Esame di Metodi Computazionali della Fisica, modulo I

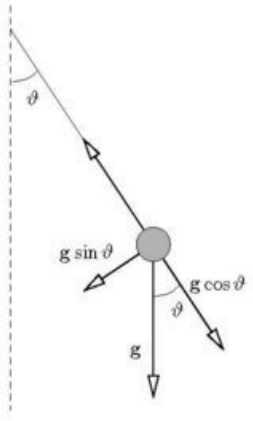
14 gennaio 2009, ore 16.00
Tempo previsto per la prova: 2 ore

Mathematica

1. **Moto armonico**
   Sia dato un sistema unidimensionale in cui un corpo puntiforme di massa m è soggetto ad un potenziale armonico $V(x) = 1/2mx^2$.

   - Si calcoli la forza cui è soggetto il corpo, in funzione della sua posizione.
   - Si calcoli analiticamente il moto del corpo (posizione in funzione del tempo) prendendo come condizioni iniziali $x(t = 0) = 2, \frac{dx}{dt}(t = 0) = 0$.
   - Si disegnino sullo stesso grafico la posizione e la velocità del corpo in funzione del tempo, nell'intervallo $0 \leq t \leq 10$.

2. **Moto di un pendolo**

   Si consideri un pendolo di lunghezza l e di massa m e si chiami θ l'ampiezza delle oscillazioni rispetto alla verticale (vedi figura). Il corpo è soggetto alla forza di gravità. La componente della forza di gravità lungo la direzione perpendicolare al filo del pendolo agisce come forza di richiamo verso la posizione di riposo ($\theta = 0$).

   - Si scriva la forza di richiamo in funzione dell'ampiezza di oscillazione.
   - Si calcoli numericamente il moto del pendolo (ampiezza in funzione del tempo), ponendo per semplicità $g = 1$, prendendo come condizioni iniziali $\theta(t = 0) = 1, \frac{d\theta}{dt}(t = 0) = 0$ e come intervallo di riferimento $0 \leq t \leq 10$.
   - Si disegnino sullo stesso grafico l'ampiezza e la velocità di oscillazione del pendolo in funzione del tempo, nell'intervallo $0 \leq t \leq 10$.
   - Si confrontino il moto del pendolo e il moto armonico del primo problema.
     Si ripeta il confronto nel caso in cui le condizioni iniziali siano per entrambi i sistemi: $x(t = 0) = \theta(t = 0) = 0.1$.

3. **Calcolo del lavoro** Nel caso dell'oscillatore armonico descritto nel primo punto, si calcoli il lavoro compiuto dalla forza di richiamo per riportare il corpo in esame dal punto di massima ampiezza all'origine delle coordinate.

C++

Nell'esercizio di C^{++} è prevista l'implementazione di una classe *Funzione* mediante la sua rappresentazione in *serie*. Una generica funzione f può essere rappresentata mediante serie nella forma

$$f(x) = \sum_{n=0}^{\infty} f_{x_0}(n)(x - x_0)^n$$

dove x_0 è il punto in cui sviluppare la funzione mentre $f_{x_0}(n)$ è caratteristica della funzione. Ad esempio, nel caso dell'esponenziale

$$f_{x_0}(n) = e^{x_0}/n!$$

mentre per il seno

$$f_{x_0}(n) = \begin{cases} (-1)^{\frac{n-1}{2}} cos(x_0)/n! & n\ dispari \\ (-1)^{\frac{n}{2}} sin(x_0)/n! & n\ pari \end{cases} .$$

Sarà dunque necessario implementare una classe madre che rappresenti una generica funzione (prima parte), e derivare le specifiche funzioni attraverso il meccanismo dell'ereditarietà in C^{++} (seconda parte). Fare attenzione al fatto che **la serie NON deve essere memorizzata** all'interno della clase, ma dovrà essere implementata una funzione che calcoli l'$n-esimo$ coefficiente ad n fissato. Ogni parte è suddivisa in punti i quali possono essere obbligatori o facoltativi. I punti obbligatori debbono essere svolti in sequenza per il corretto funzionamento della libreria mentre i punti facoltativi (se non esplicitamente indicato) possono essere svolti in qualsiasi ordine. Il codice deve essere **correttamente compilabile** e accompagnato dal rispettivo **Makefile** funzionante (se il Makefile è assente saranno decurtati dei punti). Non saranno accettati codici non compilabili (abbiate cura di commentare la parte di codice che non riuscite sviluppare. In caso di indecisione sul voto tale parte sarà comunque parzialmente valutata). Il codice deve rispondere correttamente alle richieste previste nel main. In sede di valutazione il codice sarà comunque testato per verificarne l'affidabilità.

1 I Parte: classe Funzione

1.1 Definizione della classe Funzione

(a) Costruttori/Distruttori:

(1 - Obb) Implementare il costruttore per la classe Funzione che accetti un **double** in ingresso rappresentante il punto in cui sviluppare la funzione e settato a zero per default (per semplicità dichiararlo **public**).

