π Calculation

Hit or Miss Technique Application

Marco Grisanti

Sommario

1. Problema

Lo scopo di questo progetto è quello di stimare il valore del PI-Greco, una costante matematica indicata con la lettera greca π.

Per fare ciò, utilizzeremo la Tecnica "Hit or Miss" del Metodo Monte Carlo. Quindi, useremo variabili aleatorie artificiali e troveremo una stima del valore di π partendo dalle sole definizioni di:

- Area di un Quadrato $A_Q = l^2$
- Area di una Circonferenza $A_C = \pi r^2$

Gli algoritmi verranno sviluppati su Processing, un linguaggio di programmazione che consente di sviluppare diverse applicazioni come giochi, animazioni, contenuti interattivi e opere d'arte generativa. Eredita tutta la sintassi, i comandi e il paradigma di programmazione orientata agli oggetti dal linguaggio Java, ma, in più, mette a disposizione numerose funzioni ad alto livello per gestire in modo facile gli aspetti grafici e multimediali. È compatibile con i sistemi operativi Linux, macOS e Microsoft Windows.

1.1 Il Pi-Greco

Il PI-Greco (π) è una costante matematica il cui studio risale fin dall'antichità. Il primo ad approssimare scientificamente π fu Archimede di Siracusa che nel III secolo a.C. utilizzò poligoni regolari inscritti e circoscritti a una circonferenza. Egli notò che, aumentando il numero di lati, il rapporto tra il perimetro e l'area limita superiormente e inferiormente π. Infatti, utilizzando poligoni di 96 lati lo scienziato siracusano scoprì che:

$$\frac{223}{71} < \pi < \frac{22}{7} \text{ ovvero } 3{,}140 < \pi < 3{,}142$$

Il PI-Greco è una delle costanti matematiche più importanti, infatti esso è presente in molte formule matematiche, come ad esempio:

- Formula dell'Integrale di Gauss $\int_{-\infty}^{+\infty} e^{-x^2} dx = \sqrt{\pi}$
- Formula dell'Approssimazione di Stirling $n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$
- Ecc.

Per questo motivo, gli scienziati hanno cercato di trovare quante più cifre possibili di π in modo da avere una maggiore precisione nei risultati dei calcoli dove esso è presente.

1.2 Il Metodo Monte-Carlo

Il Metodo Monte Carlo è un termine che indica tutte quelle tecniche che fanno uso di variabili aleatorie artificiali (cioè generate al computer) per la risoluzione di problemi matematici. Ovviamente questo metodo non è molto efficiente, però, risulta essere l'unica alternativa quando si hanno situazioni nelle quali è difficile utilizzare i classici strumenti numerici matematici.

1.2.1 La Tecnica Hit or Miss

Sia $f(x)$ una funzione definita nell'intervallo $[a, b]$.

Sia $f(x)$ limitata, ovvero: $\exists c \in R \mid 0 \leq f(x) \leq c$

Si consideri il rettangolo: $W = \{(x, y) | a \leq x \leq b, 0 \leq y \leq c\}$

La probabilità che il vettore (x, y) di numeri casuali uniformemente distribuiti in W cada sotto la curva $f(x)$ è:

$$p = \frac{Area\ della\ Curva}{Area\ del\ Rettangolo} = \frac{\int_a^b f(x)dx}{c(b-a)}$$

Si può riscrivere la precedente equazione come segue:

$$\int_a^b f(x)dx = c(b-a)p$$

Una volta generati N vettori casuali, è possibile stimare p:

$$p \sim p' = \frac{N_H}{N}$$

Dove N_H rappresenta il numero di "Hits", ovvero il numero di volte in cui il punto (x_i, y_i) cade sotto la curva $f(x)$.

Quindi, nella precedente equazione si può sostituire p con p' ottenendo:

$$\int_a^b f(x)dx \sim c(b-a)\frac{N_H}{N}$$

2. Lavoro Svolto

Verranno analizzati due metodi per risolvere il problema precedentemente descritto.

