Monte Carlo

Programmering II - Elixir

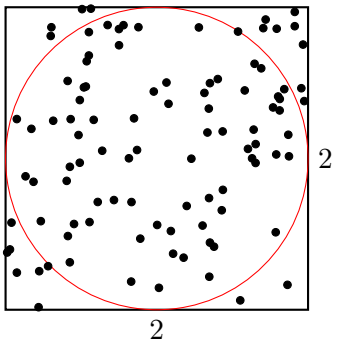
Kenan Dizdarevic

20 Februari, 2023

Inledning

Vi skall i denna uppgift approximera π bättre än vad Arkimedes och Zu Chongzhi lyckades med. Metoden som vi skall använda är *Monte Carlo metoden* som är relativt enkel.

Monte Carlo metoden bygger på att vi kan skapa en sekvens av slumpmässigt valda tal. Om vi exempelvis kastar pilar i kvadraten nedan kan vi beräkna vad sannolikheten är för att pilarna skall landa i cirkeln.



Figur 1: 100 slumpmässigt kastade pilar i kvadraten

Om vi kastar fler pilar kommer vi att ha en bättre approximation på vad sannolikheten att pilen landar i cirkeln är. Detta exempel är trivialt och sannolikheten beräknas genom att dividera cirkelns area med kvadratens area. Med enkla beräkningar kommer vi fram till att sannolikheten är $\pi/4$.

För att approximera π skall vi använda oss av samma metod. Vi vet att förhållandet mellan antal pilar i cirkeln och antal kastade pilar ger oss $\pi/4$. Detta innebär att $\pi = 4 \cdot \frac{antalpunktericirkeln}{totaltantalpunkter}$. Vi ser att om vi utför fler kast kommer vi få en bättre approximation av π.

Implementering

Slumpmässiga pilar

För att vi skall kunna approximera π måste vi generera slumpmässiga sekvenser av pilar som kastas. Dessa pilar genereras med hjälp av funktionen `dart/1` som tar emot en radie. Funktionen returnerar sedan sant eller falskt beroende på om pilen träffar cirkeln eller ej. Koden för detta presenteras nedan:

```
def dart(r) do
    x = Enum.random(0..r)
    y = Enum.random(0..r)
    :math.pow(r, 2) > :math.pow(x, 2) + :math.pow(y, 2)
end
```

Vi använder oss av `Enum.random/1` för att generera ett slumpmässigt tal i ett givet intervall. Sedan använder vi oss av *Pythagoras sats* för att avgöra om pilen hamnade innanför cirkeln eller utanför.

Ackumulator

Vi har nu möjlighet att kasta massa pilar. Vi vill veta hur vår approximation förbättras för varje kast. Det vi gör är att vi kastar ”rundor” av pilar där vi ackumulerar antalet pilar som landar innanför cirkeln. Vi skapar funktionen `round/3` som tar emot antal pilar som skall kastas, cirkelns radie och det ackumulerade värdet som argument. Koden för implementeringen presenteras nedan:

```
def round(0, _, acc) do acc end
def round(k, r, acc) do
    case dart(r) do
      true ->
        round(k - 1, r, acc + 1)
      false ->
        round(k - 1, r, acc)
    end
end
```

Vi använder oss av `dart/1` som returnerar antingen sant eller falskt. Om `dart/1` returnerar sant har vi en träff. Vi skall addera 1 till ackumulatorn och ta bort en pil från de som skall kastas. Om vi istället har en miss uppdaterar vi antalet pilar.

Approximation

Vi har nu möjligheten att approximera π, därför skall vi försöka approximera bättre än Arkimedes. Arkimedes visste att π är lite mindre än 22/7 men

han var endast övertygad om att två decimaler var korrekt. Vi skall därför försöka approximera π med tre decimaler. När vi utför approximationen har vi tre parametrar som vi måste ta hänsyn till. Antalet rundor som skall exekveras, antal pilar som skall kastas och radien på cirkeln. För att approximera π till 3.141 använde vi 1000 rundor, 10 000 kast och en radie på 10 000. Felmarginalen vi uppnådde var ungefär 0.00003 vilket innebär att vi lyckades approximera de fyra första decimalerna korrekt. Om vi inte hade haft tillgång till `:math.pi()` hade vi behövt använda oss av en mer precis approximeringsmodell, exempelvis `Lebniz formel`.

Vi skall nu försöka approximera π bättre än vad Zu Chongzhi lyckades med. Chongzhi approximerade π till 355/113 där 6 decimaler är korrekta. Vi har återigen samma parametrar att ta hänsyn till. För att lyckas med approximationen modifierar vi koden när vi utför det rekursiva anropet. Vi multiplicerar antalet pilar som skall kastas med 2.

```
rounds(k - 1, j * 2, t, r, a)
```

Efter flertal försök lyckades vi äntligen approximera π till den sjunde decimalen, $\pi = 3.141592$. När vi approximerade till den sjunde decimalen använde vi 10 000 rundor, vi började med 10 000 kast som dubblades för varje omgång, vår radie var 10 000 000. Vi provade olika värden på parametrarna vilket antingen resulterade i en felaktig approximation eller att exekveringstiden blev för lång.

Slutsats

Slutligen ser vi att detta är en rolig och intressant uppgift som är verklighetsbaserad. Monte Carlo metoden används dessvärre inte för att approximera π utan för att simulera fysikaliska system eller utföra statistiska beräkningar.

Det finns bättre metoder för att approximera π, *Lebniz formel* är en sådan.