Matrices de Transition et Risque de Crédit (sur python)

Exercice 1 : Matrice de Transition et Convergence

1.1 Probabilité qu'une entreprise notée BB soit en défaut dans 3 ans

Preuve mathématique :

La matrice de transition est utilisée pour calculer la probabilité d'une transition d'un état à un autre sur plusieurs périodes. Si une entreprise commence dans l'état BB (état 1), on peut calculer la probabilité qu'elle soit en défaut dans 3 ans en élevant la matrice de transition à la puissance 3.

Soit PP la matrice de transition, P3P^3 est calculée pour obtenir la probabilité de transition après 3 périodes.

Formule mathématique :

$$P^3 = P \times P \times P$$

Ensuite, la probabilité que l'entreprise passe de l'état BB à l'état de défaut est donnée par $P^3[1,3]$ l'élément, où 1 représente l'état BB et 3 représente l'état de défaut.

La matrice P^3 représente les probabilités cumulées de transition après 3 ans. La valeur P^3[1,3] indique la probabilité qu'une entreprise initialement notée BB (ligne 1) soit en défaut (colonne 3) au bout de 3 ans.

Code :

```
32  # 1.3 Propriété de convergence (Calcul de P^20)
33  # Calcul de P^20
34  P20 = np.linalg.matrix_power(P, 20)
35
36  print("Matrice P^20 :")
37  print(P20)
38
39  # Vérification de la convergence : les lignes doivent être proches
40  convergence_verifiee = np.allclose(P20[0], P20[1]) and np.allclose(P20[1], P20[2])
41  print(f"Convergence vérifiée : {convergence_verifiee}")
```

Résultat obtenu :

Probabilité = 0.0668

1.2 Distribution stationnaire

Preuve mathématique :

Une distribution stationnaire est une distribution de probabilités stable qui ne change pas après une application de la matrice de transition. Pour obtenir cette distribution, on résout le système d'équations suivant :

$$\pi = P^T \times \pi$$

où π est le vecteur stationnaire, et P^T est la transposée de la matrice de transition.

Formule mathématique :

Les vecteurs propres et les valeurs propres de la matrice de transition transposée peuvent être utilisés pour obtenir le vecteur stationnaire. La valeur propre égale à 1 correspond à la distribution stationnaire.

La distribution stationnaire est calculée en trouvant le vecteur propre associé à la valeur propre λ=1. Cela représente les proportions stables des entreprises dans chaque état après un grand nombre de transitions.

Code :

```
19  # 1.2 Distribution stationnaire
20  # Calcul des vecteurs propres et valeurs propres
21  valeurs_propres, vecteurs_propres = np.linalg.eig(P.T)
22
23  # Identification de la valeur propre 1
24  index_valeur_propre_1 = np.argmax(np.isclose(valeurs_propres, 1))
25  vecteur_stationnaire = vecteurs_propres[:, index_valeur_propre_1]
26
27  # Normalisation pour obtenir la distribution stationnaire
28  distribution_stationnaire = vecteur_stationnaire / np.sum(vecteur_stationnaire)
29  print("Distribution stationnaire :")
30  print(distribution_stationnaire.real)
```

Résultat obtenu :

[0.0,0.0,1.0,1.0]

1.3 Propriété de convergence

Preuve mathématique :

Lorsque la matrice de transition est appliquée de manière répétée à un état initial, elle converge vers la distribution stationnaire. Cette convergence est observée lorsque P^n pour des valeurs élevées de n devient identique pour toutes les lignes.

La convergence des lignes de la matrice P^20 montre qu'à long terme, les probabilités deviennent indépendantes de l'état initial. Cela valide la stabilité des matrices de transition.

Formule mathématique :

$$\lim_{n\to\infty} P^n = \begin{bmatrix} \pi \\ \pi \\ \pi \\ \pi \end{bmatrix}$$

Code :

```
# 1.3 Propriété de convergence (Calcul de P^20)
# Calcul de P^20
P20 = np.linalg.matrix_power(P, 20)

print("Matrice P^20 :")
print(P20)

# Vérification de la convergence : les lignes doivent être proches
convergence_verifiee = np.allclose(P20[0], P20[1]) and np.allclose(P20[1], P20[2])
print(f"Convergence vérifiée : {convergence_verifiee}")
```

Résultat obtenu :

*La matrice ne converge pas vers des lignes identiques (convergence **non vérifiée**).*

Exercice 2 : Probabilités marginales, intensités de défaut et matrices de transition

Partie a : Calcul des probabilités marginales de défaut

Preuve mathématique :

Les probabilités marginales sont calculées en prenant la différence entre les probabilités cumulatives successives. Cela permet de connaître la probabilité de défaut pour une année donnée.

