
aracne

Classificazione Decimale Dewey:

515 (23.) MATEMATICA. ANALISI

GERARDO IOVANE

MATHEMATICAL ANALYSIS VIA APPLICATIONS IN COMPUTER SCIENCE

aracne

ISBN
979-12-218-1778-2

PRIMA EDIZIONE
ROMA 7 MARZO 2025

INDICE

Introduzione

L'obiettivo di questo testo è aiutare il discente nel comprendere l'utilità dell'Analisi Matematica in corsi di studio come Informatica od Ingegneria informatica, mostrando le applicazioni di definizioni e teoremi dell'Analisi Matematica a casi concreti come la Cifratura, l'Intelligenza Artificiale, le Basi Dati, il Cloud Computing, l'Analisi delle Immagini, l'Analisi di file audio o video, Applicazioni con Arduino, Applicazioni con Raspberry, IoT, Data Science, Ingegneria del Software, ecc. Il codice di Applicazione viene realizzato in linguaggio C, ovvero il linguaggio che tutti i discenti apprendono al primo anno di studio.

Questo testo non può essere sostitutivo di testi di Analisi Matematica, perché è teso alle applicazioni e pertanto spesso al fine di non appesantire la notazione o distogliere il discente dall'applicazione pratica si è stati anche lievemente non formali.

In questo libro con esempi specifici affrontiamo varie applicazioni di definizioni e di teoremi dell'Analisi Matematica relativa a funzioni reali di una variabile reale con applicazioni in Informatica. Ogni Applicazione:

1. Indica il tema matematico affrontato,
2. Indica un Applicazione,
3. Spiega l'Applicazione,
4. Fornisce il codice in linguaggio C,
5. Spiega il codice esemplificato.

1. Insiemi Numerici: Applicazioni

Applicazione 1. Teorema di Divisione Euclidea

1. **Teorema:** Dati due interi a e b (con b>0), esistono due interi q (quoziente) e r (resto) tali che a=bq+r con 0≤r<b.
2. **Applicazione:** Calcolo del quoziente e del resto in un algoritmo di hashing.
3. **Spiegazione:** Un algoritmo di hashing può usare la divisione euclidea per generare un hash code da una chiave numerica. Il resto della divisione determina il bucket dell'hash.
4. **Codice C:**

```
#include <stdio.h>


int main() {
   int a = 125, b = 12, q, r;
   q = a / b; // Quoziente
   r = a % b; // Resto


   printf("Quoziente: %d\n", q);
   printf("Resto: %d\n", r);
   return 0;
}
```

5. **Spiegazione del Codice:**

- a / b calcola il quoziente q.
- a % b calcola il resto r.
- I risultati sono stampati per mostrare come a=bq+r.

Applicazione 2. Teorema Fondamentale dell'Aritmetica

1. **Teorema:** Ogni numero intero maggiore di 1 può essere rappresentato come un prodotto unico di numeri primi.
2. **Applicazione:** Scomposizione in fattori primi per cifratura RSA.
3. **Spiegazione:** La crittografia RSA si basa sulla difficoltà di fattorizzare grandi numeri in primi. Questo processo è implementato scomponendo un numero dato.
4. **Codice C:**

```
#include <stdio.h>
```

```
void fattori_primi(int n) {
   printf("Fattori primi di %d: ", n);
   for (int i = 2; i <= n; i++) {
      while (n % i == 0) {
         printf("%d ", i);
         n /= i;
      }
   }
   printf("\n");
}

int main() {
   int n = 84;
   fattori_primi(n);
   return 0;
}
```

5. **Spiegazione del Codice:**

- Il programma itera da 2 a nn per trovare i divisori primi.
- while (n % i == 0) verifica la divisibilità per ii, stampando ii come fattore.