2.1 Metodo 1

Il primo metodo che si analizzerà per stimare il valore di π consiste nel considerare una circonferenza di raggio R inscritta in un quadrato avente lato pari a $2R$.

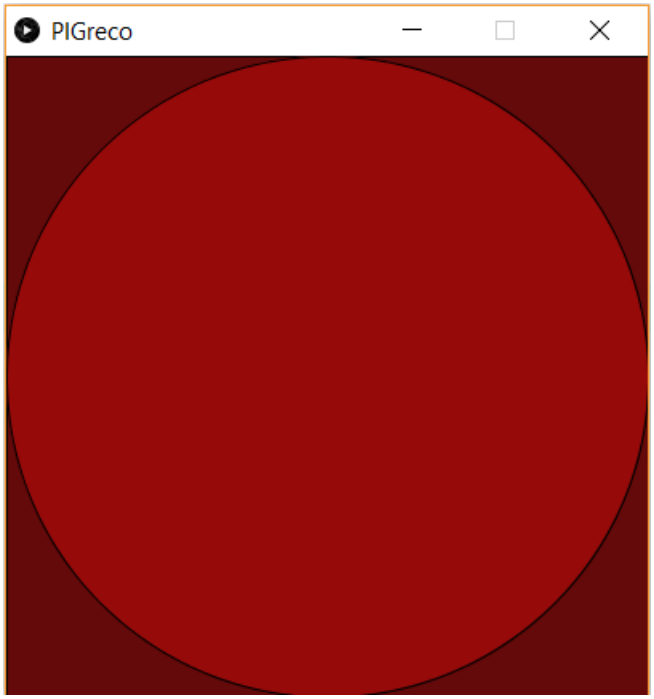

Figura 1 - Circonferenza inscritta in un Quadrato

Generati N punti nel quadrato con distribuzione uniforme, si avrà:

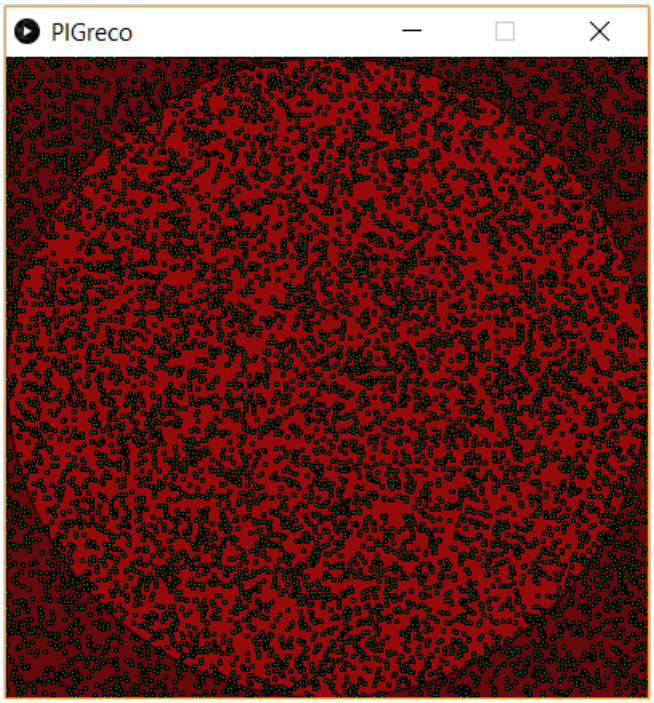

Figura 2 - Generazione di 10000 Punti nel Quadrato

In questo caso, si ha:

$$\frac{N_H}{N} = \frac{A_C}{A_Q} = \frac{\pi R^2}{(2R)^2} = \frac{\pi R^2}{4R^2} = \frac{\pi}{4}$$