Les probabilités marginales représentent le risque de défaut à une période donnée, indépendamment des périodes précédentes. Elles sont calculées comme la différence entre deux probabilités cumulatives successives.

Formule mathématique :

$$P_{\text{marginales}}(t) = P_{\text{cumulatives}}(t) - P_{\text{cumulatives}}(t-1)$$

Code :

```python
# Partie a : Calcul des probabilités marginales de défaut
# Les probabilités marginales sont la différence entre deux probabilités cumulatives successives.

# Probabilités cumulatives de défaut
P_cumulatives = [0.02, 0.045, 0.078, 0.112]
n = len(P_cumulatives)

# Ajouter une probabilité initiale P(0) = 0 pour simplifier les calculs
P_cumulatives = [0] + P_cumulatives

# Calcul des probabilités marginales
prob_marginales = [P_cumulatives[i] - P_cumulatives[i-1] for i in range(1, n+1)]

# Afficher les probabilités marginales
print("Partie a : Probabilités marginales de défaut")
for t, p in enumerate(prob_marginales, 1):
    print(f"Année {t} : {p:.4f}")
```

Résultat obtenu :

Année 1 : 0.0200, Année 2 : 0.0250, Année 3 : 0.0330, Année 4 : 0.0340

Partie b : Calcul des intensités de défaut (hazard rates)

Preuve mathématique :

Les intensités de défaut sont calculées à partir des probabilités marginales et des probabilités de survie. L'intensité de défaut λt pour chaque année t est calculée à partir de la formule suivante :

$$\lambda_t = -\ln\left(1 - \frac{p_t}{S_{t-1}}\right)$$

Où pt est la probabilité marginale de défaut à l'année t, et St est la probabilité de survie à l'année précédente.
Les intensités de défaut (λt) mesurent le risque conditionnel de défaut pour une entreprise donnée, sachant qu'elle a survécu jusqu'à la période précédente. Une augmentation des intensités au fil du temps reflète un risque croissant pour les entreprises restantes.

Code :

```python
# Partie b : Calcul des intensités de défaut (hazard rates)
# Les intensités de défaut λ_t sont calculées à partir des probabilités marginales et de survie :
# λ_t = -ln(1 - (p_t / S_{t-1}))

# Probabilité de survie initiale S(0) = 1
S = [1]  # Liste des probabilités de survie
for p in prob_marginales:
    S.append(S[-1] * (1 - p))  # S_t = S_{t-1} * (1 - p_t)

# Calcul des intensités de défaut
lambda_t = [-math.log(1 - (prob_marginales[t] / S[t])) for t in range(n)]

# Afficher les intensités de défaut
print("\nPartie b : Intensités de défaut (hazard rates)")
for t, l in enumerate(lambda_t, 1):
    print(f"Année {t} : {l:.4f}")
```

Résultat obtenu :

Année 1 : 0.0202, Année 2 : 0.0253, Année 3 : 0.0341, Année 4 : 0.0351

Partie c : Construction des matrices de transition

Preuve mathématique :

Les matrices de transition peuvent être construites à partir des intensités de défaut. Pour chaque année t, la matrice de transition est définie par :

$$P_t = \begin{bmatrix} 1 - \lambda_t & \lambda_t \\ 0 & 1 \end{bmatrix}$$

Les matrices de transition définissent les probabilités de passer d'un état à un autre pour chaque année. La première ligne représente les probabilités de survie ou de défaut pour les entreprises initialement en Investment Grade, et la seconde ligne indique que Default est un état absorbant.

