Esame di Laboratorio di Fisica Computazionale

15 luglio 2015, ore 9.30

shell scripting

1. Si corregga con uno script il testo nel file `brano.tex`, sostituendo alla parola "aumento" la parola "calo".

Mathematica

1. **Serie di Fibonacci**
   Si confrontino i tempi di esecuzione (il comando è `Timing`) dei seguenti algoritmi per calcolare il 66-esimo elemento della successione di Fibonacci:

$$f(n) = f(n-1) + f(n-2), \quad f(0) = 0, \quad f(1) = 1 \tag{1}$$

   (a) Si segua la definizione ricorsiva, salvando in memoria il valore della ricorsione precedente.

   (b) Si formuli in termini matriciali la relazione ricorsiva, osservando che le due equazioni

$$\begin{aligned} f(n) &= f(n-1) + f(n-2) \\ f(n-1) &= f(n-1) \end{aligned}$$

   possono prendere la forma

$$\begin{pmatrix} f(n) \\ f(n-1) \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} f(n-1) \\ f(n-2) \end{pmatrix} \tag{2}$$

   Dopo aver risolto analiticamente la ricorsione in Eq.2 (a mano va bene), si calcoli la potenza opportuna della matrice dei coefficienti con il comando `MatrixPower`.

   (c) Come nel punto precedente, ma si scriva una procedura equivalente a `MatrixPower`, utilizzando i comandi `Table, Dot` e `Apply`.

   (d) Si utilizzi, per il calcolo di una potenza, l'algoritmo ricorsivo seguente:

$$x^n = \begin{cases} (x^2)^{\frac{n}{2}} & \text{con } n \text{ pari} \\ x(x^2)^{\frac{n-1}{2}} & \text{con } n \text{ dispari} \end{cases} \tag{3}$$

   Si applichi questo algoritmo generale al prodotto di matrici.

   (e) Si consideri la definizione ricorsiva, senza salvare in memoria il valore della ricorsione precedente; in questo caso si stimi il tempo richiesto per valutare $f(66)$, estrapolando dai risultati di Timing per $n = 30, 31, 32, 33$.

2. Si risolva la seguente equazione differenziale

$$\begin{aligned} y'(x) &= y(x)\,(y(x)+1)\,x \\ y(0) &= b \end{aligned} \tag{4}$$

   Si disegni la soluzione, in un grafico 3D, con $x \in [-5, 5]$ e con il parametro $b \in [-2, 2]$.

3. Si calcoli, come funzione dei paramteri α, β, γ e come funzione della variabile z, il seguente integrale:

$$h(\alpha, \beta, \gamma, z) = \frac{\Gamma[c]}{\Gamma[b]\Gamma[c-b]} \int_0^1 dx t^{b-1}(1-t)^{c-b-1}(1-tz)^{-a} \tag{5}$$

(a) Si espanda $h(\alpha, \beta, \gamma, z)$ in potenze di z attorno a $z = 0$ fino al quinto ordine.

(b) Si costruisca il rapporto tra la funzione h esatta e la sua espansione e si disegni il rapporto, per $\alpha = 1, \beta = 2$ fissati, con $c \in [2.5, 4.5]$ e $z \in [-2, 2]$.

(c) Si risolva numericamente per quale valore di z l'espansione approssima la funzione $h(1, 2, 3, z)$ al 10%, ovvero per quale valore di z il rapporto tra la funzione e la sua espansione è pari a 0.9. (*suggerimento: si utilizzi il comando* `FindRoot` *per individuare questo punto*)

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 gestire il processo produttivo in una industria che produce molle.

1. Si scriva una classe `Spring`, che rappresenterà una molla. Si mettano tra i membri private la costante elastica k e il carico di rottura della molla (cioè la massima forza sostenibile). Si scriva un opportuno costruttore che inizializzi questi parametri, con valore di default $k = 1$ (tralasceremo per semplicità le unità di misura).

2. Si scriva il costruttore di copie, e due access functions per leggere i due membri privati.

3. Si scriva una classe `Checker`, che implementerà un meccanismo di controllo per le molle: si vogliono scartare i pezzi che hanno costante elastica diversa da 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 una molla e controlli (restituendo il valore di verità) se essa ha le caratteristiche volute. Per i punti successivi, si vuole che il comportamento di questa funzione sia polimorfico: si faccia in modo che lo sia.

4. Si scriva una classe `loadChecker`, che erediti pubblicamente da `Checker`. Essa implementerà un meccanismo di controllo più stringente, che richieda alle molle due specifiche: (a) la stessa implementata al punto 3, ma con tolleranza fissata a 0.1; (b) la molla deve poter poter sostenere una certa compressione (variazione di lunghezza rispetto alla posizione a riposo). Si metta tale compressione tra i membri privati; si scrivano il costruttore e la funzione `check`, che implementi il comportamento voluto.

5. Nel `main` si istanzi un oggetto di tipo `Spring`, un `Checker` e un `loadChecker`, i cui parametri siano scelti in modo che la molla passi il test del `Checker` ma non quello del `loadChecker`.

6. Si scriva l'overloading dell'operatore () per `Checker`, riusando il codice della funzione `check` (riuso non significa cut-and-paste). Si controlli, nel `main`, che il comportamento di tale operatore per la classe derivata è automaticamente quello atteso. [Si possono completare i punti seguenti anche senza aver fatto questo.]

7. Si scriva una classe `Factory` che rappresenterà il processo produttivo. Questa dovrà avere, tra i membri private, un `Checker` (si consideri se tenere una copia o una referenza, volendo ottenere un comportamento polimorfico); si scriva il costruttore.

8. Si scriva, tra i membri public, una funzione `produce`, che dovrà generare dinamicamente una nuova molla, e restituirla (si consideri attentamente quale tipo di ritorno usare). In particolare, la molla dovrà essere prodotta con costante elastica random compresa tra 0.8 e 1.2, e con carico di rottura pari a 10, e dovrà essere controllata con l'oggetto `Checker` privato. (Per ogni chiamata alla funzione `produce` dovrà essere restituita una molla con le caratteristiche volute.)

9. Nel `main`, istanziare un `Checker c1` con tolleranza 0.1 e un `loadChecker c2` con compressione 10. Si producano 1 000 molle con una industria che implementa `c1` e si verifichi che circa la metà di queste passano anche il controllo `c2`. Viceversa, si verifichi che tutte le molle prodotte con una industria che implementa `c2` passano il controllo `c1`.

10. Nel `main`, si istanzi un `vector` di 20 puntatori a oggetti di tipo `Spring`, generati da una industria che implementa `c1`. Si stampi il massimo valore della costante elastica in questo vettore, usando l'algoritmo

    ```
    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.