RAMIADAMANANA Lovaharisoa Félicité
L3RSI – 2024-2025
N° FI2301-133

Date : Novembre 2025

Rapport Technique – Régression Linéaire avec Python

1. Introduction

Ce projet s'inscrit dans le cadre de l'examen L3 en développement Python scientifique. L'objectif est de concevoir une application graphique permettant de résoudre et modéliser des problèmes mathématiques dans quatre domaines : systèmes linéaires, programmation linéaire, régression linéaire et processus stochastiques. L'interface est développée avec **PyQt5**, et les calculs sont réalisés à l'aide de bibliothèques comme **NumPy**, **Matplotlib**, **PuLP** et **Scikit-learn**.

2. Architecture du projet

Structure du projet :

```
projet_math_app/
··· ui/                                  # Interface graphique
 · ··· main_window.py                    # Fenêtre principale avec onglets
··· core/                                # Modules de calcul
 · ··· systeme_lineaire.py               # Résolution de systèmes linéaires
 · ··· programmation_lineaire.py         # Programmation linéaire
 · ··· regression_lineaire.py            # Régression linéaire
 · ··· processus_stochastique.py         # Simulation de processus
stochastiques
 ··· data/                               # Données et fichiers de test
```

```
· ··· exemple_donnees.csv                  # Données pour la régression
· ··· systeme_lineaire_test.json           #systeme_lineaire_test.json
· ··· programmation_lineaire_test.json     # Objectif et contraintes pour test
· ··· processus_stochastique_test.json     # Matrice de transition pour test
··· main.py                                # Point d'entrée pour lancer
l'application
··· requirements.txt                       # Dépendances du projet
··· rapport.pdf                            # Rapport technique
··· AI_USAGE.txt                           # Parties générées par IA
```

3. Description des modules

2.1 Système Linéaire

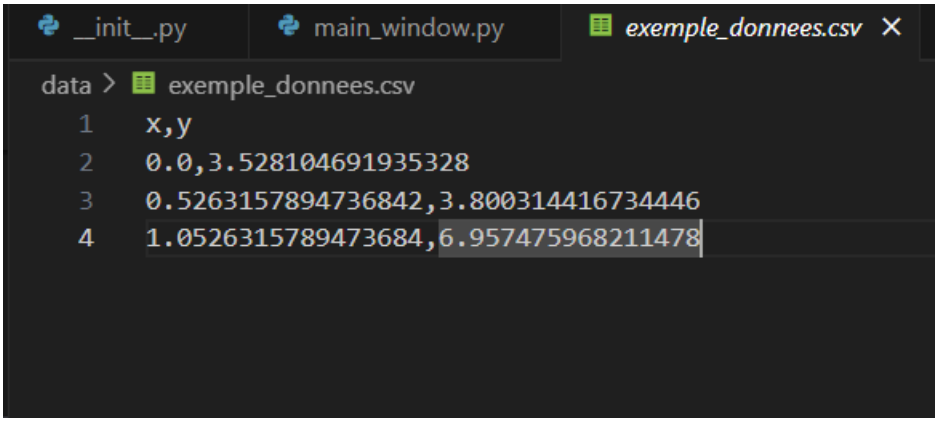
- **Fonctionnalité** : Résolution de systèmes de la forme AX = b
- **Méthodes utilisées** :
  - `numpy.linalg.solve()` pour une solution rapide.
  - Méthode de Gauss manuelle pour illustrer l'algorithme.
- **Code source** : `core/systeme_lineaire.py`
- **Test** : Matrice 3x3 et vecteur b.
- **Résultat** : Affichage des solutions dans l'interface.