Quindi:

$$\pi = 4\frac{N_H}{N}$$

Ponendo $N = 10^8$, si ottiene:

$$\pi = 3,1406066$$

2.1.1 Codice Sorgente del Metodo 1

```
public class Square {
    private float x;
    private float y;
    private float r;
    private color c;

    public Square(float _x, float _y, float _r, color _c) {
        x = _x;
        y = _y;
        r = _r;
        c = _c;
    }

    public void display() {
        fill(c);
        rectMode(RADIUS);
        rect(x, y, r, r);
    }

    public void run() {
        display();
    }
}
```

Figura 3 – Classe Square

```
public class Circle {
    private float x;
    private float y;
    private float r;
    private color c;

    public Circle(float _x, float _y, float _r, color _c) {
        x = _x;
        y = _y;
        r = _r;
        c = _c;
    }

    public void display() {
        fill(c);
        ellipseMode(RADIUS);
        ellipse(x, y, r, r);
    }

    public void run() {
        display();
    }
}
```

Figura 4 – Classe Circle

```
public class Point {
    private float x;
    private float y;
    private color c;

    public Point(float _x, float _y, color _c) {
        x = _x;
        y = _y;
        c = _c;
    }

    public void display() {
        fill(c);
        rectMode(RADIUS);
        rect(x, y, 1, 1);
    }

    public void run() {
        display();
    }
}
```

Figura 5 – Classe Point

```
import java.util.LinkedList;

// Variabili Globali
int N = 100000000;
int NHit = 0;
int R;
float centerX;
float centerY;
color squareColor = color(100, 10, 10);
color circleColor = color(150, 10, 10);
color pointColor = color(10, 150, 10);
Square s;
Circle c;
LinkedList<Point> p;

void setup() {
    size(400, 400);
    frameRate(1);
    calcPI1();
    //calcPI2();
}

void draw() {
    background(128);
    s.run();
    c.run();
    for (int i = 0; i < p.size(); i++) p.get(i).run();
}
```

Figura 6 – Parte Principale del Programma

```
void calcPI1() {
    R = width / 2; // Metà della Lunghezza della Finestra
    centerX = width / 2;
    centerY = height / 2;
    s = new Square(centerX, centerY, R, squareColor);
    c = new Circle(centerX, centerY, R, circleColor);
    p = new LinkedList<Point>();

    int x;
    int y;
    for (int i = 0; i < N; i++) {
        x = (int)(Math.random() * R * 2);
        y = (int)(Math.random() * R * 2);
        p.add(new Point(centerX - R + x, centerY - R + y, pointColor));
        if (dist(centerX, centerY, centerX - R + x, centerY - R + y) <= R) NHit++;
    }
    println(4.0 * NHit / N);
}
```

Figura 7 – Funzione "calcPI1"

2.2 Metodo 2

Il codice sorgente della precedente implementazione, con particolare riferimento alla funzione “calcPI1”, può sembrare un po’ complicato da capire. Per questo motivo, si presenta un’altra implementazione molto simile alla precedente, ma, per la quale è possibile scrivere un codice sorgente più elegante e più semplice da capire.

Questo metodo che si analizza per stimare il valore di π consiste nel considerare un quarto di circonferenza di raggio R dentro un quadrato avente lato pari a R.

