Esame di Laboratorio di Fisica Computazionale

2 settembre 2014, ore 10.00

shell scripting

1. Si calcoli, con uno script, la media dei valori della seconda colonna di ciascuno dei dieci files `distribuzione1.dat` ... `distribuzione10.dat`.

Mathematica

1. Si risolva l'equazione differenziale

$$y'(t) = a\,(1 - y(t))\,y(t) \tag{1}$$

ponendo come condizione al condizione al contorno $y(0) = k$. Si disegnino nello stesso grafico, per $t \in [0, 10]$, le tre soluzioni ottenute ponendo $a = 1$ e $k = 0.5, 1, 1.5$.

2. Si definisca una funzione che permette di valutare numericamente l'integrale

$$\begin{aligned} n(s,t,m) \;=\; & \int_0^1 dx\, \frac{1}{x(1-x) + (m^2 - s - t)/(st)} \left[-\log\left(1 - i\varepsilon - m^2 x(1-x)\right) \right. \\ & \left. + \log\left(1 - i\varepsilon - sx(1-x)\right) + \log\left(1 - i\varepsilon - tx(1-x)\right) \right] \end{aligned} \tag{2}$$

come funzione di s, t e m, essendo ε un parametro infinitesimo.

3. Si disegnino le superfici che rappresentano la parte reale e la parte immaginaria di $n(s,t,m)$, avendo posto $m = 125$ e facendo variare $s \in [20000, 40000]$ e $t \in [-2000, 0]$.

4. La soluzione analitica, che chiamiamo $a(s,t,m)$, dell'integrale (2) del secondo punto può essere espressa in termini di funzioni elementari osservando che

$$\begin{aligned} & \int_0^1 dx\, \frac{1}{x(1-x) + \frac{m^2 - s - t}{st}} \log\left(1 - i\varepsilon - vx(1-x)\right) = \\ = \; & \frac{2}{\sqrt{1 + \frac{4(m^2 - s - t)}{st}}} \times \\ & \left\{ Li_2\left(\frac{x_-}{x_- - y}\right) - Li_2\left(\frac{x_+}{x_+ - y}\right) + Li_2\left(\frac{x_-}{y - x_+}\right) - Li_2\left(\frac{x_+}{y - x_-}\right) + \right. \\ & \left. + \log\left(-\frac{x_-}{x_+}\right) \log\left(1 - i\varepsilon - vx_- x_+\right) \right\} \end{aligned} \tag{3}$$

dove

$$x_\pm = \frac{1}{2}\left(1 \pm \sqrt{1 + \frac{4(m^2 - s - t)}{st}}\right) \qquad y = \frac{1}{2}\left(1 + \sqrt{1 - \frac{4(1 - i\varepsilon)}{v}}\right) \tag{4}$$

e Li_2 è la funzione `PolyLog` di peso 2.
Si definisca la funzione $a(s,t,m)$.
(*Facoltativo*) Si confrontino, in un punto a piacere, i valori numerici di $a(s,t,m)$ e di $n(s,t,m)$.

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.

Esercizio

Si vuole abbozzare un piccolo framework per descrivere i rimbalzi di bocce di vario tipo in un biliardo rettangolare.

1. Si scriva una classe `Ball` che rappresenterà una palla. Tra i membri protected si mettano le due componenti del vettore velocità (per semplicità trascureremo ogni altra quantità, come posizione e massa). Tra i membri public si scriva un costruttore che richieda le due componenti della velocità e le assegni. Si utilizzino i parametri di default nel costruttore in modo che in mancanza di dati la palla abbia velocità di modulo 1 diretta lungo l'asse x.

2. Si scriva una funzione `print` che stampi su `cout` le componenti della velocità ed il suo modulo.

3. Si scriva una `enum` globale `Edge` con i valori "`up`", "`right`", "`down`", "`left`". Utilizzeremo `Edge` per descrivere le quattro sponde. Attenzione: alcuni di questi nomi sono presenti anche nella libreria standard sotto il namespace `std`.

4. Tra i membri pubblici di `Ball` si scriva una funzione `bounce` che richieda in ingresso un `Edge` ed esegua il rimbalzo elastico della boccia sulla sponda. Si introducano dei controlli in modo da evitare rimbalzi impossibili (ad esempio, la sponda `up` ha l'effetto di ribaltare la componente y della velocità, ma solo se la velocità iniziale è positiva, altrimenti non ci può essere rimbalzo). La funzione dovrà ritornare un `bool` che indichi se il rimbalzo è stato eseguito. Per poter risolvere il punto 6 dell'esercizio la funzione `bounce` dovrà essere dichiarata virtuale.

5. Si scriva una classe `DissipativeBall` che erediti pubblicamente da `Ball`: essa implementerà rimbalzi in cui l'energia non si conserva. Si metta dunque tra i membri privati un valore reale `delta`, che rappresenterà la frazione di energia cinetica dissipata nell'urto. Il costruttore dovrà richiedere e impostare, oltre alle due componenti della velocità, anche `delta` (selezionare valori di default compatibili con quelli della classe base). Si scriva quindi la funzione `bounce` per questo tipo di urto, ipotizzando che la direzione di uscita della palla sia la stessa che in un urto elastico. (Si faccia attenzione a non replicare codice già scritto: si riutilizzi quanto già scritto per la classe base.)

6. Si scriva una funzione `four_bounces` che richieda un puntatore ad un oggetto di tipo `Ball` e faccia compiere alla boccia un rimbalzo su ogni sponda (in senso orario a partire da

`up`: i rimbalzi effettivamente fatti dipenderanno dalla direzione iniziale della palla). Per verificare il comportamento polimorfico, si scriva un `main` che allochi dinamicamente due palle: una di tipo elastico, l'altra che dissipi il 10% della sua energia ad ogni urto, dotate della stessa velocità iniziale. Si richiami `four_bounces` su entrambe le palle e si verifichi, stampando le velocità finali su `cout`, che il comportamento è quello atteso. (Ricordarsi di liberare la memoria allocata; alternativamente utilizzare degli smart pointers.)

7. Nel `main` si riempia un `std::vector` di puntatori a `DissipativeBall` con 20 palle dissipative (tutte con lo stesso `delta`) con velocità iniziale di modulo 1 e direzione casuale. Si facciano compiere i quattro rimbalzi di `four_bounces` ad ogni boccia, utilizzando l'algoritmo

   ```
   std::for_each(it1, it2, func)
   ```

   che esegue la funzione `func` passandole come parametri ognuno degli elementi compresi tra l'iteratore `it1` (compreso) e `it2` (escluso). Si verifichi, stampando su `cout`, che i moduli delle velocità finali assumono solo tre valori possibili.