Mini-Projet B : analyse numérique

Objectif du projet

L'objectif de ce projet est d'implémenter et de comparer différentes méthodes d'intégration numérique appliquées à un polynôme de degré trois. Les méthodes étudiées sont :

La méthode des rectangles, la méthode des trapèzes, la méthode de Simpson, les méthodes préprogrammées (bibliothèques standards).

On évaluera la précision de ces méthodes en les comparant à la solution analytique, ainsi que leur convergence lorsque le nombre de segments augmente. Une analyse des performances (temps d'exécution) sera également réalisée à l'aide de la fonction timeit.

Méthodologique

Pour mener à bien cette étude, nous suivons une approche structurée en plusieurs étapes. Tout d'abord, nous implémentons une fonction integrale_analytique permettant de calculer la valeur exacte de l'intégrale du polynôme sur l'intervalle [a, b]. Cette solution analytique servira de référence pour évaluer la précision des méthodes numériques.

Ensuite, nous développons les fonctions pour tous les méthodes proposées, qui repose sur une discrétisation de l'intervalle à l'aide d'une boucle for pour approximer l'intégrale. Cette méthode, bien que simple, offre une première approximation dont nous analyserons les limites.

L'étude de la convergence est réalisée en comparant les résultats numériques à la solution exacte pour différents nombres de segments. Cette analyse permet de quantifier l'erreur commise et d'observer comment celle-ci évolue avec l'augmentation de la finesse de la discrétisation.

Nous évaluons les performances des différentes méthodes en mesurant leur temps d'exécution grâce à la bibliothèque timeit. Cette étape fournit des critères objectifs pour comparer l'efficacité des algorithmes, en plus de leur précision. Finalement Pour améliorer la performance, on a recodé l'intégration avec NumPy (vectorisation)

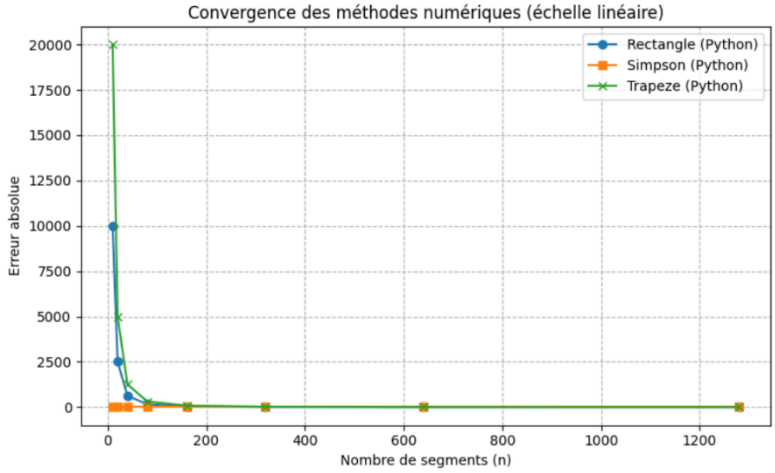
Résultats

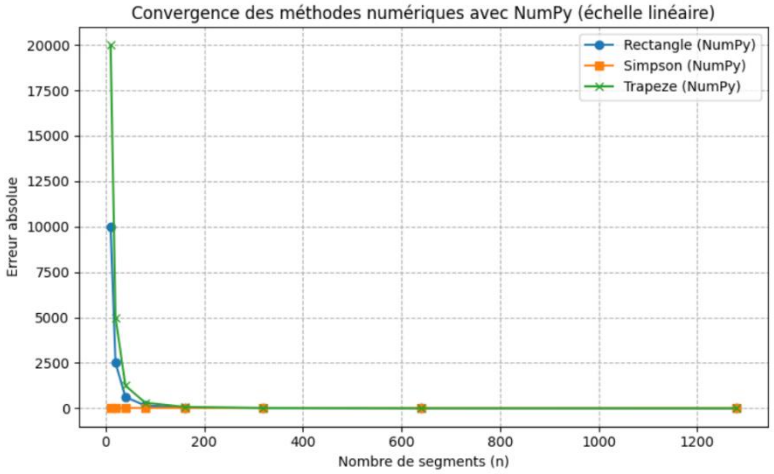
Paramètres utilisés : $f(x) = p_1 + p_2 x + p_3 x^2 + p_4 x^3$

