Rapport de stage d'immersion en entreprise

Période : 01/07/2023 - 31/08/2023

sous le thème

Résolution numérique de l'équation de transport

Réalisé par : **Omnia LAHDHIRI**

Organisme : ESPRIT

Encadré par : Mme. Mounira GHARSALLI

Adresse : 1, 2 rue Ampère – 2084 – Pôle Technologique – El Ghazela

Remerciement

Exprimer sa reconnaissance à ceux qui ont collaboré de loin ou de près à l'élaboration de ce projet est un devoir agréable.

Je présente donc mes vifs et profonds respects et reconnaissances à mon encadrante Mme. Mounira GHARSALLI, pour avoir accepté de me superviser et me diriger lors de la réalisation de ce travail, pour ses conseils et ses compétences qui m'ont considérablement servi pour mener à bien mon stage.

Intitulé du projet : Résolution numérique de l'équation de transport

Type de projet : Modélisation

Objectifs du projet :

L'objectif de ce projet est d'étudier une solution simple pour modéliser le transport d'une espèce chimique d'un fluide.

Voici les différents objectifs du projet :

- Établir le modèle physique qui consiste en une équation : $\frac{\partial u(x,t)}{\partial t} + c\frac{\partial u(x,t)}{\partial x} = f(x,t)$
- Étudier les méthodes Runge-Kutta d'ordre 2 et 4 pour approximer ces solutions
- Implémenter ces méthodes en Python
- Déterminer des applications de ce modèle.

Table des matières

Introduction Générale

Ce rapport porte sur l'étude des équations de transport des espèces chimiques dans les liquides en mouvement. Ce problème à l'interface entre la chimie et la mécanique des fluides revêt une grande importance dans divers domaines scientifiques et technologiques, comme la chimie de l'environnement, la modélisation des procédés industriels ou encore la recherche en science des matériaux.

L'objectif principal de ce projet est de déterminer les changements de concentration d'espèces chimiques spécifiques dans des liquides en mouvement en fonction de la localisation spatiale et du temps.

Pour atteindre cet objectif, deux méthodes numériques de résolution d'équations différentielles, la méthode Runge-Kutta du second ordre et la méthode Runge-Kutta du quatrième ordre, sont mises en œuvre. Ces méthodes numériques sont connues pour leur capacité à fournir des résultats précis et robustes.

Ce projet de stage est une opportunité valorisante d'approfondir ses connaissances en modélisation et méthodes numériques des phénomènes de transport tout en contribuant à l'avancement des connaissances dans les domaines de la chimie et de la mécanique des fluides.

Chapitre 1

Résolution numérique de l'équation de transport

1 Méthode des différences finies

1.1 Principe

Le but est d'approximer les dérivées partielles par des quotients finis de la forme :

$$\frac{\partial u}{\partial t} \approx \frac{u(x,t+h)-u(x,t)}{h}$$

avec h petit et fixé.

On verra qu'il existe plusieurs méthodes possibles pour calculer cette approximation.

Pour réaliser ce travail, on doit d'abord discrétiser l'espace-temps. Pour cela, on prend :

- N le nombre de points d'espace,
- T la durée,
- h le pas d'espace,
- τ le pas de temps.

1.2 Discrétiser l'espace

L'intervalle d'origine est [0,1], on choisit donc le nombre de points N ∈ N et on a $h = \frac{1}{N}$

On définit alors la suite des points de l'espace : $x_n = x_0 + nh = nh.$

On a donc : $\begin{cases} x_0 = 0 \\ x_N = 1 \end{cases}$

1.3 Discrétiser le temps

L'intervalle d'origine est $\mathbb{R}^+$

On ne peut pas faire comme pour l'espace. On fixe donc τ petit. Si possible on choisit τ dont la représentation machine est exacte. On étudie alors l'équation à T fixé.

On définit de même, la suite des points de temps : $t_p = p\tau$.

On peut alors poser les deux suites suivantes : $u_n^p = u(x_n, t_p)$ et $f_n^p = f(x_n, t_p)$

On va maintenant s'intéresser à deux méthodes d'approximation pour la résolution numérique de l'équation.

2 Méthode Runge-Kutta d'ordre 2

2.1 Approximation des dérivées partielles

On utilise les développements de Taylor suivants :

$$u(x_n, t_{p+1}) = u(x_n, t_p) + \tau \frac{\partial u}{\partial t}(x_n, t_p) + o(\tau)$$
$$u(x_{n-1}, t_p) = u(x_n, t_p) - h \frac{\partial u}{\partial x}(x_n, t_p) + o(h)$$

On obtient donc :

$$\frac{\partial u}{\partial t} \approx \frac{u_n^{p+1} - u_n^p}{\tau} \quad \text{et} \quad \frac{\partial u}{\partial x} \approx \frac{u_n^p - u_{n-1}^p}{h}$$

Les conditions initiales donnent les valeurs des $u_0^p = u_0(t_p) = u_0(p\tau)$ et les conditions aux limites donnent $u_n^0 = \varphi(x_n) = \varphi(nh)$.

2.2 Détermination de l'équation

L'équation de transport est donnée par : $\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = f(x,t)$

avec les conditions initiales : $u_0^p = u_0(t_p) = u_0(p\tau)$ si $x < ct$
$u_n^0 = \varphi(x_n) = \varphi(nh)$ si $x \geq ct$

Pour résoudre l'équation en appliquant la méthode de Runge-Kutta d'ordre 2 (RK2), on décompose le problème en trois étapes successives :

2.2.1 RK2 associée à l'évolution de la solution *u* dans la direction de l'espace

Soient $k_{1_{x\_rk2}}$, $k_{2_{x\_rk2}}$ les composantes de RK2 associée à l'évolution de *u* par rapport à la variable spatiale *x* respectivement à la position actuelle et à un point intermédiaire avec :

$$k_{1_{x\_rk2}} = c\tau u(x_n, t_p)$$

$$k_{2_{x\_rk2}} = c\tau u(x_n, t_p) + \frac{k_{1_{x\_rk2}}}{2}$$

$$u_{x\_rk2} = u(x_n, t_p) + \frac{k_{1_{x\_rk2}} + k_{2_{x\_rk2}}}{2}$$

2.2.2 RK2 associée à l'évolution de la solution *u* dans la direction du temps

Soient $k_{1_{t\_rk2}}, k_{2_{t\_rk2}}$ les composantes de RK2 associée à l'évolution de *u* par rapport à la variable temporelle respectivement à partir du temps t actuel et à un point intermédiaire avec :

$$k_{1_{t\_rk2}} = \tau f(x_n, t_p) - \frac{c\tau}{2h}\left(u(x_n, t_p) - u(x_{n-1}, t_p)\right)$$

$$k_{2_{t\_rk2}} = \tau f(x_n, t_p + \frac{\tau}{2}) - \frac{c\tau}{2h}\left(u(x_n, t_p) + \frac{k_{1_{x\_rk2}}}{2} - u(x_{n-1}, t_p) - \frac{k_{1_{t\_rk2}}}{2} \right)$$

$$u_{\frac{t}{2}rk2} = \frac{u(x_n, t_p) + u(x_{n-1}, t_p)}{2} + k_{2_{t\_rk2}}$$

