TD 4

Tom Wozniak

2 octobre 2025

1 Étude d'un nouveau code ADRS

Nous avons commencé cette séance en étudiant un code Python permettant de résoudre l'équation ADRS. Ce code a la particularité d'affiner le maillage au fur et à mesure, en calculant la métrique locale via la courbure locale de la dérivée seconde en espace. Dans le code, la métrique est définie par

$$\text{métrique}_j = \min\left(\frac{1}{h_{min}}, \max\left(\frac{1}{h_{max}}, \frac{|T_{xx}|}{err}\right)\right)$$

où err désigne le seuil de tolérance, qui vaut initialement 0.03.

Nous avons ensuite modifié le code pour y implémenter la loi suivante

$$(s_{i+1} - s_i)\frac{\mathcal{M}_{i+1} + \mathcal{M}_i}{2} = 1$$

où $\mathcal{M}_i = \max\left(\min\left(\frac{1}{\varepsilon}|\partial_s^2 u_h(s_i)|, \frac{1}{h_{min}^2}, \frac{1}{h_{max}^2}\right)\right)$.

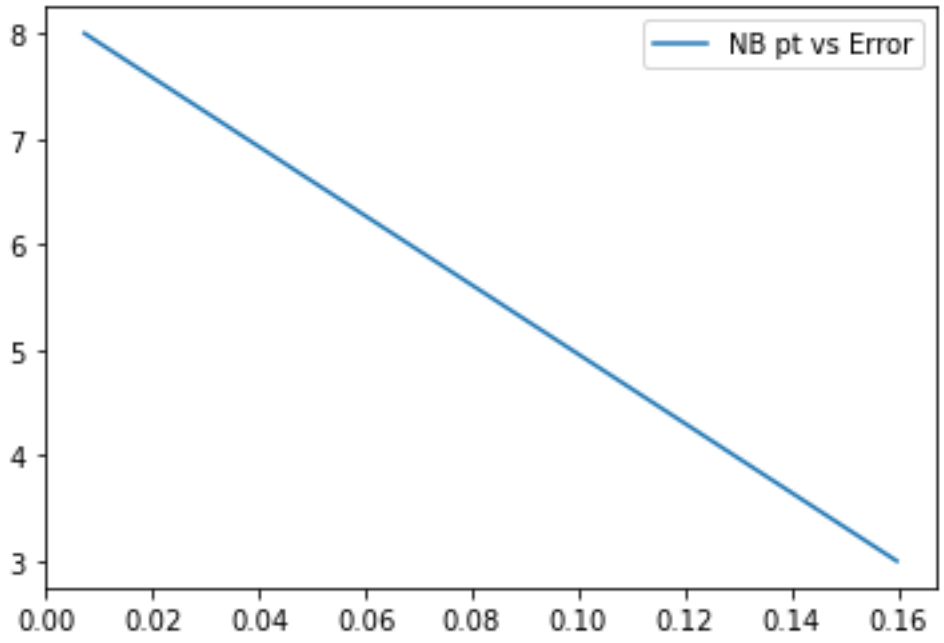

FIGURE 1 – Graphe montrant l'erreur par rapport au nombre de points

Ensuite, nous avons tracé la courbe de $NX(err)$ pour différentes valeurs de err et nous avons obtenue la figure suivante.