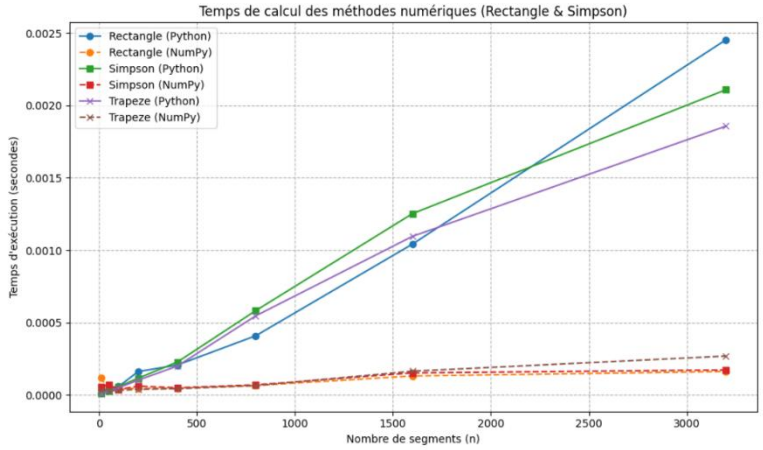
- Coefficients du polynôme: p1 = 26, p2 = 36, p3 = 12, p4 = 7
- Intervalle d'intégration: a = -50, b = 50

Méthode	Nombre de segments (n)	Intégrale numérique	Intégrale analytique	Erreur absolue	Temps d'exécution (s)
Python de base	10	992 600.0	1 002 600.0	10 000.0	0.000005
NumPy	10	992600.0	1 002 600.0	10 000.0	0.00009060
Python de base	100	1002500.0	1 002 600.0	100.00	0.000031
NumPy	100	1002500.0	1 002 600.0	100.00	0.000117
Python de base	1000	1002598.99	1 002 600.0	1.00	0.000253
NumPy	1000	1002599.0	1 002 600.0	1.00	0.0001859
Python de base	10000	1002599.99	1 002 600.0	0.0099	0.002536
NumPy	10000	1002599.99	1 002 600.0	0.01	0.0006569
Python de base	100000	0.0001	1 002 600.0	0.0001	0.025415
NumPy	100000	1002599.9999	1 002 600.0	0.0001	0.0061421

Tableau des résultats pour la méthode des rectangles et la méthode des rectangles vectorisée avec Numpay








```
=== Resultats de toutes les methodes ===
Rectangle Python     → I = 992600.000000, erreur = 1.000e+04, durée = 0.0000136000s
Simpson Python       → I = 1002600.000000, erreur = 0.000e+00, durée = 0.0000111000s
Trapeze Python       → I = 1022600.000000, erreur = 2.000e+04, durée = 0.0000079000s
Rectangle NumPy      → I = 992600.000000, erreur = 1.000e+04, durée = 0.0001124000s
Simpson NumPy        → I = 1002600.000000, erreur = 0.000e+00, durée = 0.0000496000s
Trapeze NumPy        → I = 1022600.000000, erreur = 2.000e+04, durée = 0.0000316000s
```