2.2 Programmation Linéaire

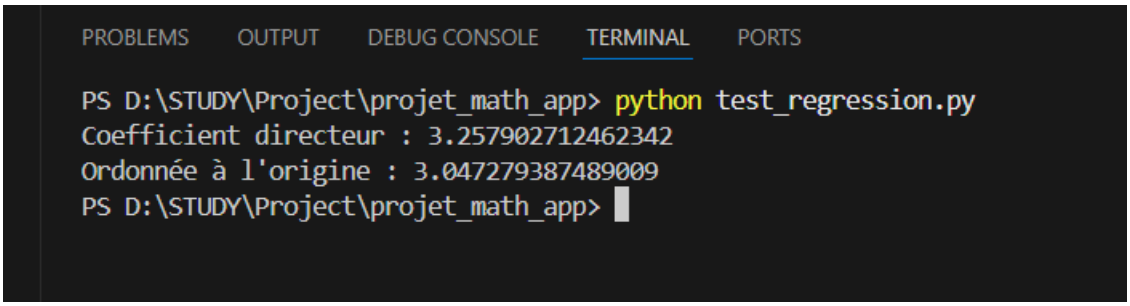
- **Fonctionnalité** : Maximisation ou minimisation d'une fonction linéaire sous contraintes.
- **Bibliothèque utilisée** : `PuLP`
- **Exemple** :
  - Max $Z=3x+2y$
  - Contraintes : $x+y\leq4$, $3x+2y\leq12$, $x,y\geq0$
- **Code source** : `core/programmation_lineaire.py`
- **Résultat** : Affichage des valeurs optimales de x et y.

2.3 Régression Linéaire

- **Fonctionnalité** : Ajustement d'un modèle linéaire à des données réelles.
- **Bibliothèque utilisée** : `scikit-learn`
- **Données** : Fichier CSV `data/exemple_donnees.csv` contenant les colonnes x et y.



- **Code source** : `core/regression_lineaire.py`
- **Résultat** :
  - Coefficient directeur : 3.2579
  - Ordonnée à l'origine : 3.0472



- ***2.4 Processus Stochastique***

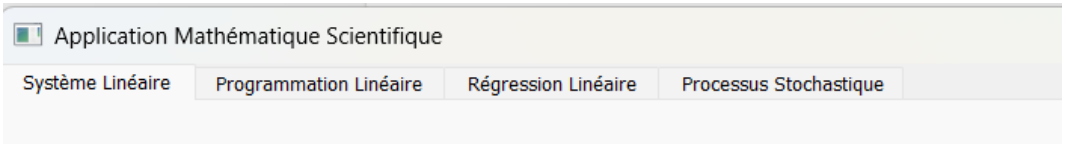
- **Fonctionnalité** : Simulation d'une chaîne de Markov.
- **Méthode utilisée** : `numpy.random.choice()` avec une matrice de transition.
- **Code source** : `core/processus_stochastique.py`
- **Exemple** : Matrice de transition 3x3.
- **Résultat** : Trajectoire simulée affichée dans l'interface.

4. Algorithmes utilisés

- **Méthode de Gauss** : Élimination et substitution arrière.
- **Programmation linéaire** : Modélisation avec `LpProblem`, résolution avec `solve()`.
- **Régression linéaire** : Ajustement par moindres carrés via `LinearRegression`.
- **Chaîne de Markov** : Simulation probabiliste avec choix aléatoire pondéré.

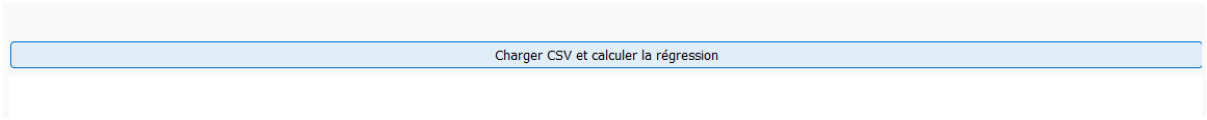
5. Interface graphique

- **Technologie** : PyQt5
- **Structure** :
  - Fenêtre principale avec onglets pour chaque module:

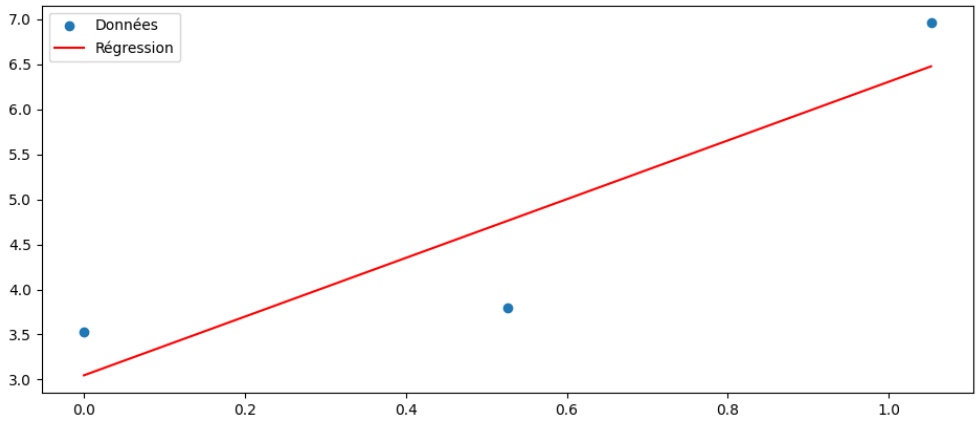


  - Saisie des données via champs ou fichiers.

  - Boutons pour lancer les calculs.



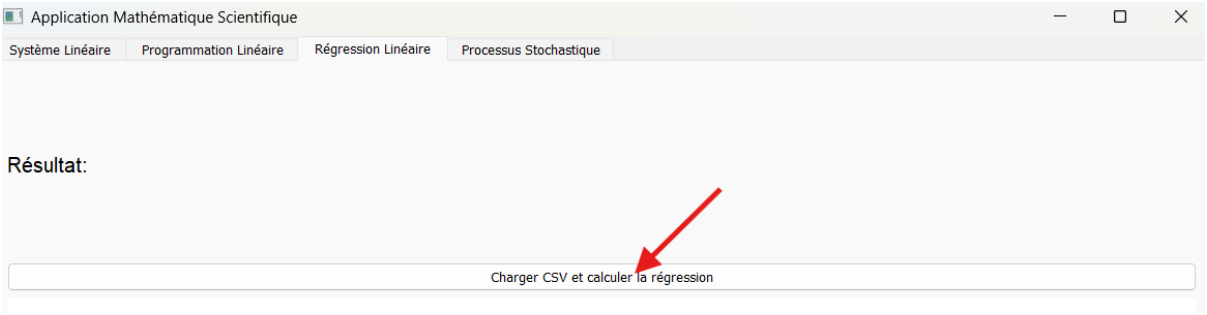
  - Affichage des résultats textuels et graphiques (courbes, trajectoires).

