

Application graphique de résolution et de modélisation mathématique

Ce dépôt contient une application Python avec interface graphique (Streamlit) permettant d'effectuer des calculs mathématiques avancés dans les domaines suivants :

1. Résolution de systèmes linéaires
2. Programmation linéaire
3. Régression linéaire
4. Processus stochastiques (simulation de chaînes de Markov)

Le code est organisé en : - core/ : implémentations algorithmiques et fonctions de calcul - ui/ : modules Streamlit (une page par fonctionnalité) - datas/ : jeux de données et captures d'écran

Table des matières

- [Installation et lancement](#)
 - [Architecture du projet](#)
 - [Modules et algorithmes](#)
 - [Interface utilisateur](#)
 - [Jeux de tests et captures d'écran](#)
 - [Limitations et pistes d'amélioration](#)
 - [Licence / Remarques](#)
-

Installation et lancement

Suivez ces instructions selon votre système d'exploitation. L'objectif : créer un environnement virtuel, installer les dépendances et lancer l'application avec une seule commande `python main.py`.

Pré-requis

- Python 3.10+ recommandé
- Git (optionnel)

Récupérer le projet

cloner depuis GitHub (optionnel)

```
git clone https://github.com/AnicetJonhia/ScientificPythonApplication.git
cd "ScientificPythonApplication"
```

Linux / macOS

créer et activer un environnement virtuel

```
python3 -m venv venv
source venv/bin/activate
```

```
# installer dépendances
python -m pip install --upgrade pip
python -m pip install -r requirements.txt
```

```
# Lancer l'application
python main.py
```

Après `python main.py`, Streamlit démarre et affiche l'URL locale (par défaut `http://localhost:8501`) : ouvrez-la dans votre navigateur.

Windows (PowerShell)

```
# créer et activer venv
python -m venv venv
venv\Scripts\Activate.ps1
```

```
# installer dépendances
python -m pip install --upgrade pip
python -m pip install -r requirements.txt
```

```
# Lancer
python main.py
```

Lancer Streamlit directement (optionnel)

```
streamlit run app.py
```

Dépannage rapide

- Erreur “Could not find the ‘streamlit’ module” : activez venv et installez les dépendances
- Si le port 8501 est occupé, Streamlit proposera un autre port

Architecture du projet

- `core/` : fonctions de calcul (systèmes linéaires, programmation linéaire, régression, stochastique)
- `ui/` : pages Streamlit pour chaque fonctionnalité
- `datas/` : jeux de données d'exemple et captures d'écran
- `docs/` : documentation additionnelle

Modules et algorithmes

1) Systèmes linéaires

- Méthodes disponibles : résolution directe (`numpy.linalg.solve`), élimination de Gauss (avec pivot partiel), décomposition LU (Doolittle), moindres carrés (`numpy.linalg.lstsq`).

- API : `core.linear_system.solve_linear_system(A, b, method='auto' | 'gauss' | 'lu' | 'lstsq' | 'direct')`.

2) Programmation linéaire

- Modélisation via PuLP (si installé). Supporte maximisation/minimisation, contraintes \leq , \geq , $=$, et bornes de variables.
- API : `core.linear_programming.solve_lp(c, A, b, maximize=True, bounds=None, signs=None)`.

3) Régression linéaire

- Utilise scikit-learn LinearRegression si disponible, sinon `numpy.polyfit` en fallback.
- Retourne coefficients et fonction de prédiction; option `return_score=True` pour récupérer R^2 .

4) Processus stochastiques

- Simulation de chaînes de Markov, distribution empirique, estimation de la distribution stationnaire (méthode des valeurs propres puis power iteration).

Interface utilisateur

L'interface est développée avec Streamlit. Un menu latéral permet de sélectionner l'un des quatre modules. Chaque module propose : - saisie manuelle des données (matrices, vecteurs, contraintes) - upload de fichiers CSV lorsque pertinent - sélection de la méthode de résolution (quand applicable) - affichage des résultats textuels et graphiques (Plotly / Matplotlib)

Pour démarrer, lancez `python main.py` puis utilisez la sidebar pour naviguer.

Jeux de tests et captures d'écran

Les captures se trouvent dans `datas/screenshots/`.

Systèmes linéaires

Light:



Dark:



Programmation linéaire

Light:

Menu

Systèmes linéaires

Programmation linéaire

Régression linéaire

Processus stochastiques

RANDRIANAMBININA

Tokiniaina Jean Anicet Jonhia

L3IDEV

ESTI

Deploy

Application graphique de résolution et de modélisation mathématique

Programmation linéaire

Coefficients de la fonction objectif (c1,c2)

3,2

Objectif

Maximiser

Minimiser

Contraintes (une par ligne, format: a1,a2=b). Exemple: 1,0<=4

1,0<=4
0,1<=3
1,1<=5

Variables non négatives (x >= 0)

(Optionnel) Bornes des variables, une par ligne 'low,up' ou 'None' pour illimité

Résoudre

Dark:



Régression linéaire

Light:



Dark:



Processus stochastiques

Light:

Menu

Systèmes linéaires

Programmation linéaire

Régression linéaire

Processus stochastiques

RANDRIANAMBININA

Tokinialina Jean Anicet Jonhia

L3IDEV

ESTI

Deploy

Application graphique de résolution et de modélisation mathématique

Processus stochastiques — Chaînes de Markov

Matrice de transition (ex: $\begin{bmatrix} 0.5 & 0.5 & 0 \\ 0.2 & 0.4 & 0.4 \\ 0.0 & 0.3 & 0.7 \end{bmatrix}$)

[[0.5,0.5,0],[0.2,0.4,0.4],[0.0,0.3,0.7]]

Etat initial (index)

0

Nombre d'étapes

200

Graine RNG (optionnel pour reproductibilité, 0 = aléatoire)

0

☒ Calculer distribution stationnaire (valeurs propres/power)

Simuler

Dark:

Menu

Systèmes linéaires

Programmation linéaire

Régression linéaire

Processus stochastiques

RANDRIANAMBININA

Tokinialina Jean Anicet Jonhia

L3IDEV

ESTI

Deploy

Distribution empirique finale: [0.1493 0.393 0.4577]

Distribution stationnaire (approx):

value	
	0.1493
	0.3959
	0.4578

Limitations et pistes d'amélioration

- La décomposition LU actuelle n'inclut pas de pivotage complet ; ajouter le pivotage améliorerait la robustesse numérique.
- Améliorer l'éditeur de matrices (grille interactive) pour la saisie directe.
- Ajouter des tests unitaires (pytest) pour core/.