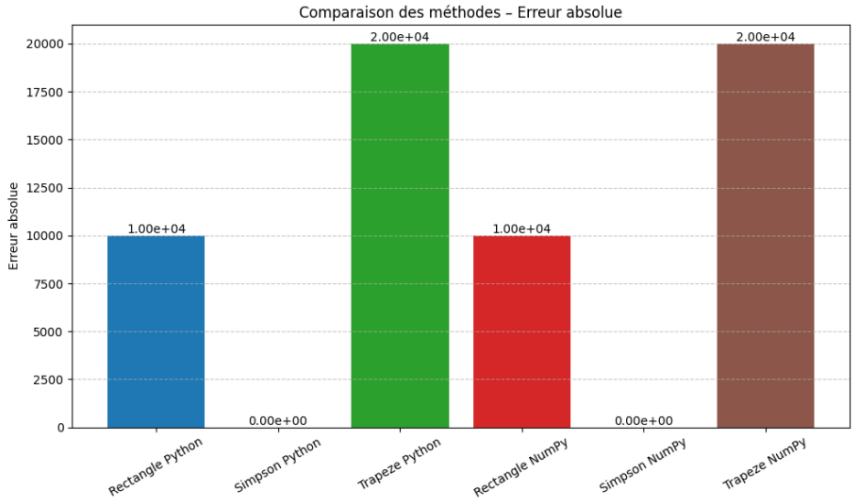
Analyse et conclusions des résultats

Précision des méthodes d'intégration

Parmi les différentes méthodes étudiées, la méthode de Simpson s'est révélée la plus précise. Cette précision découle de sa capacité à intégrer exactement les polynômes de degré inférieur ou égal à trois, ce qui est en parfaite adéquation avec la fonction test utilisée. Elle est particulièrement adaptée aux fonctions lisses ou polynomiales.

La méthode des trapèzes présente une précision intermédiaire. Elle repose sur une approximation linéaire de la courbe, ce qui peut s'avérer satisfaisant pour des fonctions linéaires par morceaux, mais entraîne une perte de précision pour les fonctions plus complexes. La méthode des rectangles, quant à elle, s'avère la moins précise. L'approximation constante par segment ne capture pas efficacement la variation de la fonction. Elle peut toutefois servir à obtenir une estimation rapide lorsque la précision n'est pas critique.

Il est important de souligner que l'utilisation de NumPy n'améliore pas la précision des résultats : les erreurs obtenues avec les implémentations vectorisées sont identiques à celles des versions en Python pur. Toutefois, NumPy permet une optimisation notable du temps d'exécution pour les calculs intensifs.