2.2.3 Application des résultats de RK2 pour résoudre toute l'équation

D'après ce qui précède, on obtient :

$$k_{1_{rk2}} = \tau f(x_n, t_p) - \frac{c\tau}{2h}\left(u_{x\_rk2} - u(x_{n-1}, t_p)\right)$$

$$k_{2_{rk2}} = \tau f(x_n, t_p + \frac{\tau}{2}) - \frac{c\tau}{2h}\left(u_{x\_rk2} - u(x_{n-1}, t_p) - \frac{k_{1_{rk2}}}{2}\right)$$

$$u_{rk2}(x_n, t_p) = u(x_n, t_p) + \frac{k_{1_{rk2}} + k_{2_{rk2}}}{2}$$

Remarque : Dans le cas de l'équation sans terme source, on a $\forall\ (n,p) \in \mathbb{N}^2, f(x_n, t_p) = 0$

3 Méthode Runge-Kutta d'ordre 4

3.1 Détermination de l'équation

Cette méthode est très similaire à celle de RK2. La seule différence réside dans le nombre des pentes calculées à chaque étape de la résolution de l'équation de transport.

3.1.1 RK4 associée à l'évolution de la solution *u* dans la direction de l'espace

Soient k_{1x_rk4}, k_{2x_rk4}, k_{3x_rk4} et k_{4x_rk4} les composantes de RK4 associée à l'évolution de *u* par rapport à la variable spatiale x à différentes étapes du calcul avec :

$$k_{1x\_rk4} = c\tau u(x_n, t_p)$$

$$k_{2x\_rk4} = c\tau\left(u(x_n, t_p) + \frac{k_{1x\_rk4}}{2}\right)$$

$$k_{3x\_rk4} = c\tau\left(u(x_n, t_p) + \frac{k_{2x\_rk4}}{2}\right)$$

$$k_{4x\_rk4} = c\tau\left(u(x_n, t_p) + k_{3x\_rk4}\right)$$

3.1.2 RK4 associée à l'évolution de la solution *u* dans la direction du temps

Soient k_{1t_rk4}, k_{2t_rk4}, k_{3t_rk4} et k_{4t_rk4} les composantes de RK4 associée à l'évolution de *u* par rapport à la variable temporelle *t* à différentes étapes du calcul avec :

$$k_{1t\_rk4} = \tau f(x_n, t_p) - \frac{c\tau}{2h}\left(u(x_n, t_p) - u(x_{n-1}, t_p)\right)$$

$$k_{2t\_rk4} = \tau f(x_n, t_p + \frac{\tau}{2}) - \frac{c\tau}{2h}\left(u(x_n, t_p) + \frac{k_{1x\_rk4}}{2} - u(x_{n-1}, t_p) - \frac{k_{1x\_rk4}}{2}\right)$$

$$k_{3t\_rk4} = \tau f(x_n, t_p + \frac{\tau}{2}) - \frac{c\tau}{2h}\left(u(x_n, t_p) + \frac{k_{2x\_rk4}}{2} - u(x_{n-1}, t_p) - \frac{k_{2t\_rk4}}{2}\right)$$

$$k_{4t\_rk4} = \tau f(x_n, t_p + \tau) - \frac{c\tau}{2h}\left(u(x_n, t_p) + k_{3x\_rk4} - u(x_{n-1}, t_p) - k_{3t\_rk4}\right)$$

3.1.3 Application des résultats de RK4 pour résoudre toute l'équation

D'après ce qui précède, on obtient :

$$u_{rk4}(x_n, t_p) = u(x_n, t_p) + \frac{k_{1t\_rk4} + 2k_{2t\_rk4} + 2k_{3t\_rk4} + k_{4t\_rk4}}{6}$$

Remarque : Dans le cas de l'équation sans terme source, on a $\forall\ (n, p) \in \mathbb{N}^2,\ f(x_n, t_p) = 0$

4 Comparaison

Les méthodes Runge-Kutta d'ordre 2 et Runge-Kutta d'ordre 4 sont testés sur 2 exemples différents

Premier exemple : On prend comme condition initiale $u_0(x) = \cos(2x)$, et comme condition aux limites $\varphi(t) = e^{t^2}$. On prend également $c = 1$ et $f(x,t) = \sin(2xt)$.

Deuxième exemple : On prend comme condition initiale $u_0(x) = \sin(x)$, et comme condition aux limites $\varphi(t) = 2t - 1$. On prend également $c = 1$ et $f(x,t) = e^{t^2} + x$.

4.1 Outils créés

On a fait différents petits programmes implémentés en Python pour traiter les résultats obtenus précédemment.

- Il y a un programme permettant de tracer les courbes des méthodes RK2 et RK4, et la courbe analytique pour un temps t donné. Ces courbes représentent donc la concentration en fonction de la position x.
- Il y a un programme qui calcule la somme des écarts ponctuels entre les courbes RK2 et RK4, et la courbe analytique. Cela permet donc de comparer les écarts entre ces 2 méthodes et la solution théorique en les traçant à un temps t donné.

Ces outils prennent la forme de scripts Python qui ont été présenté en annexe.

