UNIVERSITÉ SAVOIE MONT BLANC

Analyse de données avancée

TD 2

Université Grenoble Alpes

1 - Loi de Breit-Wigner

La loi de Breit-Wigner est utilisée pour décrire des sections efficaces en physique nucléaire, comme l'interaction d'un neutron avec un noyau atomique au voisinage de l'énergie de résonnance

La variable aléatoire X ayant pour fonction densité de probabilité (loi de BreitWigner) :

$$f(x) = \frac{1}{\pi} \frac{\Gamma/2}{(\Gamma/2)^2 + (x-\mu)^2}$$

1. Calculer la fonction caractéristique de X pour $\Gamma = 2$ et $\mu = 0$.

2. On construit alors $Y = \frac{\Gamma}{2} X + \mu$. Montrer que Y suit bien une loi de Breit-Wigner de paramètres Γ et μ

3. En déduire la fonction caractéristique de Y, c-a-d pour une loi de Breit-Wigner générale.

4. Soit deux variables aléatoires indépendantes Y_1 et Y_2 ayant pour distribution la loi de de Breit-Wigner. Quelle est la loi de distribution de $Y_1 + Y_2$?

2 - Convolution d'une loi de Poisson et d'une loi de Bernouilli

Soit un processus de poisson où des événements se produisent avec une probabilité α par unité de temps. Parmi les événements, on a une probabilité p d'avoir un événement de type A. Quelle est la loi de distribution associée au nombre d'événements A se produisant pendant le temps T.

3 - Somme de distributions uniformes (+jupyter-notebook)

Soit la variable aléatoire :

$$Z = \frac{\sum_{i=1}^{n} X_i}{\sqrt{n}}$$

où les X_i sont des variables aléatoires de distribution uniforme entre $[-1/2,1/2]$

1. Retrouver le théorème de la limite centrale pour cette distribution.

2. Vérifier ce résultat par Monte-Carlo avec un code *python*. On pourra utiliser le jupyter-notebook TD2 disponible sur Moodle. Pour les tirages aléatoires, vous pouvez utiliser *numpy* ou la librairie *scipy.stats*. Voici comment faire un tirage uniforme avec *scipy.stats*.

```python
# importation des librairies usuelles
import numpy as np
import scipy.stats as stats

# création du générateurs [-1/2,1/2]
unif = stats.uniform(loc=-0.5,scale=1)

# tirage d'une liste de nb nombres uniformes sur [-1/2,1/2]
nb=10
sample = unif.rvs(size=nb)
```

4 - Intégration Monte Carlo (+jupyter-notebook)

1. En utilisant *scipy* (voir notebook TD2), estimer par Monte Carlo $\tilde{I}$ la valeur de l'intégrale suivante :

$$I = \int_0^1 f(x)\,dx = \int_0^1 x(1-x)\,dx = \frac{1}{6}$$

2. Calculer l'incertitude $\Delta\tilde{I}$ associée à l'estimation obtenue

3. Sur un même graphique, afficher $f(x)$ ainsi que les points générer par le MC (en rouge les points rejetés, en vert les points acceptés)

5 - Loi de χ^2 (+jupyter-notebook)

Soit un vecteur aléatoire $\vec{X}(X_1,...,X_n)$ à n dimensions. Les varaibles aléatoires X_i sont indépendantes et suivent une loi normale :

$$X_i \sim \mathcal{N}(0,1)$$

1. A partir du notebook python TD2 ou du code si dessous, vérifier que la somme des X_i^2 suit une loi de χ^2 à n dimensions :

$$\sum_i \frac{(X_i-\mu)^2}{\sigma^2} = \sum_i X_i^2 \sim \chi^2_{(n)}$$

```
nb = 1000
n=10
means = np.zeros(n) # vecteur des espérances à n dimensions
sigmas = np.ones(n) # vecteur des variances à n dimensions

norm = stats.multivariate_normal(means,sigmas) # création du générateur avec une loi normale pour chaque dimension
x = norm.rvs(size=nb) # tirage de nb vecteurs aléatoires
```

6 - Ecrire un générateur uniforme [0,1] (+jupyter-notebook)

Ecrire un générateur de Park & Miller :

$$n_{i+1} = 16807 \times n_i \mod (2^{31}-1)$$

1. Ecrire l'algorithme de Park & Miller pour fabriquer une liste de 10000 nombres aléatoires $\in [0,1]$

2. Afficher l'histogramme des valeurs obtenus grâce à l'histogramme de matplotlib.pyplot

3. Calculer la longueur des séquences croissantes

4. Comparer l'histogramme obtenu à la probabilité théorique :

$$p(s) = \frac{1}{s!} - \frac{1}{(s-1)!}$$