Applicazione 3. Teorema di Bézout

1. **Teorema:** Dati due interi a e b, esistono interi x e y tali che ax+by=MCD(a,b).
2. **Applicazione:** Algoritmo esteso di Euclide per trovare coefficienti in crittografia.
3. **Spiegazione:** L'algoritmo esteso di Euclide calcola x e y, fondamentali per calcolare gli inversi modulari in RSA.
4. **Codice C:**

```
#include <stdio.h>

int gcd_extended(int a, int b, int *x, int *y) {
   if (a == 0) {
      *x = 0;
      *y = 1;
```

```
    return b;
  }
  int x1, y1;
  int gcd = gcd_extended(b % a, a, &x1, &y1);
  *x = y1 - (b / a) * x1;
  *y = x1;
  return gcd;
}

int main() {
  int a = 35, b = 15, x, y;
  int gcd = gcd_extended(a, b, &x, &y);
  printf("MCD: %d, x: %d, y: %d\n", gcd, x, y);
  return 0;
}
```

5. **Spiegazione del Codice:**

- gcd_extended calcola il massimo comune divisore e i coefficienti x e y.
- Utilizza la ricorsione per risolvere ax+by=MCD(a,b).

Applicazione 4. Teorema di Densità dei Razionali

1. **Teorema:** Tra due numeri reali distinti esiste sempre un numero razionale.
2. **Applicazione:** Interpolazione di dati discreti con valori medi razionali.
3. **Spiegazione:** L'interpolazione richiede valori razionali tra punti dati discreti per costruire curve lisce.
4. **Codice C:**

```
#include <stdio.h>

double valore_medio(double a, double b) {
  return (a + b) / 2.0;
}

int main() {
  double a = 1.5, b = 2.5;
```

```
  printf("Valore medio: %.2f\n", valore_medio(a, b));
  return 0;
}
```

5. **Spiegazione del Codice:**

- valore_medio calcola il punto medio tra due numeri reali, che è razionale.
- Dimostra l'uso di razionali tra due punti reali.

Applicazione 5. Irrazionale Moltiplicato per Razionale Non Zero

1. **Teorema:** Il prodotto di un numero irrazionale e un numero razionale non zero è irrazionale.
2. **Applicazione:** Simulazione di moltiplicazioni scalari in fisica.
3. **Spiegazione:** I coefficienti scalari irrazionali sono utilizzati nei calcoli di vettori per simulazioni fisiche.
4. **Codice C:**

```
#include <stdio.h>
#include <math.h>

int main() {
  double irrazionale = sqrt(2);
  double razionale = 3.0;
  double prodotto = irrazionale * razionale;

  printf("Prodotto: %.5f\n", prodotto);
  return 0;
}
```

5. **Spiegazione del Codice:**

- sqrt(2) è un numero irrazionale.
- La moltiplicazione con 3.0 produce un numero irrazionale.

Applicazione 6. Teorema della Somma di Numeri Razionali

1. **Teorema:** La somma di due numeri razionali è sempre un numero razionale.
2. **Applicazione:** Calcolo delle medie nei sistemi di gestione dei database.
3. **Spiegazione:** Nei database, la media di un insieme di valori (somma di razionali divisa per un numero intero) utilizza questo teorema.
4. **Codice C:**

```
#include <stdio.h>


int main() {
   double num1 = 1.25, num2 = 2.75;
   double somma = num1 + num2;


   printf("Somma: %.2f\n", somma);
   return 0;
}
```

5. **Spiegazione del Codice:**

- num1 e num2 sono numeri razionali.
- La loro somma è calcolata e stampata come un numero razionale.

Applicazione 7. Teorema di Esistenza dei Numeri Reali

1. **Teorema:** Ogni intervallo [a,b] con a<b contiene almeno un numero reale.
2. **Applicazione:** Generazione di numeri casuali all'interno di un intervallo.
3. **Spiegazione:** Gli algoritmi di generazione di numeri casuali si basano sulla selezione di un numero reale in un intervallo dato.
4. **Codice C:**

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>


int main() {
   srand(time(0));
   double a = 1.0, b = 5.0;
   double random = a + (rand() / (double)RAND_MAX) * (b - a);


   printf("Numero casuale nell'intervallo [%.2f, %.2f]: %.2f\n", a, b, random);
   return 0;
}
```

5. **Spiegazione del Codice:**

- rand() genera un numero casuale tra 0 e 1.
- Il risultato è scalato per adattarsi all'intervallo [a, b].

Applicazione 8. Teorema di Irrazionalità della Radice Quadrata

1. **Teorema:** La radice quadrata di un numero naturale non quadrato è irrazionale.
2. **Applicazione:** Calcolo della distanza in geometria euclidea.
3. **Spiegazione:** La distanza tra due punti nello spazio può essere irrazionale a causa dell'uso della radice quadrata.
4. **Codice C:**