4.2 Comparaison des deux méthodes

Premier exemple

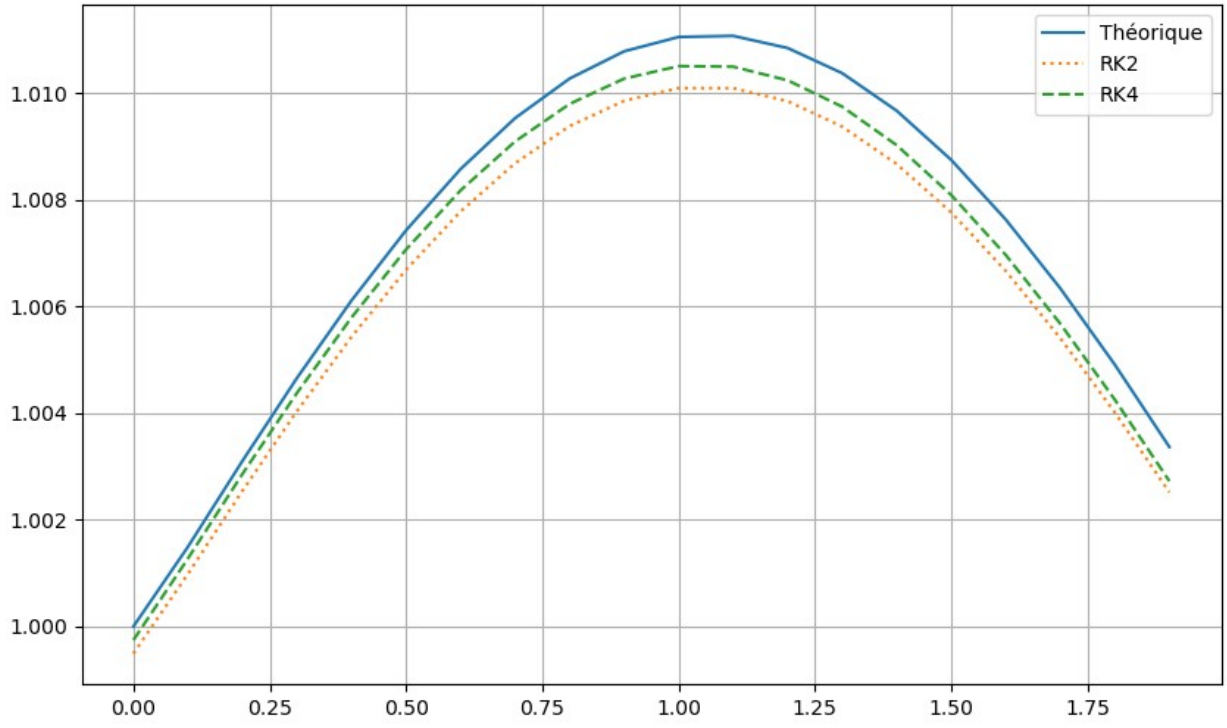

Figure 1.1 – Solution de l'équation de transport avec terme source en fonction de x à t=0,75

Grâce au programme animation.py, on observe que la courbe de la méthode RK4 est très proche de celle de la solution théorique comparant à la courbe de la méthode RK2. Ces deux dernières ont les mêmes variations, mais les écarts ponctuels de la courbe de la méthode RK2 sont relativement plus remarquables que ceux de la méthode RK4 comme l'illustre la figure ci-dessous :

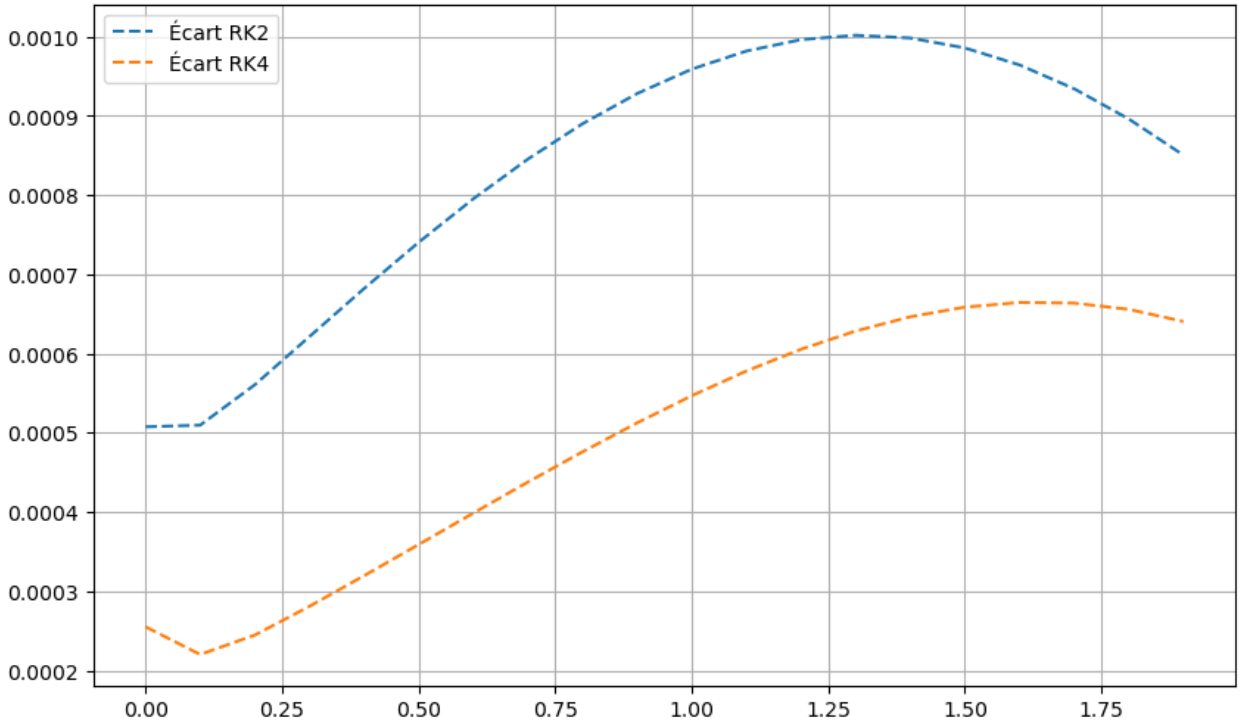

Figure 1.2 – Comparaison des méthodes de résolution de l'équation de transport avec terme source et calcul de l'écart à t=0,75

Sans terme source, on constate que la solution théorique garde sa stabilité, tandis que les deux méthodes produisent des solutions légèrement variables : La solution de la méthode RK4 reste toujours plus exacte que celle de la méthode RK2 en comparant leurs écarts ponctuels.

Conclusion : Sur ce premier exemple, on remarque que les solutions de la méthode RK4 sont très proches de celles théoriques, alors que les solutions de la méthode RK2 sont légèrement différentes, bien que les variations des courbes représentant les solutions des 2 méthodes et les solutions théoriques soient les mêmes.

