TD 2: Méthodes MCMC et diagnostics de convergence

Objectifs:

1. Implémenter l'algorithme de Gibbs et visualiser les distributions obtenues
2. Faire les diagnostics de convergence apropriés
3. Comprendre l'effet du nombre d'échantillons et la corrélation sur la qualité de la simulation
4. Manipuler les paramètres de burn-in et thinning selon les metrics $\hat{R}$ et l'ESS (Effective Sample Size).
5. Apprendre à interpréter et faire un diagnostic complet MCMC.

Soient X et Y deux variables aléatoires telles que :

$$\begin{pmatrix} X \\ Y \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} \right), \quad \rho \in]0, 1[.$$

On rappelle que $\begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}^{-1} = \frac{1}{1-\rho^2} \begin{pmatrix} 1 & -\rho \\ -\rho & 1 \end{pmatrix}$.

1. Algorithme MCMC

1. Montrez que les distributions conditionnelles de X sachant $Y = y$ et Y sachant $X = x$ sont données par $\mathcal{N}(\rho y, 1 - \rho^2)$ et $\mathcal{N}(\rho x, 1 - \rho^2)$

2. Quel algorithme est adéquat pour Simuler la loi jointe (X, Y) ? Implémentez-le – **sans l'aide d'une IA** – en numpy dans une fonction prenant en entrée 4 arguments: le nombre d'échantillons n, ρ, une valeur d'initialisation de la chaîne ainsi qu'une seed ou un PRNG.

2. Analyse simples à une chaîne

On fixe ici $\rho = 0.2$.

3. Implémentez – **avec l'aide d'une IA** – une fonction qui prend en argument deux vecteurs de grille uniforme `x_array` et `y_array` entre -4 et 4 de même taille (1000) et visualise les contours plots en 2D de la densité jointe de la Gaussienne (X, Y).

4. Adaptez cette fonction pour prendre en input les données simulées X_i et Y_i au lieu des grilles uniformes.

5. Visualisez les contours plots obtenus avec $n = 10, 100, 1000$ puis $\rho = 0.2$ et $\rho = 0.9$. Obtenez-vous la bonne densité jointe avec vos échantillons?

6. Installez le package `arviz` (pip install arviz). Arviz implémente tous les diagnostics MCMC. On doit d'abord construire un objet "inferenceData" qui prend en input un dictionnaire avec les données d'une chaîne unidimensionnelle éventuellement répétées (pour, entre autres, calculer le $\hat{R}$). "n_chains" ci-dessous correspond au nombre de répétitions de la même chaîne (n_chains = 1 pour l'instant). Ainsi, il faudra un objet 'i_data' pour X et un autre pour Y. En adaptant le code-dessous, interprétez le diagnostic obtenu avec $n = 1000$ et $\rho = 0.1$ puis $\rho = 0.5$.

```python
import arviz as az
chains_data_1d = .... # np array de taille (n_chains, n_samples)
i_data = az.convert_to_inference_data({"X"=chains_data_1d)) # on
    construit l'objet inf_data nécessaire pour arviz

az.plot_trace(i_data) # visualiser les chaines / densités
az.plot_autocorr(i_data) # visualiser auto_corr
print(az.summary(kind="diagnostics")) # afficher les métriques ESS, R
print(az.ess(i_data, relative=True)) # calcule ESS bulk normalisé par le
    nombre total d'échantillons.
```

7. Expliquez la valeur obtenue (NaN) du $\hat{R}$.

3. Analyse avec plusieurs chaînes En pratique, on utilise souvent 4 chaînes (avec des initialisations différentes) afin d'étudier la convergence.

8. Simulez 4 chaînes indépendantes avec votre fonction MCMC et construisez les objets i_data correspondants. Interprétez les résultats obtenus avec différentes valeur de ρ.

9. Pour améliorer la qualité de l'estimation, ajouter un paramètre `tune` qui correspond au *burn_in* i.e le nombre d'échantillons "jetés" pour atteindre le régime stationnaire. A-t-il un effet conséquent sur les statistiques ?

10. Pour améliorer davantage l'estimation avec $\rho = 0.95$, ajoutez un paramètre `thinning` pour prendre un échantillon sur deux ou sur trois ou sur $\{thinning\}$. Quel effet cela a-t-il sur l'autocorrélation et l'ESS ?

11. Comparez les estimations de ESS bulk et ESS tail pour différentes valeurs de $\rho = 0.$ et 0.95. Comment pouvez-vous l'expliquer ?