Esame di Laboratorio di Fisica Computazionale

23 aprile 2015, ore 9.30

shell scripting

1. Si generi un nuovo file a partire da quello allegato `spese.txt`. Nel nuovo file si aggiunga una colonna che contenga la somma dei valori riportati nelle altre colonne di ciascuna riga.

2. Successivamente i calcoli la somma delle tre colonne e la si aggiunga in fondo.

Mathematica

1. Si risolva il sistema di equazioni

$$\begin{cases} z &= x^2 + y^2 \\ z &= -(x^2+y^2)^{\frac{3}{2}} + 30 \\ z &= -40x - 40y \end{cases}$$

   e si disegnino le 3 superfici associate alle 3 equazioni, con $x \in [-3,3]$ e $y \in [-3,3]$.

2. Si definisca una funzione associata alla soluzione numerica, nell'intervallo $t \in [0,10]$, dell'equazione differenziale

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

   Questa funzione dipende dai parametri a, b e dalla costante k che determina la condizione al condizione al contorno $y(0) = k$.

   Si generi un grafico tridimensionale, facendo variare $t \in [0,10]$ e $a \in [0.5, 1.5]$, avendo fissato $k = b = 1.5$.

   Si generi un secondo grafico tridimensionale, facendo variare $t \in [0,10]$ e $b \in [0.1, 10]$, avendo fissato $a = k = 1.5$.

   Si generi un terzo grafico tridimensionale, facendo variare $t \in [0,10]$ e $k \in [0.5, 1.5]$, avendo fissato $a = b = 1.5$.

3. Si generino 100000 valori casuali x_i compresi tra -5 e 5. Per ciascun valore x_i, si estragga un secondo valore casuale y_i compreso tra 0 e 1. Si valuti la disuguaglianza $y_i < \exp(-x_i^2)$ e si salvi la coppia (x_i, y_i) solo se la disuguaglianza è soddisfatta. Si proceda quindi a disegnare l'insieme dei punti selezionati, utilizzando `ListPlot`. Si confronti la frazione di punti selezionati rispetto a quelli generati con il risultato del rapporto di integrali

$$\frac{\int_{-5}^{5} dx \exp(-x^2)}{\int_{-5}^{5} dx\, 1}$$

   suggerimento: si consideri eventualmente l'utilizzo di `DeleteCases`

4. Si generi una matrice 1000x1000 di numeri casuali compresi tra 0 e 1. Si calcolino gli autovalori di questa matrice. Si trasformi ciascun autovalore z nella coppia $\{Re[z], Im[z]\}$. Si utilizzi `ListPlot` per disegnare gli autovalori, nel piano complesso z.

5. La soluzione del problema del punto 2, assegnando ai parametri i valori $a = b = 0.5$, può essere affrontato con la tecnica di Runge-Kutta. Posto $(x_0 = 0, y_0 = 0.5)$ e fissato un intervallo $h = 0.01$, si calcoli una successione di 1000 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. Data $f(x_n, y_n) = 0.5\,(1 - y_n)\,y_n + 0.5$

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

Si visualizzi la successione delle coppie di punti e si confronti questo grafico con quello corrispondente ottenuto utilizzando la soluzione del punto 2.

6. Il comando `Distribute` permette di implementare la proprietà distributiva rispetto all'addizione di funzioni generiche (p.es. `Distribute`[h[a+b,c]] = h[a,c]+h[b,c]).

   (a) Dati due operatori A e B che non commutano, si sfrutti questo comando per generare in forma espansa tutti i termini fino al secondo ordine del prodotto di operatori $\exp(A)\exp(B)$. (*suggerimento: si utilizzi una funzione* h *di due sole variabili per mantenere l'ordine tra A e B.*)

   (b) Si consideri ora lo sviluppo al secondo ordine della formula di Baker-Campbell-Hausdorf $C = A+B+\frac{1}{2}[A,B]$. Si scrivano, a mano, **formalmente**, i primi tre termini dello sviluppo (fino al secondo ordine) di $\exp(C)$, sempre sfruttando la funzione h per mantenere l'ordinamento. Con `Distribute` si espanda anche questo risultato (può servire in questo caso il simbolo :> al posto di $->$ per indicare la regola di sostituzione).