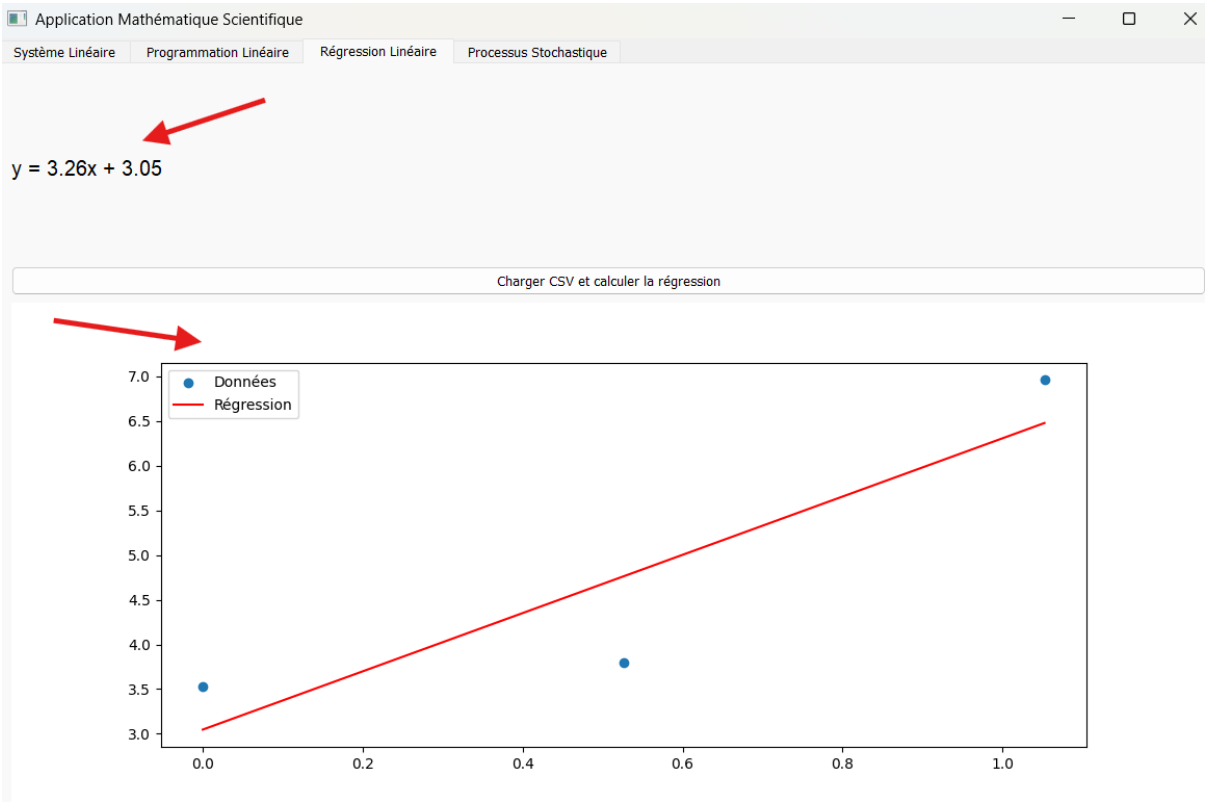


- **Code source** : `ui/main_window.py`

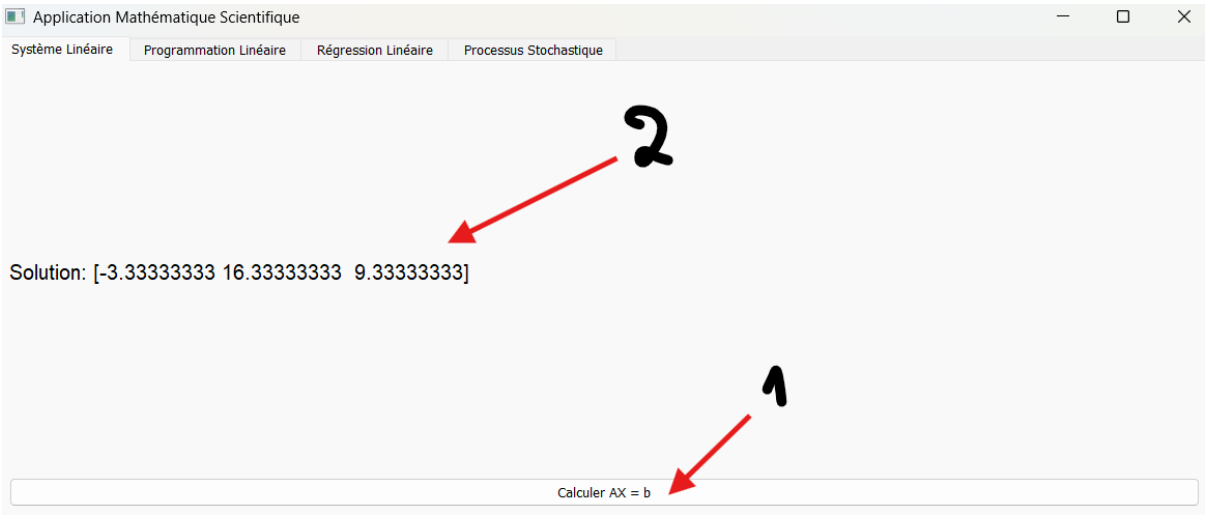
6. Jeux de tests

- **Régression** : Données simulées dans `exemple_donnees.csv`.