```
#include <stdio.h>
#include <math.h>

int main() {
   double x1 = 1.0, y1 = 1.0, x2 = 4.0, y2 = 5.0;
   double distanza = sqrt(pow(x2 - x1, 2) + pow(y2 - y1, 2));

   printf("Distanza: %.5f\n", distanza);
   return 0;
}
```

5. **Spiegazione del Codice:**

- La distanza tra (x1, y1) e (x2, y2) è calcolata con il teorema di Pitagora.
- La radice quadrata genera un numero irrazionale, se non è un quadrato perfetto.

Applicazione 9. Regola aurea

1. **Teorema:** Il rapporto aureo φ è un numero irrazionale.
2. **Applicazione:** Generazione di proporzioni estetiche nei layout web.
3. **Spiegazione:** La proporzione aurea viene utilizzata per definire larghezze e altezze visivamente piacevoli.
4. **Codice C:**

```
#include <stdio.h>
#include <math.h>

int main() {
   double phi = (1 + sqrt(5)) / 2;
```

```
    printf("Rapporto aureo: %.5f\n", phi);
    return 0;
}
```

5. **Spiegazione del Codice:**

- Il rapporto aureo è calcolato utilizzando la radice quadrata di 5.
- Questo valore è applicabile in numerosi contesti estetici.

Applicazione 10. Teorema del Valore Intermedio

1. **Teorema:** Se f(x) è continua su un intervallo chiuso [a, b] e f(a)·f(b)<0, allora esiste un c∈(a,b) tale che f(c) = 0.
2. **Applicazione:** Trovare radici di una funzione polinomiale.
3. **Spiegazione:** Questo teorema è alla base del metodo di bisezione per trovare radici di funzioni continue.
4. **Codice C:**

```
#include <stdio.h>
#include <math.h>

double f(double x) {
    return x * x - 2; // Trova la radice di x^2 - 2 = 0
}

double bisection(double a, double b, double tol) {
    double c;
    while ((b - a) >= tol) {
        c = (a + b) / 2;
        if (f(c) == 0.0) break;
        else if (f(c) * f(a) < 0) b = c;
        else a = c;
    }
    return c;
}
```

```
int main() {
    double a = 0, b = 2, tol = 0.00001;
    double radice = bisection(a, b, tol);

    printf("Radice approssimata: %.5f\n", radice);
    return 0;
}
```

5. **Spiegazione del Codice:**

- La funzione f definisce il polinomio.
- La funzione bisection implementa il metodo di bisezione per trovare una radice.
- Il risultato è una radice approssimata.

2. Principio di Induzione Matematica: Applicazioni

Il principio di induzione matematica è un metodo di dimostrazione per asserzioni che dipendono da un numero naturale n. Consiste in due passi:

1. Base: Si dimostra che l'asserzione è vera per n = 1 (o n = 0, a seconda del problema).
2. Passo Induttivo: Si assume che l'asserzione sia vera per n = k (ipotesi induttiva) e si dimostra che è vera per n = k + 1.

Se entrambi i passi sono soddisfatti, l'asserzione è vera per ogni numero naturale n.

Applicazione 11: Somma dei primi n numeri naturali

1. **Enunciato:** Dimostrare che la somma dei primi n numeri naturali è S(n) = n(n+1)/2 è la somma dei primi n numeri naturali.
2. **Applicazioni:** Somma di sequenza di numeri naturali.
3. **Spiegazione:**

   Base: Per n = 1, S(1) =1(1+1)/2 = 1, quindi è vera.

   Passo Induttivo: Supponiamo che S(k) = k(k+1)/2} sia vero. Mostriamo che

   S(k+1) = (k+1)(k+2)/2.

   S(k+1) = S(k) + (k+1) =k(k+1)/2 + (k+1) =k(k+1) + 2(k+1)/2 = (k+1)(k+2)/2.

4. **Codice C:**

