzásával meghatározom a tartomány térfogatát. Majd meghatározomom a feltételt kielégítő pontok és az összes mintavétel arányát, majd ezt megszorozva a térfogattal, megkapom az integrálás értékét. A static_cast azt a célt szolgálja, hogy double típusra váltjuk a két int típuson tárolt változót, így a szorzást követően pontosabb eredményt kapjunk.