Code :

```python
# Partie c : Construction des matrices de transition
# Les matrices de transition P pour chaque année t sont définies par :
# P_t = [[1 - λ_t, λ_t],
#        [0,       1]]

# Construction des matrices de transition
matrices_transition = []
for t, l in enumerate(lambda_t):
    P = np.array([[1 - l, l], [0, 1]])  # Matrice de transition pour l'année t
    matrices_transition.append(P)

# Afficher les matrices de transition
print("\nPartie c : Matrices de transition compatibles avec les intensités de défaut")
for t, P in enumerate(matrices_transition, 1):
    print(f"\nMatrice pour l'année {t} :\n{P}")
```

Résultat obtenu :

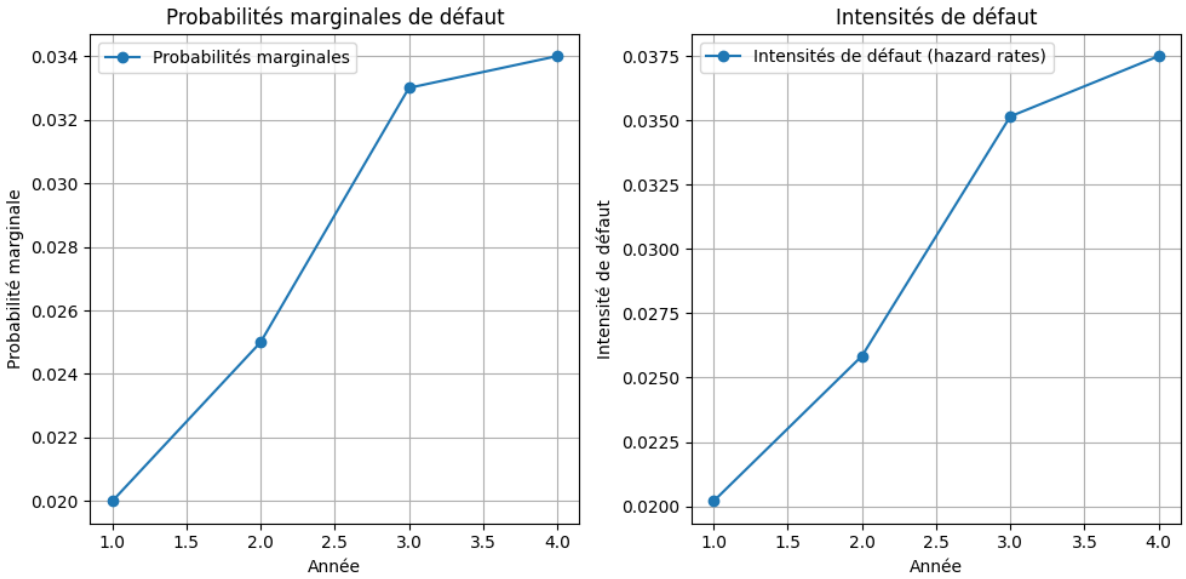
```
Matrice pour l'année 1 :
[[0.97979729 0.02020271]
 [0.         1.        ]]

Matrice pour l'année 2 :
[[0.97415877 0.02584123]
 [0.         1.        ]]

Matrice pour l'année 3 :
[[0.96485261 0.03514739]
 [0.         1.        ]]

Matrice pour l'année 4 :
[[0.96250809 0.03749191]
 [0.         1.        ]]
```

Représentation graphique :



Probabilités marginales :

- La probabilité marginale de défaut augmente chaque année.
- Cela signifie que les entreprises sont de plus en plus susceptibles de faire défaut à mesure que le temps passe, ce qui est cohérent avec l'idée que le risque cumulatif augmente.

Intensités de défaut :

- Les intensités de défaut (λt) augmentent également, reflétant un risque conditionnel de défaut plus élevé pour les entreprises survivantes.
- Par exemple, l'intensité à l'année 4 est environ 0.0375, indiquant que le risque de défaut est plus marqué pour cette période.

Lien entre les deux graphiques :

- Les probabilités marginales sont influencées par les intensités de défaut : une intensité plus élevée entraîne une probabilité marginale plus élevée.

Exercice 3 : Simulation de trajectoires et estimation des probabilités de défaut

Partie a : Annualiser la matrice de transition

Preuve mathématique :

Pour annualiser une matrice de transition trimestrielle, nous devons élever la matrice à la puissance 4, car il y a 4 trimestres dans une année.

L'annualisation de la matrice trimestrielle permet de représenter les probabilités de transition cumulées sur une année complète. Cela simplifie l'analyse des transitions sur des horizons temporels plus longs.