Deuxième exemple

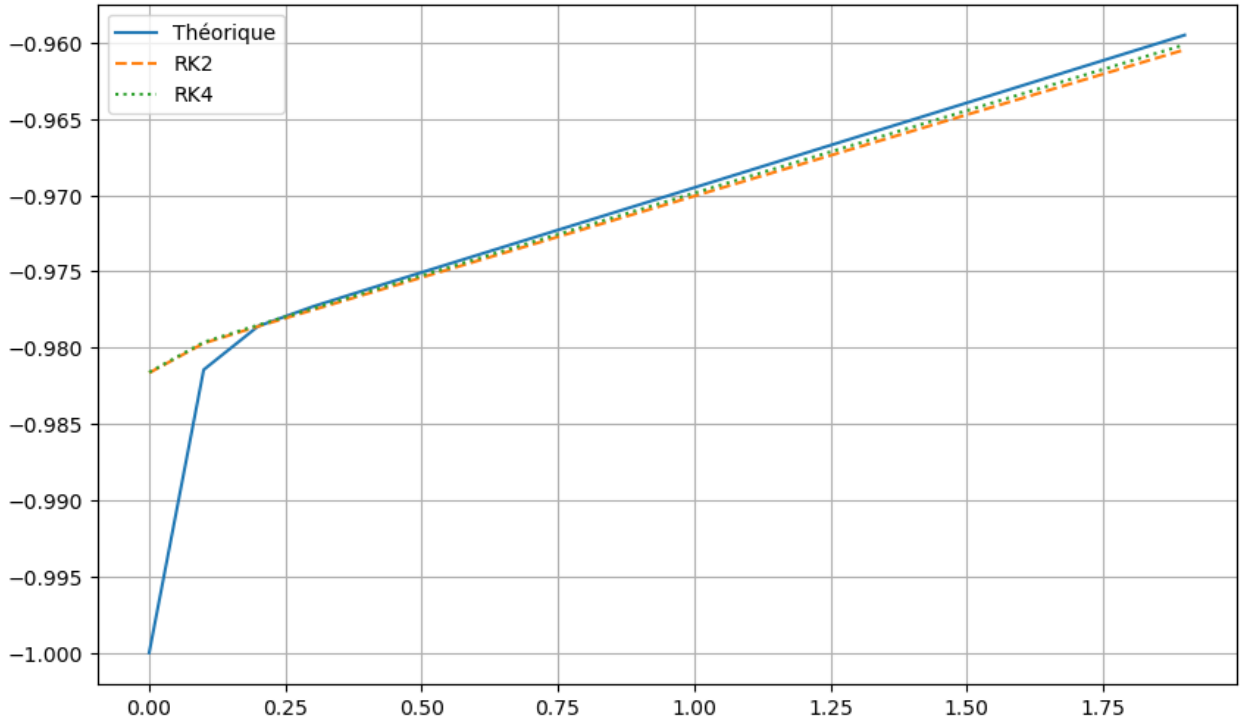

Figure 2.1 – Solution de l'équation de transport avec terme source en fonction de x à t=0,75

Dans cet exemple, on observe que les courbes des méthodes RK2 et RK4 sont presque superposées et très proches de la courbe des solutions théoriques. Selon cette représentation graphique, il est difficile de bien comparer la précision des 2 méthodes notamment sur l'intervalle [0.00 , 0.50].

On aura donc besoin de calculer les écarts ponctuels de chacune d'elles et tracer leurs courbes pour mieux interpréter les résultats obtenus (voir la figure 2.2 sur la page suivante).

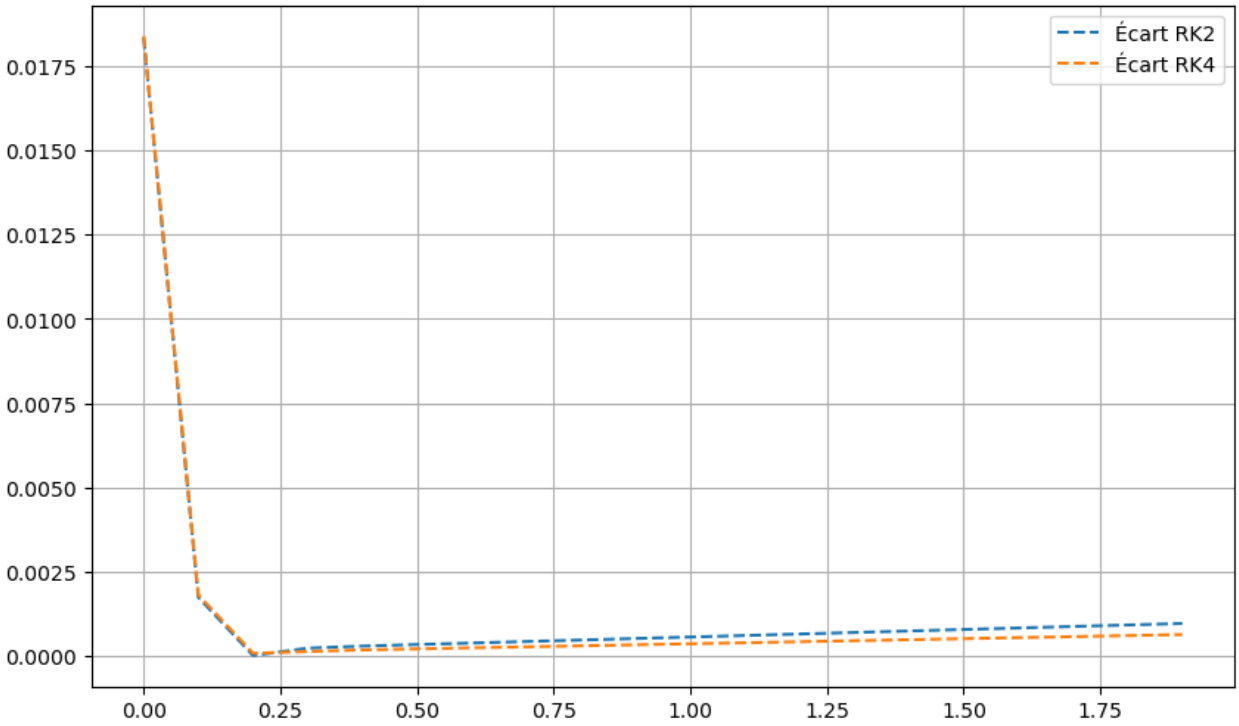

Figure 2.2 – Comparaison des méthodes de résolution de l'équation de transport avec terme source et calcul de l'écart à t=0,75

Conclusion : Dans cet exemple, les deux méthodes suivent quasiment la même trajectoire et approximent bien la solution théorique malgré que la méthode RK2 est relativement moins précise que la méthode RK4. Par contre, dans le cas de l'équation de transport sans terme source, la méthode RK4 est toujours la plus proche que celle de RK2 de la solution théorique.

