Il codice ha lo scopo di approssimare l'integrale di una funzione `integrand` nell'intervallo [a, b] utilizzando il metodo Monte Carlo. Questo metodo si basa sulla generazione di punti casuali e sul calcolo della proporzione di punti che cadono sotto la curva della funzione.

### Concetti Affrontati

1. \*\*Metodo Monte Carlo\*\*:

Il metodo Monte Carlo è una tecnica di integrazione numerica che utilizza la generazione di numeri casuali per approssimare il valore di un integrale. È particolarmente utile per funzioni complesse o in spazi ad alta dimensione. L'idea di base è generare punti casuali in un dominio noto e calcolare la proporzione di punti che soddisfano una certa condizione (ad esempio, cadere sotto una curva). Questa proporzione viene poi utilizzata per stimare l'integrale.

2. \*\*Integrale Indefinito\*\*:

Un integrale indefinito rappresenta l'insieme di tutte le primitive di una funzione, ovvero tutte le funzioni che, derivate, restituiscono la funzione originale. Nel contesto del metodo Monte Carlo, stiamo approssimando un integrale definito, che è l'area sotto la curva della funzione `integrand` nell'intervallo [a, b].

3. \*\*Funzione Integranda\*\*:

La funzione integranda è la funzione di cui si vuole calcolare l'integrale. Nel nostro codice, questa funzione è rappresentata da `integrand(x)`. La forma specifica di questa funzione non è mostrata nel codice, ma è fondamentale per il calcolo dell'integrale.

4. \*\*Rettangolo di Bounding\*\*:

Un rettangolo di bounding è un rettangolo che contiene completamente la funzione da integrare nell'intervallo considerato. Determinare il rettangolo di bounding è essenziale per limitare l'area in cui vengono generati i punti casuali. Nel nostro codice, il rettangolo di bounding ha base [a, b] e altezza `max\_f`, dove `max\_f` è il valore massimo della funzione `integrand` nell'intervallo [a, b].

5. \*\*Generazione di Numeri Casuali\*\*:

La generazione di numeri casuali è fondamentale per il metodo Monte Carlo. In questo codice, utilizziamo la funzione `srand(time(NULL))` per inizializzare il generatore di numeri casuali con un seme basato sul tempo corrente, garantendo che ogni esecuzione del programma produca una sequenza diversa di numeri casuali.

### Spiegazione del Codice

1. \*\*Determinazione del rettangolo di bounding\*\*:

```c

double max\_f = 0;

for (double x = a; x <= b; x += 0.001) {

double f\_x = integrand(x);

if (f\_x > max\_f) {

max\_f = f\_x;

}

}

```

In questa parte del codice, calcolo il valore massimo della funzione `integrand` nell'intervallo [a, b] con un incremento di 0.001. Questo valore massimo (max\_f) rappresenta l'altezza del rettangolo di bounding che contiene la funzione.

2. \*\*Apertura del file di output\*\*:

FILE \*file = fopen(output\_file, "w");

if (!file) {

printf("Errore nell'apertura del file %s\n", output\_file);

return 1;

}

```

Apro un file di output per salvare i punti generati. Se l'apertura del file fallisce, il programma termina con un messaggio di errore.

3. \*\*Generazione dei punti casuali e calcolo dell'integrale\*\*:

```c

int inside = 0;

srand(time(NULL)); // Inizializzazione del generatore di numeri casuali

for (int i = 0; i < n\_points; i++) {

double x = random\_double(a, b);

double y = random\_double(0, max\_f);

if (y <= integrand(x)) {

inside++;

}

// Salva i punti nel file

fprintf(file, "%f %f\n", x, y);

}

```

Inizializzo il generatore di numeri casuali e genero n\_points punti casuali all'interno del rettangolo di bounding. Per ogni punto (x, y), verifico se cade sotto la curva della funzione `integrand`. Se sì, incremento il contatore inside. Salvo ogni punto generato nel file di output.

4. \*\*Calcolo dell'area del rettangolo e dell'integrale approssimato\*\*:

```c

double rectangle\_area = (b - a) \* max\_f;

double integral = rectangle\_area \* ((double)inside / n\_points);

```

Calcolo l'area del rettangolo di bounding come prodotto della base (b - a) e dell'altezza max\_f. L'integrale approssimato è dato dall'area del rettangolo moltiplicata per la proporzione di punti che cadono sotto la curva (inside / n\_points).

5. \*\*Output dei risultati\*\*:

```c

printf("\nValore approssimato dell'integrale: %.6f\n", integral);

printf("Numero di punti interni: %d\n", inside);

printf("Errore stimato rispetto al valore noto (7.47626): %.6f\n", fabs(integral - 7.47626));

printf("Punti salvati nel file: %s\n", output\_file);

```

Stampo il valore approssimato dell'integrale, il numero di punti che cadono sotto la curva, l'errore stimato rispetto a un valore noto dell'integrale e il nome del file in cui sono stati salvati i punti.

6. \*\*Chiusura del file e termine del programma\*\*:

```c

fclose(file);

return 0;

```

Chiudo il file di output e termino il programma.

### Approfondimenti

- \*\*Precisione e Efficienza\*\*:

L'incremento di 0.001 nella determinazione del rettangolo di bounding è scelto per bilanciare la precisione e l'efficienza computazionale. Un passo più piccolo aumenterebbe la precisione ma richiederebbe più tempo di calcolo, mentre un passo più grande ridurrebbe il tempo di calcolo ma diminuirebbe la precisione.

- \*\*Proporzione di Punti\*\*:

La proporzione di punti che cadono sotto la curva rispetto al totale dei punti generati è utilizzata per stimare l'integrale. Questo approccio è basato sulla legge dei grandi numeri, che garantisce che la stima diventi più accurata con l'aumentare del numero di punti generati.

- \*\*Errore di Stima\*\*:

L'errore stimato rispetto a un valore noto dell'integrale è calcolato per valutare l'accuratezza del metodo Monte Carlo. Questo permette di confrontare l'approssimazione ottenuta con il valore esatto o noto dell'integrale.

```