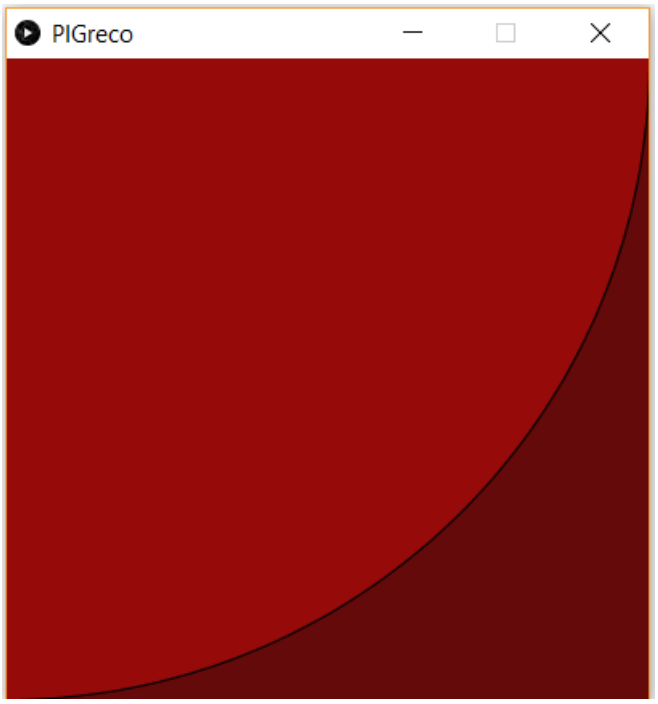

Figura 8 – Un quarto di Circonferenza dentro un Quadrato

Generati N punti nel quadrato con distribuzione uniforme, si avrà:

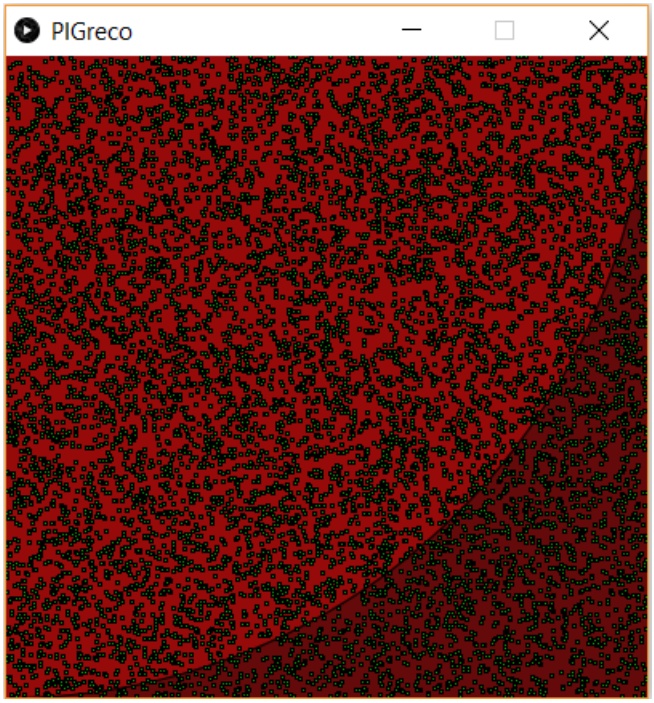

Figura 9 - Generazione di 10000 Punti nel Quadrato

In questo caso, si ha:

$$\frac{N_H}{N} = \frac{\frac{1}{4}A_C}{A_Q} = \frac{\frac{1}{4}\pi R^2}{R^2} = \frac{\pi}{4}$$

Quindi:

$$\pi = 4\frac{N_H}{N}$$

Ponendo $N = 10^8$, si ottiene:

$$\pi = 3,1417205$$

2.2.1 Codice Sorgente del Metodo 2

```
void calcPI2() {
    R = width; // Lunghezza della Finestra
    centerX = 0;
    centerY = 0;
    s = new Square(centerX, centerY, R, squareColor);
    c = new Circle(centerX, centerY, R, circleColor);
    p = new LinkedList<Point>();

    int x;
    int y;
    for (int i = 0; i < N; i++) {
        x = (int)(Math.random() * R);
        y = (int)(Math.random() * R);
        p.add(new Point(x, y, pointColor));
        if (dist(centerX, centerY, x, y) <= R) NHit++;
    }
    println(4.0 * NHit / N);
}
```

Figura 10 - Funzione "calcPI2"

2.3 Intervallo di Confidenza

Si vuole calcolare l'intervallo di confidenza per quest'ultimo metodo adoperato al variare di N con livello di fiducia α.

