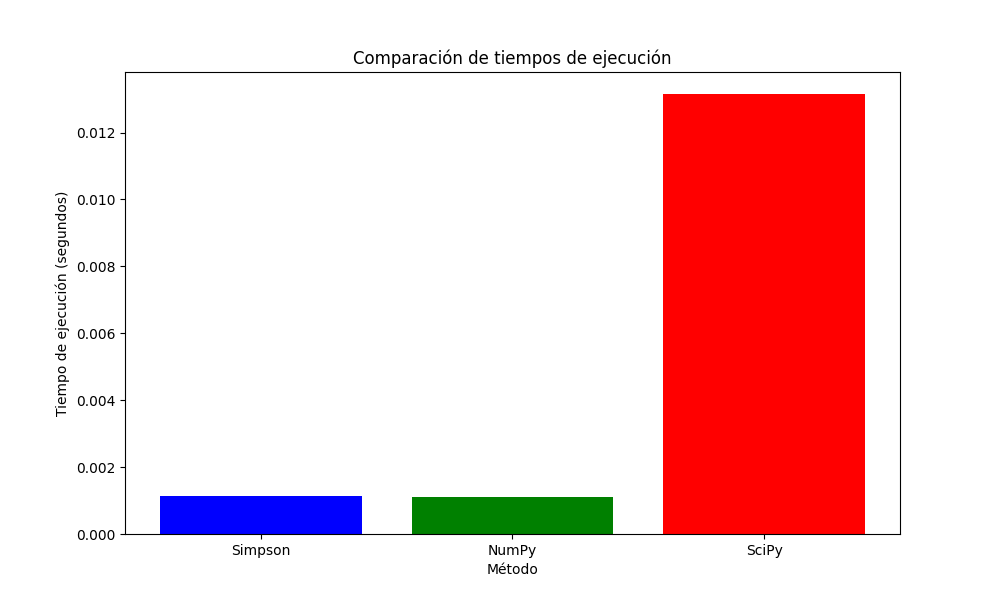
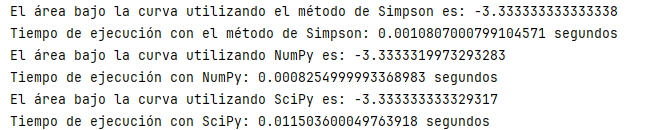
**Explication des temps de calcul des 3 méthodes**

En comparant les temps de calcul des trois implémentations différentes pour calculer l'aire sous la courbe d'un polynôme du troisième ordre, nous observons les différences clés suivantes :

À l'aide du code, 3 procédures ont été effectuées qui correspondent à l'aire sous la courbe

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Simpson** | **Numpy** | **SciPy** |
| **Description** | La méthode implémente manuellement la règle de Simpson, en divisant l'intervalle d'intégration en sous-intervalles et en appliquant la formule pour chaque paire de sous-intervalles. | cette méthode utilise numpy.trapz pour l'intégration, qui implémente la règle du trapèze, à des fins de comparaison et de référence. | La méthode utilise scipy.integrate.simpson, qui est une implémentation optimisée de la règle Simpson fournie par SciPy. |
| **Avantages** | C’est une implémentation manuelle qui permet un contrôle détaillé du calcul et peut être un bon exemple pour comprendre le fonctionnement | NumPy est optimisé pour les opérations grands tableaux, ce qui le rend beaucoup plus rapide que les boucles de forme manuelle. | La fonction simpson dans SciPy implémente efficacement la règle de Simpson car il a été conçu à cet effet précis. |
| **Inconvénients** | Peut-être plus lent que les implémentations optimisées dans les bibliothèques scientifiques en raison du manque d'optimisations | Il peut y avoir de légères différences de précision par rapport à une implémentation précise de Simpson car il n'utilise pas spécifiquement la règle de Simpson mais la règle du trapèze. | Il faut connaître et utiliser les librairies python |
| **Runtime** | C'est généralement la plus lente des trois méthodes en raison de l’implémentation de calcul manuelle. | Il est généralement plus rapide que la méthode manuelle de Simpson en raison des optimisations de NumPy. | C'est généralement la plus rapide et la plus précise des trois méthodes en raison des optimisations internes de SciPy. |

Le graphique de temps d’exécution généré avec matplotlib permettra de visualiser les différences :



**Observations :**

Selon la théorie des avantages de chaque méthode, la précision du résultat peut varier de petites quantités à haute résolution. En ce qui concerne les temps d'exécution, on peut observer que la méthode ScyPy est la plus retardée, en cours d'investigation, cela peut être dû aux raisons suivantes :

1. Vitesse du processeur :
2. Architecture du processeur
3. Caché de la CPU
4. Nombre de cœurs
5. Système d’exploitation
6. Gestion de l'énergie

De plus, le calcul de l'aire d'une équation du troisième degré n'est pas une opération si complexe, le temps de chargement de la bibliothèque ScyPy est ajouté au temps mesuré par la fonction timeit.

Références bibliographiques :

([SciPy Documentation] (<https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.simpson.html)>)

([NumPy Documentation] (<https://numpy.org/doc/stable/reference/generated/numpy.trapz.html)>)