Chapitre 2

Applications

1 Définition physique de l'équation de transport

L'équation de transport est une équation aux dérivées partielles qui décrit le transport d'une quantité physique dans un milieu. La quantité physique peut être la température, la concentration d'une substance ou la vitesse d'un fluide. Cette équation prend en compte les effets de la diffusion, de la convection et de la réaction chimique.

Elle est utilisée dans de nombreux domaines de la physique, de la chimie, de la météorologie et de l'ingénierie. Par exemple, l'équation de transport est utilisée pour modéliser la propagation de la chaleur, l'écoulement d'un fluide ou la propagation d'une onde acoustique dans l'air.

2 Application à une situation physique concrète

On peut prendre un exemple concret de l'équation de transport : **la diffusion d'une odeur dans l'air.** Dans notre cas, l'équation de transport décrit comment la concentration d'odeur évolue dans l'espace et dans le temps. Pour modéliser la diffusion d'une odeur, on va se servir de la fonction du temps suivante :

$$\varphi(t) = \begin{cases} C_0.e^{-\beta t} & \text{si } t \geq 0 \\ 0 & \text{si } t < 0 \end{cases}$$

avec :

C_0 : la concentration initiale de l'odeur

β : le paramètre de décroissance de la concentration de l'odeur

L'ajout du terme source est indispensable car il représente la libération continue d'odeur dans l'air au fil du temps.

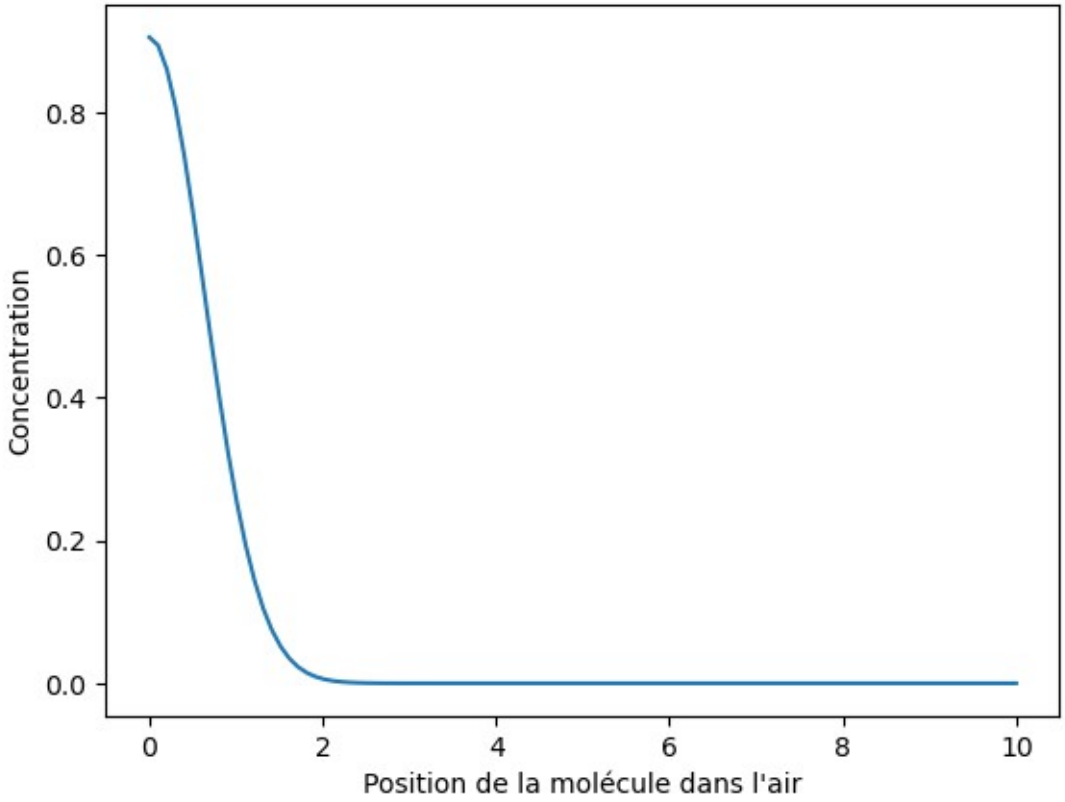

Figure 3.0 – Courbe de diffusion de l'odeur dans l'air
en fonction de x à t=2

La courbe de diffusion d'une odeur en fonction de la position x à un temps t=2, présente une caractéristique distinctive. La concentration d'odeur diminue de manière exponentielle avec la distance à la source d'introduction de l'odeur. Cela signifie que la concentration est plus élevée près de la source et diminue progressivement à mesure qu'on s'éloigne de celle-ci.

La courbe de diffusion d'odeur peut être décrite par une équation mathématique appelée loi de Fick. Cette loi montre que la concentration d'odeur diminue de façon exponentielle avec la distance à la source. Cette courbe peut être utilisée pour prédire la distribution de la concentration d'odeur dans l'espace. Cela peut être utile pour des applications telles que la conception de systèmes de ventilation et la gestion des odeurs industrielles.

On a utilisé les programmes expliqués dans le chapitre précédent pour calculer la théorie de cette expérience, reflétant ainsi le processus de dispersion caractéristique des odeurs dans un espace.