   (c) Si implementino delle definizioni (oppure si applichino delle sostituzioni) per semplificare le espressioni

   (d) Si sottraggano le espressioni dei punti (a) e (b), mostrando che i termini rimanenti sono di ordine superiore.

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 quattro punti.

Esercizio

Si vuole abbozzare un piccolo framework per descrivere i rimbalzi in un flipper (tra palline e bumpers).

1. Si scriva una classe `Ball` che rappresenterà una palla. Tra i membri private si mettano le due componenti del vettore velocità e la massa (si usino variabili `double`: per semplicità trascureremo le unità di misura). Tra i membri public si scriva un costruttore che richieda come parametri massa e componenti della velocità, con valori di default tutti e tre uguali a 1.

2. Si scrivano due funzioni (membri di `Ball`): una funzione `energy` che restituisca il valore dell'energia cinetica, e una funzione `print` che stampi su `cout` le componenti della velocità e l'energia cinetica (usando `energy`).

3. Si scriva una funzione `change_speed` (anch'essa membro) che prenda come parametro un numero reale λ e riscali di un fattore λ il modulo della velocità, mantenendone invariata la direzione.

4. Nel `main` si istanzi un oggetto di tipo `Ball` di massa 1 e velocità $(1, 1)$, se ne stampino componenti ed energia cinetica, e si verifichi che se si raddoppia la velocità con `change_speed` l'energia quadruplica.

5. Si scriva poi una classe `Bumper` che rappresenterà i respingenti, cioè bersagli su cui la palla può rimbalzare. Nell'urto, la palla inverte il verso del suo moto e aumenta la sua energia cinetica di un fattore μ (cioè $E \mapsto \mu E$). Si ponga μ come membro private e si scriva un opportuno costruttore che prenda un parametro e inizializzi μ.

6. Tra i membri public di `Bumper` si scriva una funzione `bounce`, che prenda come parametro una palla e le faccia compiere l'urto descritto al punto precedente. Tale funzione dovrà essere dichiarata virtuale per risolvere i punti successivi. [Si valuti se passare la palla per copia o per referenza.]

7. Si scriva una classe `ThresholdBumper`, che erediti pubblicamente da `Bumper`. L'urto con questo particolare respingente ha il comportamento usuale (con $\mu = 1.5$ fissato) se l'energia cinetica supera una soglia ϵ, altrimenti è un urto elastico (conserva l'energia). Si ponga ϵ tra i membri private e si scriva un opportuno costruttore che prenda come parametro la soglia e inizializzi sia ϵ che il parametro μ della classe base.

8. Si scriva la funzione `bounce` (override di quella ereditata da `Bumper`) in modo che esegua l'urto con il controllo sulla soglia descritto al punto precedente. (Si riutilizzi il codice già scritto per la funzione della classe base.)

9. Si scriva una funzione globale `flipper` che prenda due parametri: una referenza a `Ball` e un `std::vector` di respingenti. Sapendo che l'obiettivo è quello di realizzare un comportamento polimorfico, si consideri se usare un vettore di referenze, di puntatori, oppure di copie. L'effetto di `flipper` deve essere quello di far eseguire alla palla i rimbalzi con tutti i respingenti nel vettore (in ordine).

10. Nel `main`, istanziare due `std::vector` (di referenze, puntatori o copie, vedi punto precedente). Riempire il primo con oggetti di tipo `Bumper` allocati dinamicamente con parametri μ lanciati a caso tra 1 e 2. Riempire il secondo con oggetti di tipo `ThresholdBumper` (anch'essi dinamici), con parametri ϵ lanciati a caso tra 0 e 5. Per ognuno dei vettori, eseguire il flipper a partire da una palla inizializzata coi valori di default, poi stampare lo stato finale della palla.

    [Per generare numeri reali pseudo-random tra 0 e 1 si può usare `drand48()`.]

    [Ricordarsi di chiamare `delete` su ogni oggetto creato con `new`.]

11. Verificare che lo stato finale della palla dopo il flipper con i `Bumper` non cambia se si inverte l'ordinamento dei respingenti, mentre lo fa con i `ThresholdBumper`.