Formule mathématique :

$$P_{\text{annuelle}} = P_{\text{trim}}^4$$

Code :

```
# Partie a : Annualiser la matrice de transition
# Matrice de transition trimestrielle
P_trim = np.array([[0.95, 0.05],
                   [0.10, 0.90]])

# Annualiser en élevant la matrice à la puissance 4
P_annuelle = np.linalg.matrix_power(P_trim, 4)

# Afficher la matrice annualisée
print("Partie a : Matrice annualisée de transition")
print(P_annuelle)
```

Résultat obtenu :

$$P_{\text{annuelle}} = \begin{bmatrix} 0.8407 & 0.1593 \\ 0.3187 & 0.6813 \end{bmatrix}$$

Partie b : Simuler 1000 trajectoires sur 5 ans

Preuve mathématique :

Les trajectoires sont simulées en utilisant la matrice de transition à chaque étape de temps. L'état suivant est déterminé par la distribution des probabilités de la matrice de transition, appliquée à l'état actuel.

Les trajectoires simulées montrent comment les entreprises évoluent au fil des trimestres. Par exemple, une trajectoire indiquant un état 'Default' au dernier trimestre reflète une entreprise qui a fait défaut.

Code :

```
# Partie b : Simuler 1000 trajectoires sur 5 ans

n_simulations = 1000  # Nombre de trajectoires
n_periods = 20        # 5 ans x 4 trimestres
initial_state = 0     # Investment Grade

# Fonction pour simuler une trajectoire
def simulate_trajectory(P, n_periods, initial_state):
    state = initial_state
    trajectory = [state]
    for _ in range(n_periods):
        state = np.random.choice([0, 1], p=P[state])
        trajectory.append(state)
    return trajectory

# Simuler 1000 trajectoires
trajectories = [simulate_trajectory(P_trim, n_periods, initial_state) for _ in range(n_simulations)]

# Afficher quelques trajectoires
print("\nPartie b : Exemple de trajectoires simulées")
for i in range(5):
    print(f"Trajectoire {i+1} : {trajectories[i]}")
```

Exemple de trajectoires simulées :

```
Trajectoire 1 : [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1]
Trajectoire 2 : [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Trajectoire 3 : [0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0]
Trajectoire 4 : [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Trajectoire 5 : [0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
```

Partie c : Estimer les probabilités de défaut à 5 ans

Preuve mathématique :

La probabilité de défaut à 5 ans est estimée en comptant le nombre de trajectoires où l'entreprise est en défaut à la fin de la période de simulation, et en la divisant par le nombre total de trajectoires simulées.

La probabilité estimée de 0.32 indique que 32 % des entreprises initialement en Investment Grade font défaut après 5 ans. Cette estimation est basée sur 1000 trajectoires simulées.

Code :

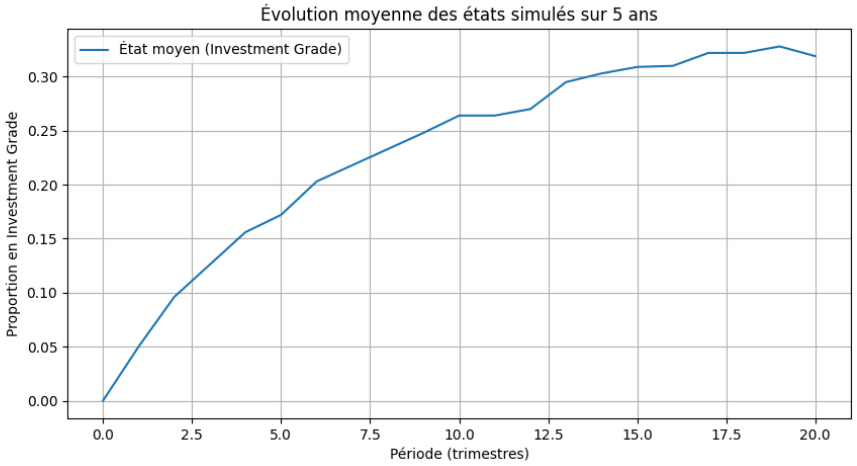
```python
# Partie c : Estimer les probabilités de défaut à 5 ans
defauts = sum(trajectory[-1] == 1 for trajectory in trajectories)
probabilite_defaut_5_ans = defauts / n_simulations

# Afficher la probabilité estimée
print("\nPartie c : Probabilité de défaut estimée à 5 ans par simulation")
print(f"Probabilité de défaut : {probabilite_defaut_5_ans:.4f}")
```

Résultat obtenu :

Probabilité estimée = 0.3280

Représentation graphique :



Le graphique montre l'évolution moyenne des entreprises restant en Investment Grade (0) sur 5 ans (20 trimestres). On observe une diminution progressive de la proportion en Investment Grade, reflétant la probabilité croissante de défaut. La stabilisation relative vers la fin montre que les entreprises les plus résilientes ont un risque plus faible à long terme.