Chapitre 3

Annexes – Programmes réalisés

1 Premier exemple

1.1 Solution de l'équation de transport avec terme source en fonction de x à t=0,75

```
import numpy as np
import matplotlib.pyplot as plt

c = 1.0
h = 0.1
tau = 0.01
T = 2.0
N = 2.0
x_valeurs = np.arange(0, N, h)
t_valeurs = np.arange(0, T, tau)

def u0(x):
    return np.cos(2*x)

def phi(t):
    return np.exp(t**2)

def source(x, t):
    return np.sin(2*x*t)

udiff = np.zeros(len(x_valeurs))
urk2 = np.zeros(len(x_valeurs))
urk4 = np.zeros(len(x_valeurs))

for i in range(len(x_valeurs)):
    if x_valeurs[i] < c * t_valeurs[0]:
        udiff[i] = u0(x_valeurs[i])
        urk2[i] = u0(x_valeurs[i])
        urk4[i] = u0(x_valeurs[i])
    else:
        udiff[i] = phi(t_valeurs[0])
        urk2[i] = phi(t_valeurs[0])
```

```
    urk4[i] = phi(t_valeurs[0])

for i in range(1, len(x_valeurs)):
    udiff[i] = udiff[i] - c * (tau / h) * (udiff[i] - udiff[i-1]) + tau * source(x_valeurs[i] , 0.75)

for i in range(len(x_valeurs)):
    k1_t_rk2 = tau * (source(x_valeurs[i] , 0.75) - c * (urk2[i] - urk2[i-1]) / (2 * h))
    k1_x_rk2 = c * tau * (urk2[i])

    k2_t_rk2 = tau * (source(x_valeurs[i] , 0.75 + (tau/2)) - c * (urk2[i] + k1_x_rk2 / 2 - urk2[i-1] - k1_t_rk2 / 2) / (2 *
h))
    k2_x_rk2 = c * tau * (urk2[i] + k1_x_rk2 / 2)

    u_t_moitie_rk2 = (urk2[i] + urk2[i-1]) / 2 + k2_t_rk2
    u_x_nouv_rk2 = urk2[i] + (k1_x_rk2 + k2_x_rk2) / 2

    k1_rk2 = tau * (source(x_valeurs[i] , 0.75) - c * (u_x_nouv_rk2 - urk2[i-1]) / (2 * h))
    k2_rk2 = tau * (source(x_valeurs[i] , 0.75 + (tau/2)) - c * (u_x_nouv_rk2 - urk2[i-1] - k1_rk2 / 2) / (2 * h))
    urk2[i] = urk2[i] + (k1_rk2 + k2_rk2) / 2

for i in range(len(x_valeurs)):
    k1_t_rk4 = tau * (source(x_valeurs[i] , 0.75) - c * (urk4[i] - urk4[i-1]) / (2 * h))
    k1_x_rk4 = c * tau * (urk4[i])

    k2_t_rk4 = tau * (source(x_valeurs[i] , 0.75 + (tau/2)) - c * (urk4[i] + k1_x_rk4 / 2 - urk4[i-1] - k1_t_rk4 / 2) / (2 *
h))
    k2_x_rk4 = c * tau * (urk4[i] + k1_x_rk4 / 2)

    k3_t_rk4 = tau * (source(x_valeurs[i] , 0.75 + (tau/2)) - c * (urk4[i] + k2_x_rk4 / 2 - urk4[i-1] - k2_t_rk4 / 2) / (2 *
h))
    k3_x_rk4 = c * tau * (urk4[i] + k2_x_rk4 / 2)

    k4_t_rk4 = tau * (source(x_valeurs[i] , 0.75 + tau) - c * (urk4[i] + k3_x_rk4 - urk4[i-1] - k3_t_rk4) / (2 * h))
    k4_x_rk4 = c * tau * (urk4[i] + k3_x_rk4)

    urk4[i] = urk4[i] + (k1_t_rk4 + 2*k2_t_rk4 + 2*k3_t_rk4 + k4_t_rk4) / 6

plt.figure(figsize=(10, 6))
plt.plot(x_valeurs, udiff, label='Théorique')
plt.plot(x_valeurs, urk2, label='RK2', linestyle='dotted')
plt.plot(x_valeurs, urk4, label='RK4', linestyle='dashed')
plt.legend()
plt.grid(True)
plt.show()
```

1.2 Comparaison des méthodes de résolution de l'équation de transport avec terme source et calcul de l'écart à t=0,75

```
import numpy as np
import matplotlib.pyplot as plt

c = 1.0
h = 0.1
tau = 0.01
T = 2.0
N = 2.0
x_valeurs = np.arange(0, N, h)
t_valeurs = np.arange(0, T, tau)

def u0(x):
    return np.cos(2*x)

def phi(t):
    return np.exp(t**2)

def source(x, t):
    return np.sin(2*x*t)

udiff = np.zeros(len(x_valeurs))
urk2 = np.zeros(len(x_valeurs))
urk4 = np.zeros(len(x_valeurs))

for i in range(len(x_valeurs)):
    if x_valeurs[i] < c * t_valeurs[0]:
        udiff[i] = u0(x_valeurs[i])
        urk2[i] = u0(x_valeurs[i])
        urk4[i] = u0(x_valeurs[i])
    else:
        udiff[i] = phi(t_valeurs[0])
        urk2[i] = phi(t_valeurs[0])
        urk4[i] = phi(t_valeurs[0])

for i in range(1, len(x_valeurs)):
    udiff[i] = udiff[i] - c * (tau / h) * (udiff[i] - udiff[i-1]) + tau * source(x_valeurs[i] , 0.75)

for i in range(len(x_valeurs)):
    k1_t_rk2 = tau * (source(x_valeurs[i] , 0.75) - c * (urk2[i] - urk2[i-1]) / (2 * h))
    k1_x_rk2 = c * tau * (urk2[i])

    k2_t_rk2 = tau * (source(x_valeurs[i] , 0.75 + (tau/2)) - c * (urk2[i] + k1_x_rk2 / 2 - urk2[i-1] - k1_t_rk2 / 2) / (2 *
h))
    k2_x_rk2 = c * tau * (urk2[i] + k1_x_rk2 / 2)
```

```
    u_t_moitie_rk2 = (urk2[i] + urk2[i-1]) / 2 + k2_t_rk2
    u_x_nouv_rk2 = urk2[i] + (k1_x_rk2 + k2_x_rk2) / 2

    k1_rk2 = tau * (source(x_valeurs[i] , 0.75) - c * (u_x_nouv_rk2 - urk2[i-1]) / (2 * h))
    k2_rk2 = tau * (source(x_valeurs[i] , 0.75 + (tau/2)) - c * (u_x_nouv_rk2 - urk2[i-1] - k1_rk2 / 2) / (2 * h))
    urk2[i] = urk2[i] + (k1_rk2 + k2_rk2) / 2

for i in range(len(x_valeurs)):
    k1_t_rk4 = tau * (source(x_valeurs[i] , 0.75) - c * (urk4[i] - urk4[i-1]) / (2 * h))
    k1_x_rk4 = c * tau * (urk4[i])

    k2_t_rk4 = tau * (source(x_valeurs[i] , 0.75 + (tau/2)) - c * (urk4[i] + k1_x_rk4 / 2 - urk4[i-1] - k1_t_rk4 / 2) / (2 *
h))
    k2_x_rk4 = c * tau * (urk4[i] + k1_x_rk4 / 2)

    k3_t_rk4 = tau * (source(x_valeurs[i] , 0.75 + (tau/2)) - c * (urk4[i] + k2_x_rk4 / 2 - urk4[i-1] - k2_t_rk4 / 2) / (2 *
h))
    k3_x_rk4 = c * tau * (urk4[i] + k2_x_rk4 / 2)

    k4_t_rk4 = tau * (source(x_valeurs[i] , 0.75 + tau) - c * (urk4[i] + k3_x_rk4 - urk4[i-1] - k3_t_rk4) / (2 * h))
    k4_x_rk4 = c * tau * (urk4[i] + k3_x_rk4)

    urk4[i] = urk4[i] + (k1_t_rk4 + 2*k2_t_rk4 + 2*k3_t_rk4 + k4_t_rk4) / 6

diff_rk2 = np.abs(urk2 - udiff)
diff_rk4 = np.abs(urk4 - udiff)

plt.figure(figsize=(10, 6))
plt.plot(x_valeurs, diff_rk2, label='Écart RK2', linestyle='dashed')
plt.plot(x_valeurs, diff_rk4, label='Écart RK4', linestyle='dashed')
plt.legend()
plt.grid(True)
plt.show()
```

