Esame di Laboratorio di Fisica Computazionale

14 luglio 2016, ore 14.00

shell scripting

Viene fornito un file `X_pt-chi2_report.dat` in cui compaiono diverse colonne.
Si scriva uno script che rimuova tutte le variabili e tutti i parametri delle prime colonne e lasci solo le ultime due colonne. Ulteriormente, lo script deve rimuovere la stringa `X_pt-` presente nella penultima colonna, lasciando solo le ultime tre cifre intere. Ancora, lo script deve aggiungere nel file finale, come prima colonna, il numero di riga elaborata.

Mathematica

1. Si risolva il sistema di equazioni differenziali

$$\begin{cases} x'(t) &= x(t) + y(t) \\ y'(t) &= x(t) - y(t) \end{cases} \tag{1}$$

2. Si consideri lo stesso sistema di equazioni del punto precedente, riscritto in forma matriciale, avendo definito $X(t) = (x(t),\, y(t))$.

$$\dot{X}(t) = A \cdot X(t) \tag{2}$$

$$A = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{3}$$

   Si calcoli la matrice B del cambio di base che rende A diagonale.
   In questa base definiamo $Y(t) = B \cdot X(t) = (a(t),\, b(t))$.
   Si scriva e si risolva il sistema di equazioni differenziali (senza la notazione matriciale!) per le funzioni $a(t)$ e $b(t)$, utilizzando gli autovalori di A.
   Si sostituisca l'espressione di queste funzioni nel vettore $Y(t)$.
   Si ricavi quindi $X(t)$ a partire da $Y(t)$.
   Si verifichi che il risultato coincide con quello del punto precedente.
   facoltativo: Per quanto riguarda le costanti di integrazione, si confrontino le due combinazioni di costanti che si ottengono valutando le due soluzioni $X(t)$ (quella di questo punto e quella del punto precedente) per $t = 0$.

3. Si disegni la superficie definita parametricamente da

$$\begin{aligned} x &= (R + s\cos(t/2))\cos(t) \\ y &= (R + s\cos(t/2))\sin(t) \\ z &= s\sin(t/2) \end{aligned}$$

   con $R = 1$ e con $s \in (-0.5, 0.5),\ t \in (0, 2\pi)$.

4. Si utilizzino i comandi `Table` e `Dt` (derivata totale) per definire due funzioni `grad` e `div` che restituiscono formalmente il gradiente di un campo scalare e la divergenza di un vettore, assumendo che le coordinate spaziali si chiamino (x_1, x_2, x_3).
   Avvertenza: quando si verifica il funzionamento di questi comandi, nel caso della divergenza si utilizzino, come nomi per le componenti del vettore di test, simboli **senza** le parentesi quadre (p.es. `v1, v2, v3`).

5. Si definisca una funzione che calcola il prodotto esterno $\vec{x} \wedge \vec{p}$ tra due vettori utilizzando la matrice

$$\begin{pmatrix} \vec{e}_1 & \vec{e}_2 & \vec{e}_3 \\ x_1 & x_2 & x_3 \\ p_1 & p_2 & p_3 \end{pmatrix}. \tag{4}$$

Se si potessero sfruttare i versori $\vec{e}_i$, il prodotto esterno sarebbe dato dal determinante della matrice.
Rinunciando ai versori, si proceda calcolando i minori di questa matrice con il comando `Minors`; si selezioni la riga dei minori che corrisponde allo sviluppo del determinante lungo la prima riga della matrice e si applichino i riordinamenti e i cambi di segno opportuni.

6. Utilizzando la funzione del punto precedente, si definisca la funzione rotore di un campo vettoriale.
   A questo scopo si introduca un vettore fittizio `dd = {ddx[1],ddx[2],ddx[3]}` ; una volta calcolato il prodotto esterno tra `dd` e il campo vettoriale generico, si esegua la sostituzione che rimpiazza il prodotto generico `ddx[i_] v_` con la derivata totale dell'elemento `v` rispetto alla coordinata `x[i]` di indice `i`.
   Si verifichino le due seguenti identità:

$$\vec{\nabla} \cdot (\vec{\nabla} \wedge \vec{A}) = 0$$
$$\vec{\nabla} \wedge (\vec{\nabla} f) = 0$$

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

1. Si scriva una classe `Auto` (attenzione alla maiuscola), che rappresenterà una automobilina elettrica. Si mettano tra i membri *private* la massa dell'auto e la forza che il motore elettrico esercita. Si scriva un opportuno costruttore che inizializzi questi parametri, con *valore di default* 1 per la massa (tralasceremo per semplicità le unità di misura).

2. Si scriva il costruttore di copie.

3. Si scriva una *access function* per leggere la massa e una funzione che restituisca l'accelerazione dell'auto.

4. Si scriva una classe `Checker`, che implementerà un meccanismo di controllo per la produzione delle auto, che devono avere una massa uguale a 1 (entro una soglia di tolleranza). Tra i membri *private* si metta la tolleranza, e si scriva un opportuno costruttore che la imposti. Si scriva quindi una funzione membro `check`, che prenda un'auto e controlli (restituendo il valore di verità) se essa ha le caratteristiche volute. Si vuole che il comportamento di questa funzione sia polimorfico: si faccia in modo che lo sia.

5. Si scriva una classe `CheckerAcc`, che erediti pubblicamente da `Checker`. Essa implementerà un meccanismo di controllo più stringente, che richieda alle auto due specifiche: (a) la stessa implementata al punto 4, ma con tolleranza fissata a 0.05; (b) l'auto deve avere un'accelerazione superiore ad una soglia minima. Si metta tale soglia tra i membri privati. Si scrivano il costruttore e l'*overriding* della funzione `check`, che implementi il comportamento voluto. Si abbia cura di riutilizzare il codice della funzione `check` già scritta per la base (riuso non significa cut-and-paste).

6. Nel `main` si istanzi un `Checker` con tolleranza 0.03 e un `CheckerAcc` con accelerazione minima 2. Si istanzi quindi un oggetto di tipo `Auto`, i cui parametri siano scelti in modo che esso passi il test del `CheckerAcc` ma non quello del `Checker`. Si verifichi che questo accade, scrivendo il codice in modo che venga usato il comportamento polimorfico, non la distinzione statica tra i tipi.

7. Si scriva una funzione globale `fastest`, che prenda due iteratori a un `std::vector` contenente oggetti di tipo `Auto`. Questa funzione dovrà restituire l'auto con accelerazione maggiore compresa tra i due iteratori (il primo compreso, il secondo escluso). Si implementi questa funzione eseguendo esplicitamente un ciclo su tutti gli elementi da considerare.

8. [facoltativo] Si riscriva la funzione del punto precedente in versione *template*, in modo che accetti iteratori in altri tipi di container (come `deque` o `list`).

9. Si scriva una funzione `fastest_algo`, che riproduca il comportamento di `fastest`, ma sfruttando il seguente algoritmo della libreria standard:

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

   che compara a due a due gli elementi compresi tra gli iteratori `it1` e `it2` usando il predicato binario `pred`, e restituisce un iteratore all'elemento che realizza il massimo. Il predicato andrà implementato a tale scopo.

10. Nel main, si riempia un vettore di 1000 auto, generate con massa random tra 0.5 e 1.5 e con forza random tra 0 e 1 (con la misura piatta). Si controlli che l'auto più veloce tra queste ha una massa di poco più grande di 0.5.

11. Si risponda brevemente alla seguenti domande. Se invece di usare un `std::vector` si volesse usare `std::set`, quali modifiche al codice andrebbero apportate? Quale sarebbe il modo più rapido di implementare la funzione `fastest`?