RÉPUBLIQUE DÉMOCRATIQUE DU CONGO

UNIVERSITÉS NOUVEAUX HORIZONS

Faculté des sciences informatiques

ALGO.NUM ET APPROXIMATION

EXAMEN/rapport

Nom : BANZA MUKALAY NINA.

Promotion : licence 3 info/gestion Administration

Dispensé par le professeur : jordan masakuna

ANNÉE UNIVERSITAIRE 2023-2024

Rapport du Projet d'Approximation de Fonctions

- ***Objectif***

L'objectif de ce projet est d'explorer différentes méthodes d'approximation de fonctions pour modéliser un ensemble de données simulées. Les méthodes étudiées sont la méthode de Gauss (Directe), la méthode de Lagrange, et la méthode de Newton. L'analyse inclut l'estimation de la fonction, le calcul de l'erreur d'approximation, et la visualisation des résultats.

- ***Génération des Données***

Les données ont été générées en simulant 20 observations avec la relation sous-jacente \(y = 3x^2 + 2x + 1\), à laquelle un bruit aléatoire a été ajouté pour simuler des variations réalistes. Les valeurs de \(x\) ont été choisies aléatoirement dans l'intervalle [0, 10].

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import lagrange
from sklearn.metrics import mean_squared_error

# Génération des données
np.random.seed(0)
x = np.random.rand(20) * 10
y = 3 * x ** 2 + 2 * x + 1 + np.random.randn(20) * 2
```

- ***Méthodes d'Approximation***

Méthode de Gauss (Directe)

Cette méthode résout un système d'équations linéaires pour trouver les coefficients du polynôme qui approxime le mieux les données.

```
# Méthode de Gauss (Directe)
A = np.vstack([x**2, x, np.ones_like(x)]).T
m, c, _ = np.linalg.lstsq(A, y, rcond=None)[0]
```

- ***Méthode de Lagrange***

La méthode de Lagrange génère un polynôme qui passe exactement par tous les points de données.

```
# Méthode de Lagrange
poly = lagrange(x, y)
```

- ***Méthode de Newton***

Similaire à la méthode de Lagrange, la méthode de Newton trouve un polynôme qui passe par tous les points.

```
# Méthode de Newton
coeffs = np.polyfit(x, y, 2)
```

- ***Calcul de l'Erreur***

L'erreur d'approximation a été calculée en utilisant l'erreur quadratique moyenne (RMSE) pour chaque méthode.

```
# Calcul de l'erreur
y_pred_gauss = m * x**2 + c * x + 1
y_pred_lagrange = poly(x)
y_pred_newton = np.polyval(coeffs, x)

rmse_gauss = np.sqrt(mean_squared_error(y, y_pred_gauss))
rmse_lagrange = np.sqrt(mean_squared_error(y, y_pred_lagrange))
rmse_newton = np.sqrt(mean_squared_error(y, y_pred_newton))

print(f"RMSE Gauss: {rmse_gauss}")
print(f"RMSE Lagrange: {rmse_lagrange}")
print(f"RMSE Newton: {rmse_newton}")
```

- ***Visualisation des Données***

Les données originales et les courbes d'approximation obtenues par chaque méthode ont été visualisées sur un graphique.

```
# Visualisation des données
plt.figure(figsize=(10, 6))
plt.scatter(x, y, label='Données originales')
plt.plot(x, y_pred_gauss, label='Gauss (Directe)')
plt.plot(x, y_pred_lagrange, label='Lagrange')
plt.plot(x, y_pred_newton, label='Newton')
plt.legend()
plt.xlabel('x')
plt.ylabel('y')
plt.title('Approximation de la fonction')
plt.show()
```

- ***Conclusion***

La comparaison des valeurs RMSE a permis de déterminer la méthode offrant la meilleure approximation. La méthode avec le RMSE le plus bas a été considérée comme la plus précise pour modéliser les données fournies. La visualisation a également joué un rôle clé en offrant une compréhension intuitive de la performance de chaque méthode d'approximation.

le code source en .py

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import lagrange
from sklearn.metrics import mean_squared_error

# Génération des données

np.random.seed(0)
x = np.random.rand(20) * 10
y = 2 * x ** 2 - 3 * x + 1 + np.random.randn(20) * 2

# Méthode de Gauss (Directe)

A = np.vstack([x**2, x, np.ones_like(x)]).T
m, c, _ = np.linalg.lstsq(A, y, rcond=None)[0]

# Méthode de Lagrange

poly = lagrange(x, y)

# Méthode de Newton
coeffs = np.polyfit(x, y, 2)

# Calcul de l'erreur

y_pred_gauss = m * x**2 + c * x + 1
y_pred_lagrange = poly(x)
y_pred_newton = np.polyval(coeffs, x)

rmse_gauss = np.sqrt(mean_squared_error(y, y_pred_gauss))
rmse_lagrange = np.sqrt(mean_squared_error(y, y_pred_lagrange))
rmse_newton = np.sqrt(mean_squared_error(y, y_pred_newton))

print(f"RMSE Gauss: {rmse_gauss}")
print(f"RMSE Lagrange: {rmse_lagrange}")
print(f"RMSE Newton: {rmse_newton}")

# Visualisation des données
```

```
plt.figure(figsize=(10, 6))
plt.scatter(x, y, label='Données originales')
plt.plot(x, y_pred_gauss, label='Gauss (Directe)')
plt.plot(x, y_pred_lagrange, label='Lagrange')
plt.plot(x, y_pred_newton, label='Newton')
plt.legend()
plt.xlabel('x')
plt.ylabel('y')
plt.title('Approximation de la fonction')
plt.show()
```

RESULTAT