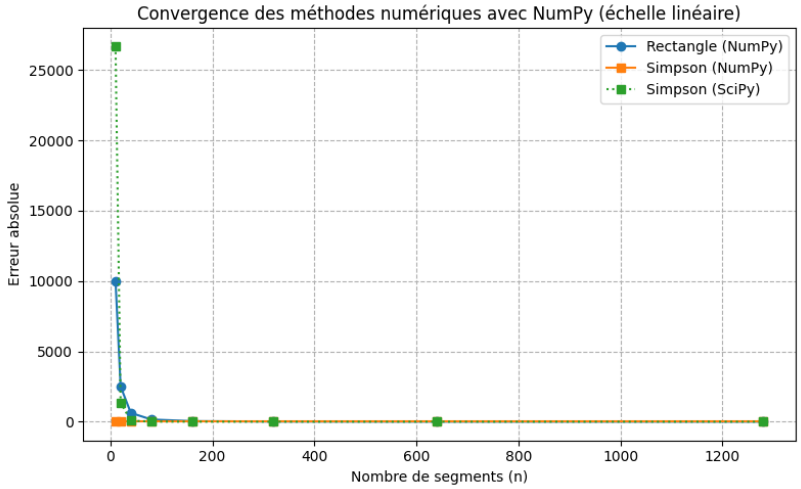
Performances en temps d'exécution

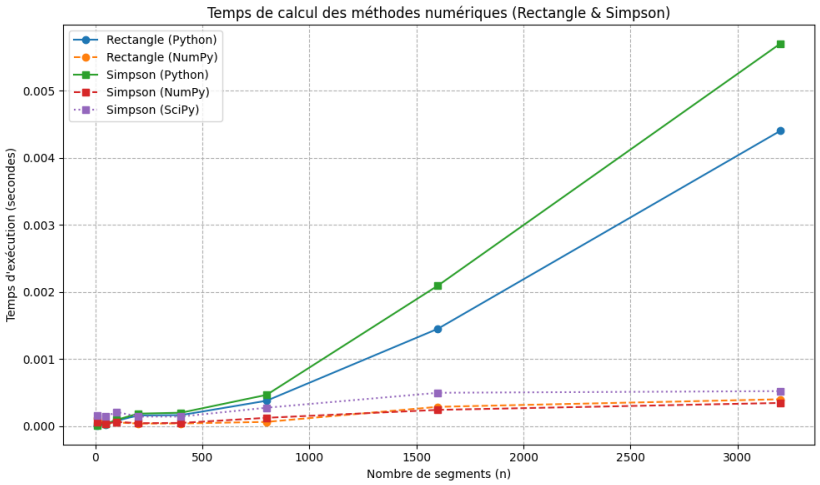
Les résultats montrent que les versions en Python de base sont plus rapides pour de petites valeurs de n, car elles ne subissent pas la surcharge associée à la gestion des tableaux NumPy. En revanche, NumPy devient avantageux lorsque le nombre de segments augmente, grâce à sa capacité à effectuer des opérations vectorielles massivement parallèles. Concernant la rapidité d'exécution, l'ordre observé va généralement de la méthode des trapèzes (en Python pur) comme la plus rapide, suivie de Simpson et des rectangles. Les versions NumPy deviennent compétitives voire plus rapides que les versions classiques à partir de tailles de segments élevées.

Convergence des méthodes

L'étude de la convergence confirme que toutes les méthodes tendent vers la valeur exacte de l'intégrale lorsque le nombre de segments augmente. La méthode de Simpson se distingue par sa convergence rapide : même pour un nombre modéré de sous-intervalles, elle fournit une solution exacte dans notre cas. En comparaison, les méthodes des trapèzes et des rectangles nécessitent un nombre de segments plus élevé pour atteindre un niveau d'erreur similaire. Les graphiques d'erreur illustrent cette tendance : la courbe associée à Simpson reste pratiquement

plate, traduisant une erreur quasi nulle, tandis que les erreurs des rectangles et des trapèzes diminuent plus lentement, de manière quasi linéaire en fonction de n.





L'analyse comparative des méthodes d'intégration met en évidence que la méthode de Simpson, qu'elle soit implémentée avec NumPy ou SciPy, est la plus précise pour les fonctions polynomiales de degré 3. Elle atteint une erreur quasi nulle dès un nombre modéré de sous-intervalles, ce qui confirme sa capacité théorique à intégrer exactement ce type de fonction. En revanche, la méthode des rectangles, même vectorisée, présente une erreur élevée et nécessite un nombre de segments beaucoup plus grand pour converger vers une solution acceptable.

Sur le plan des performances, les versions Python de base sont plus rapides pour les petits n, tandis que NumPy devient plus avantageux pour des calculs plus lourds grâce à la vectorisation. SciPy, bien que légèrement plus lent, est recommandé pour son adaptabilité à des fonctions plus complexes. Enfin, il est important de noter que Simpson est particulièrement précise sur de grands intervalles, mais peut perdre en efficacité sur de très petits intervalles à cause des erreurs numériques. Le choix de la méthode doit donc être guidé par le compromis entre précision, complexité de la fonction et coût computationnel.

Conclusion générale

Les expériences réalisées confirment les résultats attendus en théorie. La méthode de Simpson est la plus précise et converge rapidement, surtout pour des fonctions polynomiales. Elle fournit des résultats exacts avec un nombre modéré de sous-intervalles lorsque l'intervalle d'intégration est large, ce qui en fait un choix particulièrement judicieux pour des domaines étendus. En revanche, dans des intervalles très petits, sa précision peut se dégrader légèrement en raison de la sensibilité numérique accrue et des effets d'arrondi. Les méthodes des trapèzes et des rectangles, bien que moins précises, restent utiles pour des estimations rapides ou des fonctions moins régulières. Par ailleurs, les versions vectorisées avec NumPy n'apportent pas de gain en précision, mais deviennent nettement plus performantes en termes de temps d'exécution lorsque le nombre de segments est élevé. NumPy s'impose donc comme un outil puissant pour les intégrations numériques à grande échelle, bien qu'il introduise une légère surcharge pour les petits calculs.