Corso di python novembre-dicembre 2020 - ISM
Fulvio Paleari – fulvio.paleari@mlib.ism.cnr

---

1. Introduzione a python

a. Utilizzando `numpy`, generate una matrice 3×3 contentente elementi casuali compresi fra -1 e 1 e trovatene il determinante utilizzando la regola di Sarruss. Fate una funzione per generare la matrice e un'altra per il calcolo del determinante. Controllate il risultato utilizzando il metodo di `numpy` per trovare il determinante di una matrice.

b. Considerando la matrice 3×3 costruita precedentemente, calcolatene autovalori e autovettori con gli opportuni metodi `numpy`, dopodiché visualizzatene lo spettro usando `matplotlib.pyplot` secondo la seguente formula:

$$S(\omega) = \mathrm{Im}\left\{ \sum_{i=1}^{3} \frac{|\mathbf{e}_i|^2}{\omega - E_i - \mathrm{i}\eta} \right\},$$

dove $\mathbf{e}_i$ è un autovettore e E_i il corrispondente autovalore della matrice 3×3, mentre $\eta > 0$ rappresenta un parametro di *linewidth* (molto piccolo rispetto alla scala di ω considerata) introdotto *ad hoc*.

c. La sequenza di Fibonacci è definita dalla seguente relazione ricorsiva:

$$F_n = F_{n-1} + F_{n-2} \qquad \text{con } F_1 = 1 \; e \; F_2 = 1.$$

Quindi i primi 12 termini saranno:

$$F_1 = 1,\ F_2 = 1,\ F_3 = 2,\ F_4 = 3,\ F_5 = 5,\ F_6 = 8,\ F_7 = 13,\ F_8 = 21,\ F_9 = 34,\ F_{10} = 55,\ F_{11} = 89,\ F_{12} = 144.$$

Il dodicesimo termine, F_{12}, è il primo termine formato da tre cifre.
Qual è l'indice del primo termine della sequenza di Fibonacci formato da 10 cifre? Potete usare funzioni, cicli `for`, `if` e `numpy` array.

[Questa è una versione del problema 25 di Project Euler]

d. Utilizzate `numpy.loadtxt` per caricare il file `testo.dat` (fornendo il corretto *keyword argument* per far funzionare `loadtxt` con caratteri invece che con numeri). A questo punto, costruite una funzione che conti quante vocali sono presenti nel testo, sia in totale (N_{tot}), sia per singola vocale (N_a, N_e, N_i, ...). Rappresentate i risultati (es. N_a/N_{tot},...) in un istogramma.

3. Visualizzazione dati

a. Costruite una funzione $y = \sin(s_i * \varphi)$ nel dominio $\varphi \in [0, 2\pi]$ dove gli s_i sono i fattori di scala seguenti $s = \{1, 2, 3, 4, 5\}$. Fate il plot della funzione al variare dei fattori di scala, e personalizzatelo definendo limiti, nomi e valori degli assi, colori, tipi di linea, etc. Ripetete l'esercizio, ma questa volta facendo i plot in coordinate polari utilizzando le opzioni messe a disposizione da `pyplot`.

b. Fate il plot di una distribuzione gaussiana normalizzata $G(x; \mu, \sigma)$ con media $\mu = 3$ e deviazione standard $\sigma = 0.25$. Usate `numpy` o `scipy` oppure costruite una vostra funzione. Valutate il punto x in cui l'integrale a partire da $-\infty$ è pari a 0.7, cioè:

$$x \text{ t.c. } \int_{-\infty}^{x} \mathrm{d}x' \; G(x'; \mu, \sigma) = 0.7,$$

e colorate l'area corrispondente.

[Esercizio preso dalle lezioni di python tenute all'Università di Milano-Bicocca da Matteo Bonetti]

4. Analisi dati

a. Il governo pubblica quotidianamente i dati relativi all'andamento nazionale dell'epidemia da COVID-19 in un repository github pubblicamente accessibile. I dati sono in formato `csv` e sono accessibili a questo indirizzo.
Scaricate il database `dpc-covid19-ita-andamento-nazione.csv` dal sito (potete cliccare sul tasto `Raw`, che vi porta ad una pagina web contenente unicamente i dati non formattati, copiarne l'indirizzo e da terminale digitare `wget indirizzo/pagina/web` per ottenere il database).
A questo punto, aprite il database in python utilizzando `pandas`, e producete un plot dell'andamento della variabile `totale_positivi` dal 24 febbraio (data di inizio presa dati) ad oggi.
Se volete, sperimentate con `pandas` e `pyplot` per ottenere altri plot ed abbozzare un'analisi dei dati.

b. Supponiamo di aver fatto delle misure in funzione del tempo per un certo fenomeno, e supponiamo di conoscere che la legge fisica che governa questo fenomeno è quella di un oscillatore smorzato:

$$F(t) = I_0 \mathrm{e}^{-\gamma t} \cos(\Omega t + \varphi) + C$$

Considerate il file `damped_osc_low_noise.dat`. Questo contiene due colonne: la prima è il tempo, la seconda il valore di $F(t)$ misurato sperimentalmente. Caricatelo con `numpy.loadtxt` ed eseguite – utilizzando `scipy` – il fit dei dati secondo la funzione indicata. Stimate i valori del fattore di smorzamento γ e della frequenza di risonanza Ω e fate un plot dei risultati sperimentali e del fit.
Considerate ora i file `damped_osc_high_noise.dat` e `damped_osc_very_high_noise.dat`. Questi contengono misurazioni relative allo stesso esperimento, ma i dati sono sensibilmente più rumorosi. Ripetete la procedura di fit anche per questi dati (magari importando tutti i file `*.dat` utilizzando `glob`) e fate un plot delle varie stime di Ω e γ. Provate a estrapolare il valore reale di queste grandezze.

5. Programmazione orientata ad oggetti

a. È possibile stimare il valore di π attraverso una simulazione MonteCarlo. Per farlo si campiona l'area di un quadrato di lato uguale a 2, ovvero si estraggono attraverso un generatore di numeri random coppie (x, y) di numeri nell'intervallo $[-1, 1]$. Per ogni coppia estratta si controlla se il punto è interno a un cerchio di raggio 1, ovvero si verifica che $x^2 + y^2 <= 1$. Il rapporto fra il numero di punti interni al cerchio e il numero totale di punti tende al valore $\pi/4$ al crescere del numero di campionamenti.

Realizzate una classe python che gestisca questa simulazione. La classe richiede il numero di campionamenti come parametro da passare al costruttore (metodo `__init__`) e definisce un metodo `run` che esegue la simulazione. Oltre al valore finale il metodo deve produrre il plot di convergenza, ovvero il plot del valore del rapporto durante la simulazione. A questo scopo è necessario definire come membro della classe una lista, o un array `numpy`, che contenga i valori del rapporto punti interni / punti totali durante la simulazione.

[Esercizio proposto da Marco D'Alessandro]