Esame di Laboratorio di Fisica Computazionale

15 giugno 2018, ore 14.00

1 Mathematica

1. Si determinino le equazioni delle rette passanti per il punto $P = (-2, 4)$ e tangenti alla circonferenza Γ di raggio $R = 3$ centrata nel punto $C = (2, 3)$. Si calcolino le coordinate dei punti di tangenza.

2. Si risolva il sistema di equazioni differenziali, per generiche condizioni al contorno.

$$\begin{cases} \dot{x} &= x - y + 10\cos(t) \\ \dot{y} &= 4x + y \end{cases} \tag{1}$$

3. La trasformata di Laplace $F(s)$ di una funzione $f(t)$ è definita

$$F(s) = \int_0^\infty dt\, f(t) \exp(-st) \tag{2}$$

e gode della seguente proprietà, estremamente utile nella soluzione di (sistemi di) equazioni differenziali.

$$\int_0^\infty dt\, f'(t) \exp(-st) = sF(s) - f(0) \tag{3}$$

   - Si scriva una lista di regole di sostituzione con cui si rimpiazzano **simbolicamente** le funzioni incognite $(x(t), y(t))$ presenti in equazione (1) e le loro derivate con le corrispondenti trasformate di Laplace $(X(s), Y(s))$, sfruttando anche l'equazione (3). Si scriva una regola di sostituzione con cui si sostituisce la funzione $\cos(t)$ con la sua trasformata di Laplace.
   - Si applichino queste regole al sistema di equazione (1); si risolva quindi il sistema trasformato come un sistema algebrico rispetto alle incognite $(X(s), Y(s))$.
   - Si utilizzi il comando `InverseLaplaceTransform` per ricavare l'espressione delle funzioni incognite iniziali $(x(t), y(t))$, a partire dal risultato ottenuto per $(X(s), Y(s))$.
   - Si determini la relazione tra le costanti di integrazioni della soluzione ottenuta nel punto precedente e i valori delle funzioni incongnite per $t = 0$, ovvero $(x(0), y(0))$.

4. Il polinomio di grado 5 $p(x) = \sum_{i=0}^{5} a_i x^i$ assume i valori $(58, 337, 1264, 3619, 8662, 18253)$ quando viene valutato rispettivamente per $x = 2, 3, 4, 5, 6, 7$. Si determinino i coefficienti a_i.

5. La seguente formula dovuta a Newton permette la rappresentazione di un polinomio $q(x)$ di grado R tramite i valori che esso assume nei punti $y_j,\ j \in [0, R-1]$:

$$\begin{aligned} q(x) &= \sum_{i=0}^{R} a_i \prod_{j=0}^{i-1} (x - y_j) \\ &= a_0 + (x - y_0)\,(a_1 + (x - y_1)(a_2 + (x - y_2)(...))) \end{aligned} \tag{4}$$

Si calcoli l'espressione analitica dei coefficienti a_i, scrivendo un sistema di equazioni in cui si uguaglia il valore $\bar{q}(y_i)$ (si utilizzi un simbolo $\bar{q}$ distinto dal simbolo q che abbrevia il polinomio) con l'espressione data dall'equazione (4) valutata per $x = y_i$, avendo posto $R = 5$.

2 C++: Esercizio

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 scrivere un semplice DSP (*digital signal processor*), cioè un infrastruttura che rappresenti segnali digitali e trasformazioni sui segnali (che chiamiamo "effetti").

1. Si scriva una classe `DSP`, che rappresenterà un buffer contenente un segnale, cioè un elenco di valori di tipo `double`, che si immaginano campionati a tempi discreti successivi. Come buffer si usi il container `std::deque`. Tale buffer avrà una lunghezza massima consentita (numero massimo di valori contenuti), fissata al momento della costruzione. Si mettano tra i membri privati il buffer e la lunghezza massima.