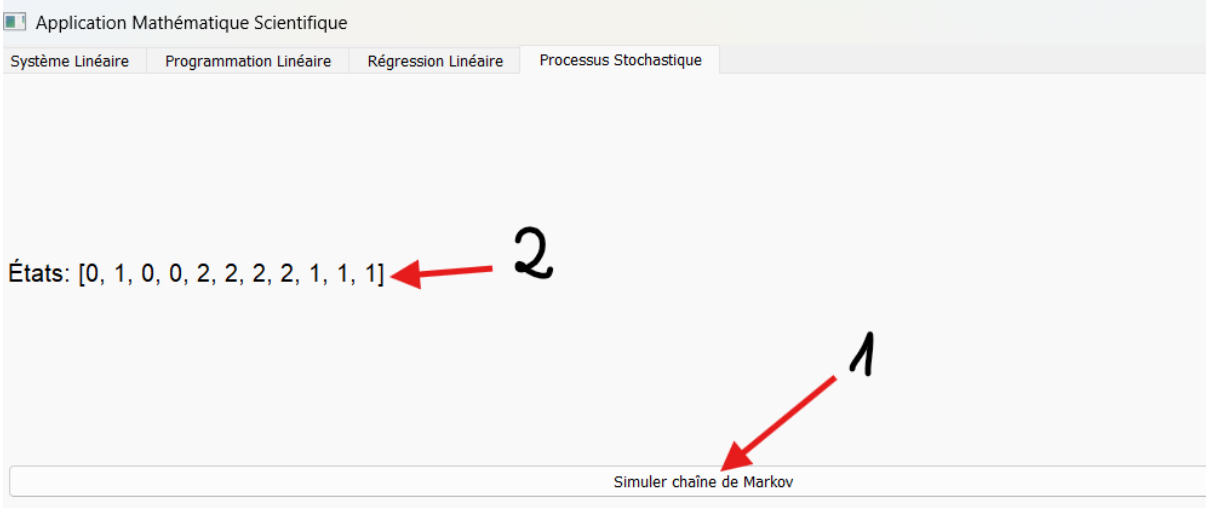


- **Système linéaire** : Matrices et vecteurs définis en dur.



- **Programmation linéaire** : Problème classique avec contraintes.

- **Processus stochastique** : Matrice de transition 3x3.



7. Limites et améliorations

- **Limites** :
  - Interface encore basique (pas de validation des entrées):
    - L'application ne dispose pas encore de mécanismes de validation des entrées. Ainsi, une saisie incorrecte (valeurs non numériques, dimensions incompatibles, etc.) peut provoquer des erreurs ou empêcher le bon fonctionnement du programme.
  - Pas de sauvegarde des résultats:
    - Les calculs effectués ne sont pas encore enregistrés, ce qui oblige l'utilisateur à relancer le programme pour chaque nouvelle utilisation. Aucun historique ou suivi n'est donc disponible.
  - Pas de gestion d'erreurs avancée.
    - Le système ne prévoit pas encore de gestion d'erreurs avancée. En cas d'entrée invalide ou d'exception (par exemple, matrice non

inversible), aucune explication claire n'est fournie à l'utilisateur.

- **Améliorations possibles** :
    - Amélioration de l'interface utilisateur :
        - Intégrer une validation automatique des données saisies (par exemple, vérifier le format des nombres ou la compatibilité des dimensions matricielles) et ajouter des messages d'erreur clairs et conviviaux.
    - Ajout de visualisations interactives:
        - Permettre la représentation graphique des résultats (par exemple, tracé des courbes, affichage des vecteurs ou des plans de régression) à l'aide de bibliothèques comme Matplotlib ou Plotly.
    - Intégration de nouvelles méthodes de calcul :
        - Étendre les fonctionnalités en ajoutant d'autres approches mathématiques, telles que la décomposition LU, la méthode de Jacobi, ou encore la régression multiple pour les analyses statistiques.
    - Sauvegarde et exportation des résultats :
        - Offrir la possibilité d'enregistrer les résultats sous différents formats (PDF, Excel, CSV) afin de faciliter leur exploitation ou leur partage.
    - Ajout de tests unitaires et automatisation:
        - Mettre en place une suite de tests unitaires pour garantir la fiabilité du code et simplifier la maintenance. L'utilisation d'outils comme *pytest* permettrait d'automatiser la vérification du bon fonctionnement des modules.