Si calcola

$$p' = \frac{N_H}{N}$$

Si stima la deviazione standard

$$\sigma = \sqrt{\frac{p'(1-p')}{N}}$$

Si fissa $\alpha = 0{,}003$ (pari al 99.7%).

L'intervallo di confidenza sarà uguale a $[4(p' - 3\sigma),\ 4(p' + 3\sigma)]$

```
PI: 3.184000    [3.031084, 3.336916]    N = 1000
PI: 3.138800    [3.089476, 3.188124]    N = 10000
PI: 3.141440    [3.125860, 3.157020]    N = 100000
PI: 3.139836    [3.134906, 3.144766]    N = 1000000
PI: 3.142208    [3.140650, 3.143765]    N = 10000000
PI: 3.141422    [3.140929, 3.141914]    N = 100000000
PI: 3.141525    [3.141370, 3.141681]    N = 1000000000
```

Figura 11 - Risultati di π al variare di N

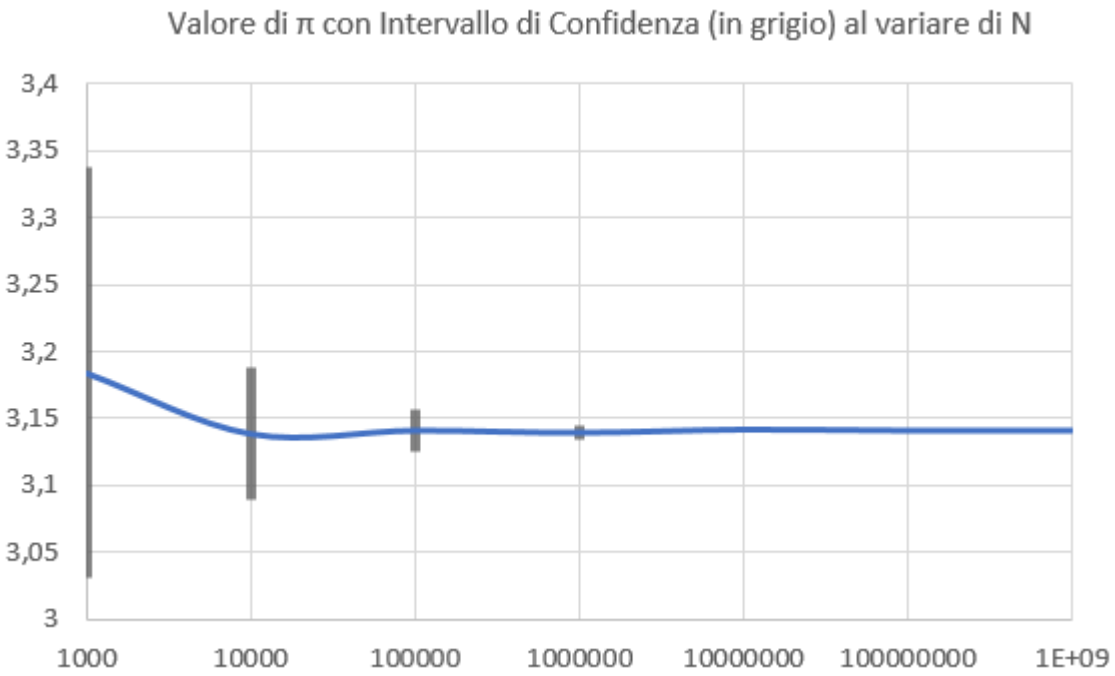

Figura 12 - Grafico Finale

3. Conclusioni

Quanto asserito da Archimede già nel III secolo a.C., senza la possibilità di utilizzare i potenti calcolatori esistenti oggi, era corretto, avendo sostenuto che il valore di π è compreso tra 3,140 e 3,142.

Grazie all'utilizzo di calcolatori specializzati per effettuare tali tipologie di calcoli, è possibile generare anche miliardi di numeri in pochi secondi, permettendo, utilizzando le tecniche sopra descritte, di trovare molte cifre di π all'istante.

4. Bibliografia

To Do