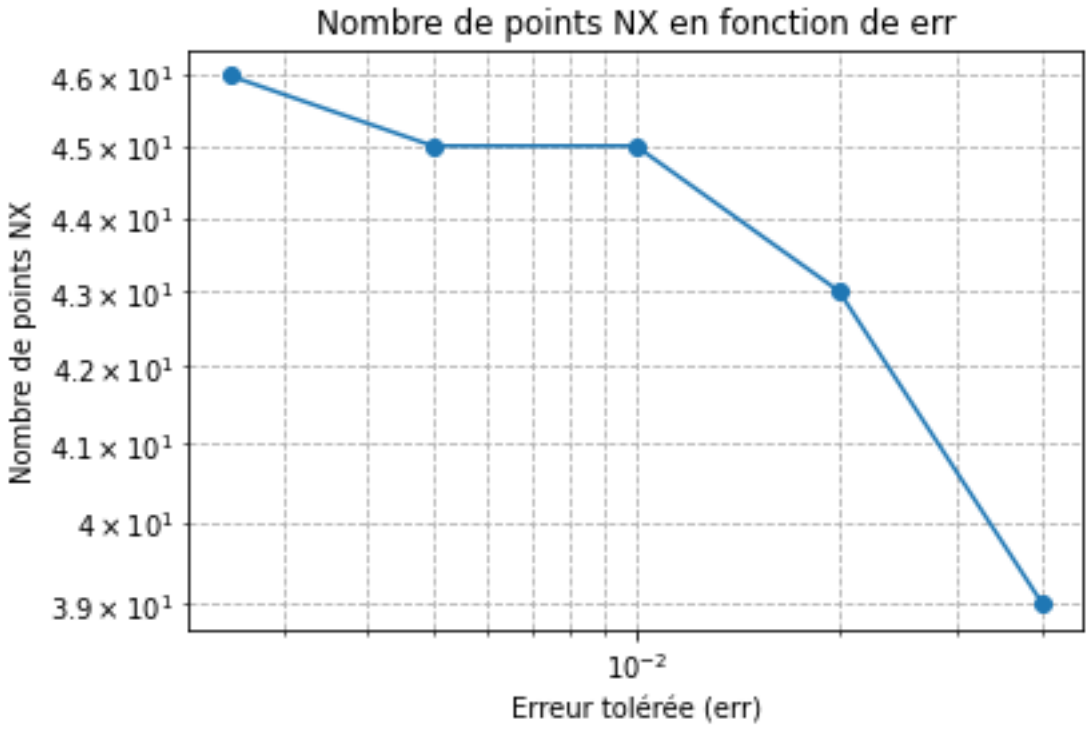

FIGURE 2 – Graphe de NX en fonction de err

On observe que la courbe de NX diminue lorsque l'erreur augmente, ce qui était prévisible. De plus, pour l'instant, l'itération d'adaptation s'arrête lorsqu'on a une assez bonne convergence en termes de nombre de points, c'est-à-dire que l'on continue tant que la différence du nombre de points entre deux itérations consécutives est supérieure à 2 (et que le nombre max d'itérations n'est pas atteint).

Ensuite, nous avons modifié le code pour y introduire un critère portant sur le nombre de points de maillage et l'erreur L^2. Enfin, nous avons cherché à localiser la contraction dans ce code. Elle se situe au niveau du fait que res/res_0 tende vers 0, ce qui montre que la solution converge, et cette condition est assurée par le choix du pas de temps dt.

2 Le code adrs insta.py

Nous nous sommes ensuite intéressés à un nouveau code permettant de résoudre l'équation ADRS 1D en régime transitoire. Ce code utilise une méthode du type Runge-Kutta explicite. Nous avons d'abord cherché à visualiser l'erreur en fin de maillage et à la moitié du maillage.