2 Deuxième exemple

2.1 Solution de l'équation de transport avec terme source en fonction de x à t=0,75

```
import numpy as np
import matplotlib.pyplot as plt

c = 1.0
h = 0.1
tau = 0.01
T = 2.0
```

```
N = 2.0
x_valeurs = np.arange(0, N, h)
t_valeurs = np.arange(0, T, tau)

def u0(x):
    return np.sin(x)

def phi(t):
    return 2*t-1

def source(x, t):
    return np.exp(t**2)+x

udiff = np.zeros(len(x_valeurs))
urk2 = np.zeros(len(x_valeurs))
urk4 = np.zeros(len(x_valeurs))

for i in range(len(x_valeurs)):
    if x_valeurs[i] < c * t_valeurs[0]:
        udiff[i] = u0(x_valeurs[i])
        urk2[i] = u0(x_valeurs[i])
        urk4[i] = u0(x_valeurs[i])
    else:
        udiff[i] = phi(t_valeurs[0])
        urk2[i] = phi(t_valeurs[0])
        urk4[i] = phi(t_valeurs[0])

for i in range(1, len(x_valeurs)):
    udiff[i] = udiff[i] - c * (tau / h) * (udiff[i] - udiff[i-1]) + tau * source(x_valeurs[i] , 0.75)

for i in range(len(x_valeurs)):
    k1_t_rk2 = tau * (source(x_valeurs[i] , 0.75) - c * (urk2[i] - urk2[i-1]) / (2 * h))
    k1_x_rk2 = c * tau * (urk2[i])

    k2_t_rk2 = tau * (source(x_valeurs[i] , 0.75 + (tau/2)) - c * (urk2[i] + k1_x_rk2 / 2 - urk2[i-1] - k1_t_rk2 / 2) / (2 *
h))
    k2_x_rk2 = c * tau * (urk2[i] + k1_x_rk2 / 2)

    u_t_moitie_rk2 = (urk2[i] + urk2[i-1]) / 2 + k2_t_rk2
    u_x_nouv_rk2 = urk2[i] + (k1_x_rk2 + k2_x_rk2) / 2

    k1_rk2 = tau * (source(x_valeurs[i] , 0.75) - c * (u_x_nouv_rk2 - urk2[i-1]) / (2 * h))
    k2_rk2 = tau * (source(x_valeurs[i] , 0.75 + (tau/2)) - c * (u_x_nouv_rk2 - urk2[i-1] - k1_rk2 / 2) / (2 * h))
    urk2[i] = urk2[i] + (k1_rk2 + k2_rk2) / 2

for i in range(len(x_valeurs)):
    k1_t_rk4 = tau * (source(x_valeurs[i] , 0.75) - c * (urk4[i] - urk4[i-1]) / (2 * h))
    k1_x_rk4 = c * tau * (urk4[i])
```

```
    k2_t_rk4 = tau * (source(x_valeurs[i] , 0.75 + (tau/2)) - c * (urk4[i] + k1_x_rk4 / 2 - urk4[i-1] - k1_t_rk4 / 2) / (2 *
h))
    k2_x_rk4 = c * tau * (urk4[i] + k1_x_rk4 / 2)

    k3_t_rk4 = tau * (source(x_valeurs[i] , 0.75 + (tau/2)) - c * (urk4[i] + k2_x_rk4 / 2 - urk4[i-1] - k2_t_rk4 / 2) / (2 *
h))
    k3_x_rk4 = c * tau * (urk4[i] + k2_x_rk4 / 2)

    k4_t_rk4 = tau * (source(x_valeurs[i] , 0.75 + tau) - c * (urk4[i] + k3_x_rk4 - urk4[i-1] - k3_t_rk4) / (2 * h))
    k4_x_rk4 = c * tau * (urk4[i] + k3_x_rk4)

    urk4[i] = urk4[i] + (k1_t_rk4 + 2*k2_t_rk4 + 2*k3_t_rk4 + k4_t_rk4) / 6

plt.figure(figsize=(10, 6))
plt.plot(x_valeurs, udiff, label='Théorique')
plt.plot(x_valeurs, urk2, label='RK2', linestyle='dashed')
plt.plot(x_valeurs, urk4, label='RK4', linestyle='dotted')
plt.legend()
plt.grid(True)
plt.show()
```