```
#include <stdio.h>


int sum_n(int n) {
   return n * (n + 1) / 2;
}


int main() {
   int n = 10; // Numero di elementi da sommare
   printf("Somma dei primi %d numeri naturali: %d\n", n, sum_n(n));
   return 0;
}
```

5. **Spiegazione del Codice:**

- La funzione sum_n implementa la formula matematica n(n+1)/2.
- Il programma calcola la somma per un valore di n dato e la stampa.

Applicazione 12: Prodotto dei primi n numeri naturali (fattoriale)

1. **Enunciato:** Dimostrare che P(n) = n! è il prodotto dei primi n numeri naturali.
2. **Applicazione:** Calcolo del fattoriale.
3. **Spiegazione:**

   Base: Per n = 1, P(1) = 1! = 1, quindi è vera.

   Passo Induttivo: Supponiamo che P(k) = k! sia vero.

   Mostriamo che P(k+1) = (k+1)!. P(k+1)=P(k) (k+1)=k! (k+1).

4. **Codice C:**

```
#include <stdio.h>


int factorial(int n) {
   if (n == 0 || n == 1) return 1;
   return n * factorial(n - 1);
}


int main() {
   int n = 5; // Numero per calcolare il fattoriale
   printf("Fattoriale di %d: %d\n", n, factorial(n));
   return 0;
}
```

5. **Spiegazione del Codice:**

- La funzione factorial calcola il fattoriale utilizzando una chiamata ricorsiva.
- Il programma calcola n! e lo stampa.

Applicazione 13: Somma dei quadrati dei primi n numeri naturali

1. **Enunciato:** Dimostrare che S(n) = n(n+1)(2n+1)/6 è la somma dei quadrati dei primi n numeri naturali.
2. **Applicazione:** Somma dei quadrati dei primi n numeri naturali.
3. **Spiegazione:**

   Base: Per n = 1, S(1) = 1, quindi è vera.

   Passo Induttivo: Assumiamo che S(k) sia vero e dimostriamo per k+1.

4. **Codice C:**

```
#include <stdio.h>


int sum_of_squares(int n) {
   return n * (n + 1) * (2 * n + 1) / 6;
}


int main() {
   int n = 5; // Numero per calcolare la somma dei quadrati
   printf("Somma dei quadrati dei primi %d numeri: %d\n", n, sum_of_squares(n));
   return 0;
}
```

5. **Spiegazione del Codice:**

- La funzione sum_of_squares implementa direttamente la formula.
- Il programma calcola la somma e la stampa.

Applicazione 14: Sequenza di Fibonacci

1. **Enunciato:** Dimostrare che la sequenza di Fibonacci soddisfa la relazione di ricorrenza F(n) = F(n-1) + F(n-2) con F(0) = 0, F(1) = 1.
2. **Applicazioni:** I numeri di Fibonacci trovano applicazioni in vari contesti anche molto diversi tra loro.
3. **Spiegazione:** Ogni numero di Fibonacci è la somma dei due precedenti. Un tipico esempio è nel contesto dei livelli di prezzo in cui un tipico indicatore delle piattaforme di trading è proprio Fibonacci.
4. **Codice C:**

```
#include <stdio.h>


int fibonacci(int n) {
   if (n == 0) return 0;
   if (n == 1) return 1;
   return fibonacci(n - 1) + fibonacci(n - 2);
}


int main() {
```

```
  int n = 10; // Numero di Fibonacci da calcolare
  for (int i = 0; i <= n; i++) {
    printf("F(%d) = %d\n", i, fibonacci(i));
  }
  return 0;
}
```

5. **Spiegazione del Codice:**

- La funzione Fibonacci usa una definizione ricorsiva per calcolare i termini della sequenza.
- Il programma stampa i primi n numeri di Fibonacci.

Applicazione 15: Calcolo di potenze intere positive

1. **Enunciato:** Dimostrare che a^n può essere calcolato ricorsivamente utilizzando la relazione: a^n = a*a^{n-1}, con a^0 = 1.
2. **Applicazione:** Qualsiasi contesto dove sia necessario il calcolo dipotenze intere positive.
3. **Spiegazione:**

   Base: Per n = 0, a^0 = 1, quindi è vera.

   Passo Induttivo: Assumiamo che a^k sia calcolabile per k e dimostriamo che a^{k+1} = a*a^k.

4. **Codice C:**

```
#include <stdio.h>

int power(int a, int n) {
  if (n == 0) return 1;
  return a * power(a, n - 1);
}

int main() {
  int a = 2; // Base
  int n = 5; // Esponente
  printf("%d^%d = %d\n", a, n, power(a, n));
  return 0;
}
```