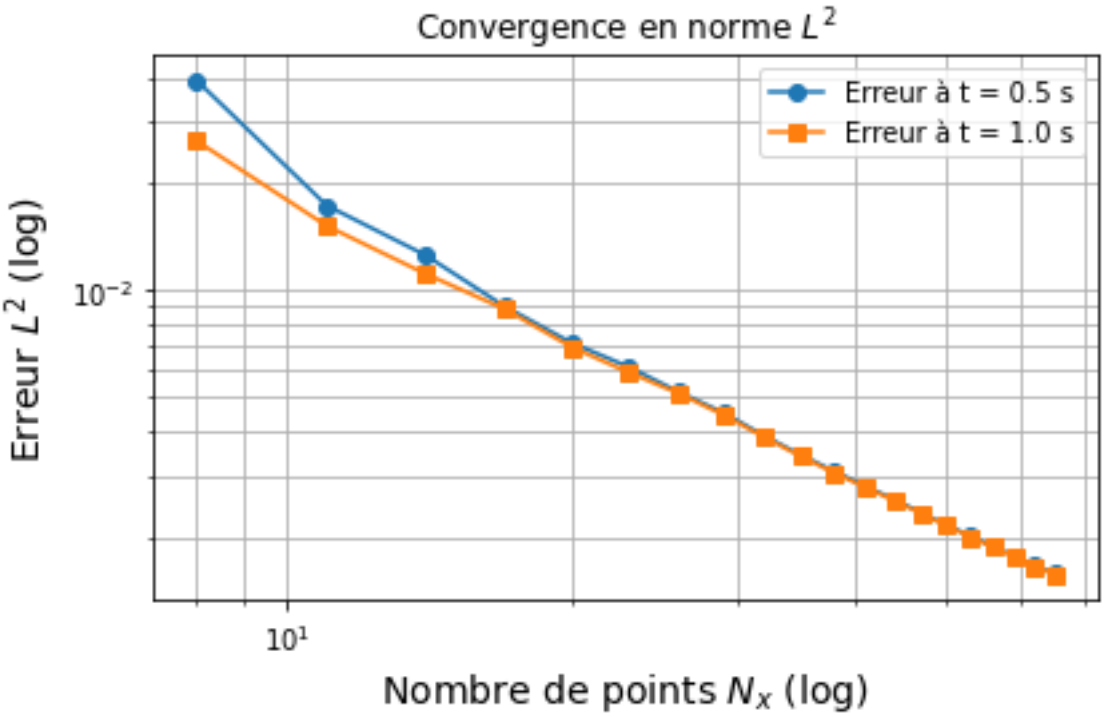

FIGURE 3 – Graphe montrant l'erreur en norme L^2 à différentes valeurs de t

Ensuite, nous avons modifié le code pour visualiser l'évolution de l'erreur au centre du domaine pour des ordres différents de la méthode Runge-Kutta, plus précisément pour les ordres 1,2,3 et 4.

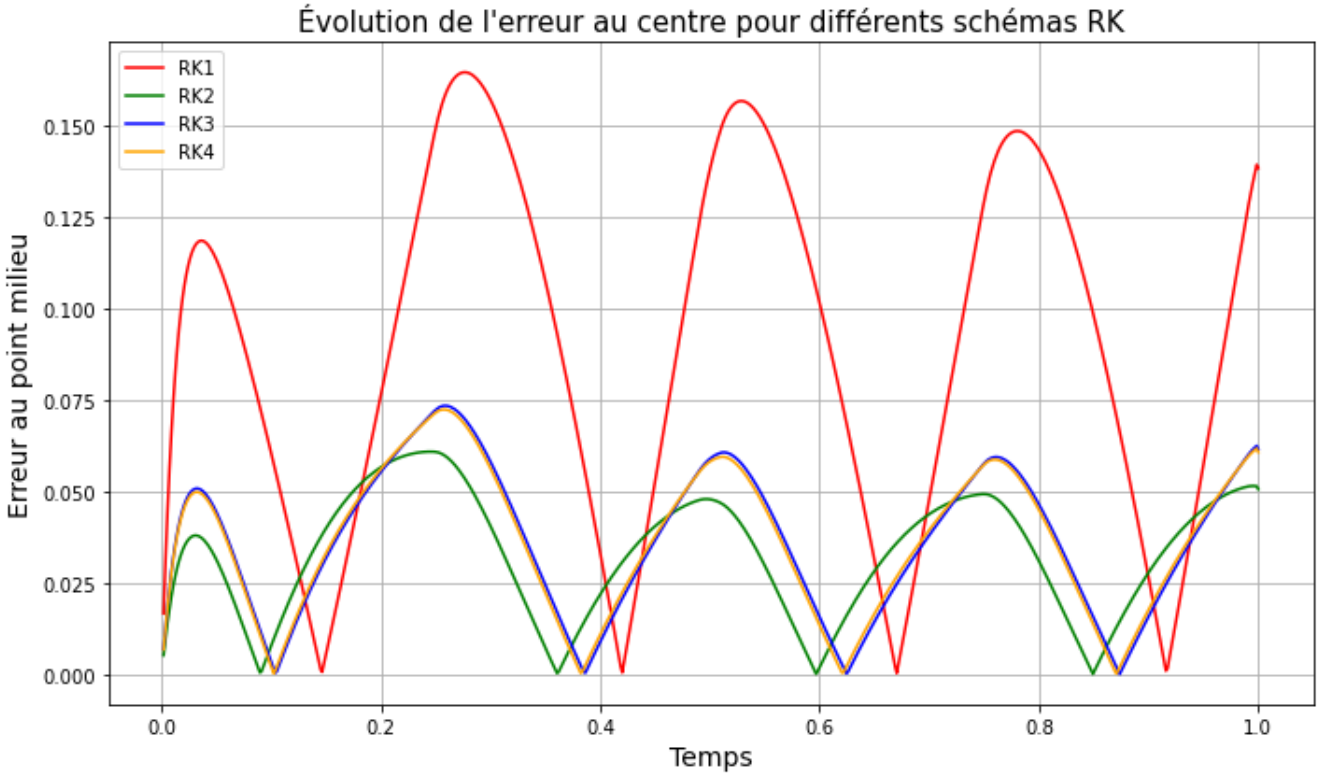

FIGURE 4 – Graphe montrant l'évolution de l'erreur au centre pour différents schémas Runge-Kutta