2.2 Comparaison des méthodes de résolution de l'équation de transport avec terme source et calcul de l'écart à t=0,75

```
import numpy as np
import matplotlib.pyplot as plt

c = 1.0
h = 0.1
tau = 0.01
T = 2.0
N = 2.0
x_valeurs = np.arange(0, N, h)
t_valeurs = np.arange(0, T, tau)

def u0(x):
    return np.sin(x)

def phi(t):
    return 2*t-1

def source(x, t):
    return np.exp(t**2)+x

udiff = np.zeros(len(x_valeurs))
urk2 = np.zeros(len(x_valeurs))
urk4 = np.zeros(len(x_valeurs))
```

```
for i in range(len(x_valeurs)):
    if x_valeurs[i] < c * t_valeurs[0]:
        udiff[i] = u0(x_valeurs[i])
        urk2[i] = u0(x_valeurs[i])
        urk4[i] = u0(x_valeurs[i])
    else:
        udiff[i] = phi(t_valeurs[0])
        urk2[i] = phi(t_valeurs[0])
        urk4[i] = phi(t_valeurs[0])

for i in range(1, len(x_valeurs)):
    udiff[i] = udiff[i] - c * (tau / h) * (udiff[i] - udiff[i-1]) + tau * source(x_valeurs[i] , 0.75)

for i in range(len(x_valeurs)):
    k1_t_rk2 = tau * (source(x_valeurs[i] , 0.75) - c * (urk2[i] - urk2[i-1]) / (2 * h))
    k1_x_rk2 = c * tau * (urk2[i])

    k2_t_rk2 = tau * (source(x_valeurs[i] , 0.75 + (tau/2)) - c * (urk2[i] + k1_x_rk2 / 2 - urk2[i-1] - k1_t_rk2 / 2) / (2 *
h))
    k2_x_rk2 = c * tau * (urk2[i] + k1_x_rk2 / 2)

    u_t_moitie_rk2 = (urk2[i] + urk2[i-1]) / 2 + k2_t_rk2
    u_x_nouv_rk2 = urk2[i] + (k1_x_rk2 + k2_x_rk2) / 2

    k1_rk2 = tau * (source(x_valeurs[i] , 0.75) - c * (u_x_nouv_rk2 - urk2[i-1]) / (2 * h))
    k2_rk2 = tau * (source(x_valeurs[i] , 0.75 + (tau/2)) - c * (u_x_nouv_rk2 - urk2[i-1] - k1_rk2 / 2) / (2 * h))
    urk2[i] = urk2[i] + (k1_rk2 + k2_rk2) / 2

for i in range(len(x_valeurs)):
    k1_t_rk4 = tau * (source(x_valeurs[i] , 0.75) - c * (urk4[i] - urk4[i-1]) / (2 * h))
    k1_x_rk4 = c * tau * (urk4[i])

    k2_t_rk4 = tau * (source(x_valeurs[i] , 0.75 + (tau/2)) - c * (urk4[i] + k1_x_rk4 / 2 - urk4[i-1] - k1_t_rk4 / 2) / (2 *
h))
    k2_x_rk4 = c * tau * (urk4[i] + k1_x_rk4 / 2)

    k3_t_rk4 = tau * (source(x_valeurs[i] , 0.75 + (tau/2)) - c * (urk4[i] + k2_x_rk4 / 2 - urk4[i-1] - k2_t_rk4 / 2) / (2 *
h))
    k3_x_rk4 = c * tau * (urk4[i] + k2_x_rk4 / 2)

    k4_t_rk4 = tau * (source(x_valeurs[i] , 0.75 + tau) - c * (urk4[i] + k3_x_rk4 - urk4[i-1] - k3_t_rk4) / (2 * h))
    k4_x_rk4 = c * tau * (urk4[i] + k3_x_rk4)

    urk4[i] = urk4[i] + (k1_t_rk4 + 2*k2_t_rk4 + 2*k3_t_rk4 + k4_t_rk4) / 6

diff_rk2 = np.abs(urk2 - udiff)
diff_rk4 = np.abs(urk4 - udiff)

plt.figure(figsize=(10, 6))
```

```
plt.plot(x_valeurs, diff_rk2, label='Écart RK2', linestyle='dashed')
plt.plot(x_valeurs, diff_rk4, label='Écart RK4', linestyle='dashed')
plt.legend()
plt.grid(True)
plt.show()
```

Conclusion

Les méthodes Runge-Kutta sont un ensemble de méthodes numériques utilisées pour résoudre les équations différentielles ordinaires (EDO). Elles ont été initialement développées pour les EDO, mais dans cette étude, elles ont été adaptées et améliorées pour répondre aux défis plus complexes posés par les équations aux dérivées partielles (EDP).

Les méthodes Runge-Kutta d'ordre 2 (RK2) et d'ordre 4 (RK4) ont été comparées pour résoudre l'équation de transport. Les résultats ont montré que RK4 est supérieure à RK2 à la fois en terme de précision et de stabilité. En particulier, RK4 converge plus rapidement et est moins susceptible de souffrir d'instabilités numériques. RK4 devient donc un outil de choix pour la résolution d'EDP qui nécessitent une grande précision et une bonne stabilité.

Bibliographie

[1] *Analyse numérique des équations aux dérivées partielles*
Laurent DI MENZA

[2] *Méthodes de Monte-Carlo pour les équations de transport et de diffusion*
Bernard LAPEYRE, Étienne PARDOUX et Rémi SENTIS.

[3] *Méthodes numériques pour les EDP en mécanique*
Cours de Pierre BUFFAT (Université C.Bernard de Lyon)
http://www.ufrmeca.univ-lyon1.fr/~buffat/COURS/COURSDF_HTML/node25.html

[4] *Justification physique de l'équation de transport*
Projet de Physique P6 STPI /P6/2015-7 (Institut National des Sciences Appliquées de Rouen)
Marie-Eve CLAVEL, Eliass EL ALAMI, Joseph LEFEVRE, Xavier LEMAHIEU, Nathan ROUXELIN, Mathilde TAVERNIER

[5] *Résolution analytique de l'équation de transport*
Projet de Physique P6 STPI /P6/2015-7 (Institut National des Sciences Appliquées de Rouen)
Marie-Eve CLAVEL, Eliass EL ALAMI, Joseph LEFEVRE, Xavier LEMAHIEU, Nathan ROUXELIN, Mathilde TAVERNIER

[6] *Méthode des différences finies*
Wikipédia
http://fr.wikipedia.org/wiki/M%C3%A9thode_des_caract%C3%A9ristiques

[7] *Résolution numérique des équations différentielles*
https://www.iro.umontreal.ca/~mignotte/IFT2425/Chapitre6_cor_.pdf

[8] *Étude analytique et numérique du problème de Saint-Venant avec une dynamique biologique*
Soumaya OUESLATI (École Polytechnique de Tunisie)

Résumé : Ce rapport de stage présente l'utilisation des méthodes Runge-Kutta d'ordre 2 et 4 pour la résolution de l'équation de transport. Ces méthodes ont été utilisées auparavant pour les équations différentielles ordinaires, mais elles peuvent aussi être utilisées pour les équations aux dérivées partielles. Elles permettent d'obtenir des solutions adaptatives et fiables, implémentées en Python.
Mots clés : Runge-Kutta, EDP, Convection, Diffusion, Modélisation, Simulation, Python

---

Abstract : This internship report presents the use of Runge-Kutta methods of order 2 and 4 to solve the transport equation. These methods have previously been used for ordinary differential equations, but they can also be used for partial differential equations. They provide adaptive and reliable solutions, implemented in Python.
Key words: Runge-Kutta, PDE, Convection, Diffusion, Modelling, Simulation, Python

---

ملخص : يعرض تقرير التربص إستخدام طرق رونج- كوتا درجة 2 و 4 لحل معادلة النقل. تم إستخدام هذه الطرق سابقا في المعادلات التفاضلية العادية، ولكن يمكن أيضا إستخدامها للمعادلات التفاضلية الجزئية حيث أنها توفر حلولا دقيقة و موثوقة، تم تنفيذها بلغة البرمجة بايثون.