

Esame di Laboratorio di Fisica Computazionale

16 giugno 2014, ore 10.00

shell scripting

1. Si scriva uno script che genera 10 files denominati `input1`, `input2`,... in cui vengono scritti in due righe diverse il numero del file e, nella seconda riga, un numero intero random.

Mathematica

1. Si disegni nell'intervallo $x \in [0, 3]$ la soluzione dell'equazione differenziale

$$g'(x) = \sin(x) - x^2 g(x), \quad g(0) = 1 \quad (1)$$

2. La soluzione del problema del punto precedente può essere affrontato con la tecnica di Runge-Kutta. Assegnati i valori $(x_0 = 0, y_0 = 1)$ e fissato un intervallo $h = 0.01$, si calcoli una successione di 300 coppie di punti (x_n, y_n) in cui $x_n = x_0 + n h$ e in cui il valore di y è ottenuto secondo il seguente algoritmo. Posta $f(x_n, y_n) = \sin(x_n) - x_n^2 y_n$

$$\begin{aligned} k_1 &= hf(x_n, y_n) \\ k_2 &= hf(x_n + h/2, y_n + k_1/2) \\ k_3 &= hf(x_n + h/2, y_n + k_2/2) \\ k_4 &= hf(x_n + h, y_n + k_3) \\ y_{n+1} &= y_n + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} \end{aligned} \quad (2)$$

Si visualizzi la successione delle coppie di punti e si confronti questo grafico con quello del punto precedente.

3. Si riscriva il seguente sistema, disaccoppiando le due equazioni differenziali.

$$\begin{cases} y_1''(t) = 3y_1(t) + 5y_2(t) \\ y_2''(t) = 2y_1(t) + 4y_2(t) \end{cases} \quad (3)$$

Si risolvano le due equazioni disaccoppiate, senza imporre condizioni al contorno. Si ricavi infine l'espressione delle funzioni $y_{1,2}(t)$.

4. Si calcoli il quinto numero di Bernoulli, noti $B_0 = 1$ e $B_1 = -1/2$, utilizzando la formula ricorsiva

$$\sum_{j=0}^m \binom{m+1}{j} B_j = 0 \quad (4)$$

C++

Si risolva l'esercizio proposto. Per facilitare la correzione, se possibile includere tutto in un unico file sorgente. La sufficienza è raggiunta risolvendo correttamente i primi tre punti.

La sequenza di Thue-Morse

1. Si scriva una classe `Sequence` che rappresenti una sequenza di 16 numeri interi. Tra i membri `protected` si metta un `std::vector` di interi, che conterrà gli elementi della sequenza. Tra i membri `public` si scriva un costruttore che richieda tre parametri interi a, b, c (con valori di default nulli) e generi la sequenza $ai + bi^2 + ci^3$, con $i = 1, 2, \dots, 16$.
2. Si scriva il costruttore di copie ed una funzione `print` che stampi su `cout` la sequenza.
3. Si implementi un overloading dell'operatore "*" (membro della classe) che calcoli il prodotto scalare tra due sequenze. Il prodotto scalare tra $\{a_1, a_2, \dots, a_{16}\}$ e $\{b_1, b_2, \dots, b_{16}\}$ è definito come $a_1b_1 + a_2b_2 + \dots + a_{16}b_{16}$.
4. Si scriva un `main` che istanzi due sequenze con parametri a, b, c diversi, ad esempio $(1, 0, 0)$ e $(0, 0, 1)$. Si controlli, stampando il risultato su `cout`, che il prodotto tra loro sia commutativo.
5. Si scriva una classe `BinarySequence` che erediti pubblicamente da `Sequence`: essa rappresenterà sequenze composte solo da 0 e 1. Il costruttore di default (senza parametri) dovrà generare una sequenza di zeri e uni scelti in modo casuale.
6. Tra i membri pubblici, si implementi una funzione `not` (la negazione logica) che restituisca una `BinarySequence` con uni e zeri invertiti. Si implementi inoltre una funzione membro `cost`, che prenda come parametro una referenza ad un oggetto di tipo `Sequence` s , e restituisca un numero intero k , calcolato come segue. Chiamiamo b la sequenza binaria (l'oggetto che fa la chiamata a `cost`); allora k è la differenza tra il prodotto scalare di s con b e il prodotto scalare di s con la negazione di b . La quantità k è chiamata costo di b su s .
7. Si implementi un costruttore di `BinarySequence` che prenda come parametro un numero intero positivo e costruisca la sua rappresentazione binaria [può tornare utile la funzione `pow(double base, double esponente)` della libreria `<cmath>`, che restituisce la potenza $\text{base}^{\text{esponente}}$].
8. Nel `main` si riempia un `std::vector` di `BinarySequence` corrispondenti alle rappresentazioni binarie di tutti gli interi tra 1 e $2^{16} - 1$. Tra queste, se ne vuole trovare una (chiamiamola t) con la proprietà che il costo di t su qualunque sequenza di quelle generate dal costruttore del punto (1) sia nullo (una condizione equivalente è che sia nullo

il costo sulle tre sequenze i , i^2 e i^3). Per fare questo, si scriva un predicato (una funzione unaria globale che restituisca un `bool`), che identifichi se una sequenza ha questa proprietà. Quindi, per trovare la sequenza cercata tra quelle nel `std::vector`, si utilizzi l'algoritmo

```
std::find_if(it1, it2, pred)
```

che scorre gli elementi tra l'iteratore `it1` (compreso) e l'iteratore `it2` (escluso) e restituisce un iteratore al primo elemento per cui il predicato `pred` è vero. Si stampi la sequenza su `cout`.