(2 - Obb) Implementare il distruttore

(3 - Fac) Aggiungere una variabile **string** come membro **private** dichiarata **const**, rappresentante il nome della funzione. Aggiungere una funzione che restituisca una referenza costante a tale variabile.

(4 - Fac) È necessario aver svolto il punto (3). Modificare il costruttore del punto (1) in modo tale che accetti in ingresso un **double** ed una variabile **string** per settare il nome della funzione.

(5 - Fac) È necessario aver svolto il punto (3). Modificare il costruttore del punto (1) (o implementare un nuovo costruttore nel caso sia stato già svolto il punto (4)) che accetti in ingresso un **double** ed una stringa *C-Style* **char *** per settare il nome della funzione.

TIP: La classe **string** si trova nella libreria **string**.

TIP: I costruttore per la classe **string** sono due: **string(const string &s)** e **string(const char *s)**.

(b) Calcolo dei coefficienti $f(n)$:

(1 - Obb) Implementare una funzione **protected** per il calcolo del fattoriale (per problemi di dimensione massima di un intero, usare all'interno della funzione solamente dei double e far restituire un double alla funzione anziché un int).

(2 - Obb) Definire una funzione **Coeff**, **virtual pura** per il calcolo dell'$n-esimo$ coefficiente (con n intero come paramtro) $f_{x_0}(n)$.

(3 - Obb) Overloadare l'operatore [ ] affinché restituisca l'$n-esimo$ coefficiente (con n intero) sfruttando la funzione definita nel punto (2).

(4 - Fac) Implementare una funzione che restituisca il valore di **f(x)** a fissato valore in ingresso **x**. A tal scopo, definire in testa all'header la pseudo variabile

$$\#define \;\; MAX \;\; 200$$

ed utilizzare solamente i primi **MAX** coefficienti per calcolare il valore della funzione.

(5 - Fac) Overloadare l'operatore **( )** affinché restituisca il valore della funzione **f(x)** a fissato valore di **x**, sfruttando la funzione definita nel punto (4).

TIP: La funzione fattoriale deve essere dichiarata protected affinché possa essere ereditata dalle classi figlie ma non visibile dall'esterno.

TIP: Notare bene che in questa classe l'*unica* funzione virtual è la funzione che calcola i coefficienti.

2 II Parte: classe Esponenziale/Seno

(c - Obb) Funzione Esponenziale:

(1) Implementare il costruttore per la classe derivata esponenziale. Se si è sviluppato il punto (a-3) o (a-4), inizializzare la stringa della classe madre a *Esponenziale*.

(2) Implementare il distruttore.

(3) Implementare la funzione **Coeff** (**virtual** nella classe madre) per l'esponenziale.

TIP:

$$f_{x_0}(n) = e^{x_0}/n!$$

(d - Fac) Funzione Seno:

(1) Implementare il costruttore per la classe derivata seno. Se si è sviluppato il punto (a-3) o (a-4), inizializzare la stringa della classe madre a *Seno*.

(2) Implementare il distruttore.

(3) Implementare la funzione **Coeff** (**virtual** nella classe madre) per il seno.

TIP:

$$f_{x_0}(n) = \begin{cases} (-1)^{\frac{n-1}{2}} cos(x_0)/n! & n\ dispari \\ (-1)^{\frac{n}{2}} sin(x_0)/n! & n\ pari \end{cases} .$$

3 III Parte: Main e Integrazione

(e - Fac) Funzione Integrazione:

(1) Implementare la funzione **Integrate_Trapezi** (esterna alla classe Funzione e alle sue derivate, oppure friend nella classe madre) che dato un oggetto di tipo Funzione e gli estremi di integrazione ne calcola l'integrale definito numerico con il metodo dei trapezi.

TIP: Definire in testa all'header la pseudo variabile

$$\#define \;\; STEP \;\; 0.01$$

ed usare il metodo dei trapezi per il calcolo dell'integrale, usando come passo **STEP**.

(2) Implementare la funzione **Integrate_Primitiva** (esterna alla classe Funzione e alle sue derivate, oppure friend nella classe madre) che dato un oggetto di tipo Funzione e gli estremi di integrazione ne calcola l'integrale definito numerico con il metodo della primitiva.

TIP: Poiché abbiamo rappresentato la funzione come serie, possiamo integrare membro a membro ed ottenere

$$\int_a^b dx \; f(x) = \sum_{i=0}^{\infty} f_{x_0}(n) \frac{(x - x_0)^{n+1}}{n+1}.$$

La funzione dovrà dunque sommare in modo opportuno i coefficienti modificati della funzione.

(f - Obb) Main:

(1) Creare un file main che dia le seguenti informazioni (se le funzioni sono state opportunamente implementate)

Nome Funzione	x_0	Coeff. $n = 3$	$f(x)$ per $x = 0.3$	$\int_0^{2\pi} f(x)dx$, $(\pi = acos(-1))$
Esponenziale	0	$1/3! \approx 0.166667$	1.34986	≈ 534.4
Esponenziale	2.7	2.47996	1.34986	≈ 534.4
Seno	0	$-1/3! \approx -0.166667$	0.29552	≈ 0
Seno	1.9	0.0538816	0.29552	≈ 0