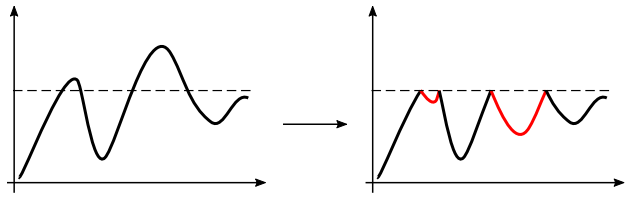
2. Si scriva un costruttore di default che inizializzi il DSP con il segnale vuoto (cioè senza valori campionati) e lunghezza massima 1024. Si scriva un altro costruttore, che prenda come argomenti due iteratori dentro una struttura dello stesso tipo del buffer, e inizializzi il buffer privato con i valori compresi tra i due iteratori (il primo compreso, il secondo escluso). La lunghezza massima dovrà essere inizializzata pari a questo range. Bonus: si usi la sintassi della lista di inizializzazione, scrivendo costruttori con corpo vuoto {}.

3. Si scriva una *access function* che restituisca il buffer, ma solo in lettura. Si scriva inoltre una funzione `print` che stampi i valori del buffer su `std::cout`.

4. Tra i membri pubblici, si scriva una funzione membro `push`, che prenda come argomento un `double` e lo aggiunga con `push_back` in coda al buffer. Se il buffer ha già raggiunto la lunghezza massima, si dovrà anche eliminare l'elemento in testa al buffer (è il comportamento dei buffer "circolari"). Si commenti brevemente sulla utilità di utilizzare, per il buffer, un `std::deque` invece che un `std::vector`.

5. Si scriva una classe `Effect`, che rappresenterà una operazione da eseguire su ogni singolo valore del segnale. In particolare, ogni oggetto di tipo `Effect` agirà sul segnale moltiplicandolo per un fattore di riscalamento. Si metta tale fattore tra i membri privati e si scriva un opportuno costruttore che lo inizializzi. Si scriva inoltre una funzione membro `execute`, che prenda come argomento un valore `double` e restituisca il valore riscalato. Si vuole che questa funzione abbia comportamento polimorfico.

6. Si scriva una classe `Folder`, derivata di `Effect`, che rappresenterà una operazione di ripiegamento: la parte di segnale con valore assoluto maggiore di una soglia fissata verrà riflessa rispetto al valore di soglia (si veda la figura). Il fattore di riscalamento in un `Folder` è fissato a 2, e il riscalamento viene eseguito prima del ripiegamento. Si metta la soglia tra i membri privati e si scriva un opportuno costruttore che la imposti.



7. Si scriva l'*override* della funzione `execute`, che implementi il comportamento descritto al punto precedente. Si abbia cura di *riutilizzare* il codice già scritto nella classe base, tramite chiamata a funzione.

8. Tra i membri pubblici della classe `DSP` si scriva una funzione `process`, che prenda come argomento un puntatore a un effetto e lo utilizzi per trasformare ogni valore del segnale. Si usi un *range-based for loop*.

9. Nel `main` si istanzi un oggetto di tipo `DSP` con il costruttore di default, e lo si riempia (usando il membro `push`) con i valori

$$x_i = \sin(\pi i/256), \quad i = 0, \ldots, 1023.$$

   Si processi quindi il segnale facendolo passare due volte da un `Folder` con soglia 1.3, e poi da un `Effect` che riporti il segnale a essere compreso tra -1 e 1.

10. Si scriva una funzione globale `peak`, che prenda un un `std::deque<double>` e restituisca il massimo valore assoluto assunto dal segnale. Si usi il seguente algoritmo della STL:

    `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. Nel `main` si verifichi che il picco del segnale sia minore o uguale a 1.

11. Si renda *generico* il codice del punto precedente, in modo che la funzione `peak` possa essere chiamata passandole qualunque tipo di sequenza (come ad esempio `std::vector<double>` o `std::list<int>`).

3 Python

Il comando Unix:
`find   -size +1M -ls > all_big_files.txt`
genera il file di testo "all_big_files.txt" in cui ad ogni riga sono contenute varie informazioni su ogni file che nelle propria $HOME supera la taglia di 1 Mbyte. Si scriva uno script in Python che esegua questo comando e che dal file all_big_files.txt estragga solo le linee relative a tutti i files che terminano col suffisso .txt e che infine scriva queste linee complete in un file di testo il cui nome passato direttamente da riga di comando.