tpe_analyse_numerique

July 9, 2023

0.1 Méthode de Runge kunta d'ordre 2

Le but de ce travail est d'étudier le phénomène de convergence sur les méthodes numériques de résolution d'équations différentielles, pour y parvenir nous allons tout d'abord résoudre analytiquement le problème de Cauchy puis résoudre par deux méthodes numériques

Premièrement, Nous allons résoudre le problème de Cauchy en utilisant la méthode Runge kutta d'ordre 2 et ensuite tracer un graphique pour visualiser et analyser les résultats obtenus.

[3]:
```
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
%matplotlib inline

@np.vectorize
def solution(t):
    """
    cette fonction est la solution analytque du probleme de cauchy
    """
    return -(1.0/(np.sin(t)-0.5))

def f(t,y):
    """
    la fonction du probleme de cauchy
    """
    return np.cos(t) * y * y

## initialisation des conditions initiales
t, y, h, n=np.float64(0.0), np.float64(2.0), np.float64(0.1),5

## liste des resultats qui nous servirons pour afficher les resultats
y_liste, t_liste =[y], [t]

for i in range(n):
    K1 = h*f(t,y)
    K2 = h*f(t + h/2 , y + K1/2 )
    y = y + K2
    t = np.round(t + h,4)
    print("k1=", K1,"K2=",K2,"t=",t,"y=",y,"\n")
```

```
    y_liste.append(y)
    t_liste.append(t)

# print(y_liste,t_liste)

plt.figure(figsize=(15,8))
plt.plot(t_liste, y_liste)
plt.plot(t_liste,solution(t_liste))
plt.scatter(t_liste, y_liste)
plt.scatter(t_liste, solution(t_liste))
plt.ylabel('valeurs de $f(t,y)$')
plt.xlabel('value de t')
plt.legend(["runge kutta d'ordre 2", "solution analytique"])
plt.show()
print("les solutions analytiques: ",solution(t_liste))
print("Erreur absolue :", solution(t_liste)-y_liste)
```

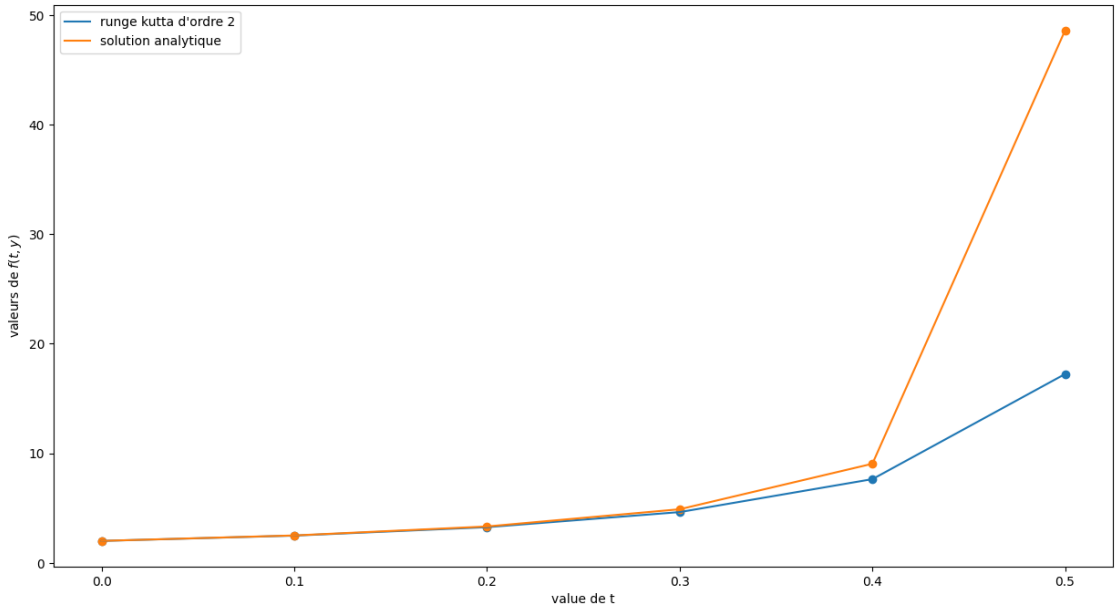
```
k1= 0.4 K2= 0.4833951260311638 t= 0.1 y= 2.483395126031164

k1= 0.6136440783551901 K2= 0.7697891169253865 t= 0.2 y= 3.2531842429565505

k1= 1.0372248171371685 K2= 1.3784184102909192 t= 0.3 y= 4.6316026532474694

k1= 2.049363297466948 K2= 3.005386983434381 t= 0.4 y= 7.63698963668185

k1= 5.371960285511385 K2= 9.595497944370821 t= 0.5 y= 17.232487581052673
```



```
les solutions analytiques:  [ 2.          2.49895929  3.31861341  4.89045878
9.04309106 48.60394548]
Erreur absolue : [0.00000000e+00 1.55641614e-02 6.54291681e-02 2.58856123e-01
 1.40610142e+00 3.13714579e+01]
```

0.2 Méthode de runge kunta d'ordre 4

Dans cette partie de nous travailler nous allons résoudre le problème de Cauchy de l'exercice en utilisant cette fois-ci la méthode de Runge kutta d'ordre 4.

[4]:
```python
import numpy as np
import matplotlib.pyplot as plt

def f(t,y):
    return np.cos(t) * y * y

## initialisation des conditions initiales
t, y, h, n=np.float64(0.0), np.float64(2.0), np.float64(0.1),5

## liste des resultats qui nous servirons pour afficher les resultats
y_liste, t_liste =[y], [t]

for i in range(n):
    K1 = h*f(t,y)
    K2 = h*f(t + h/2, y + K1/2)
    K3 = h*f(t + h/2, y + K2/2)
    K4 = h*f(t + h , y + K3)

    y = y +  (K1 + 2*K2 + 2*K3 + K4)/6
    t = np.round(t + h,2)
    print("k1=", K1,"K2=",K2,"K3=",K3,"K4=",K4,"t=",t,"y=",y,"\n")
    y_liste.append(y)
    t_liste.append(t)

# print(y_liste,t_liste)

plt.figure(figsize=(15,8))
plt.plot(t_liste, y_liste)
plt.plot(t_liste,solution(t_liste))
plt.scatter(t_liste, y_liste)
plt.scatter(t_liste, solution(t_liste))
plt.ylabel('valeurs de $f(t,y)$')
plt.xlabel('valeur de t')
plt.legend(["runge kutta d'ordre 4", "solution analytique"])
plt.show()
print("les solutions analytiques: ", solution(t_liste))
print("Erreur absolue :", solution(t_liste)-y_liste)
```

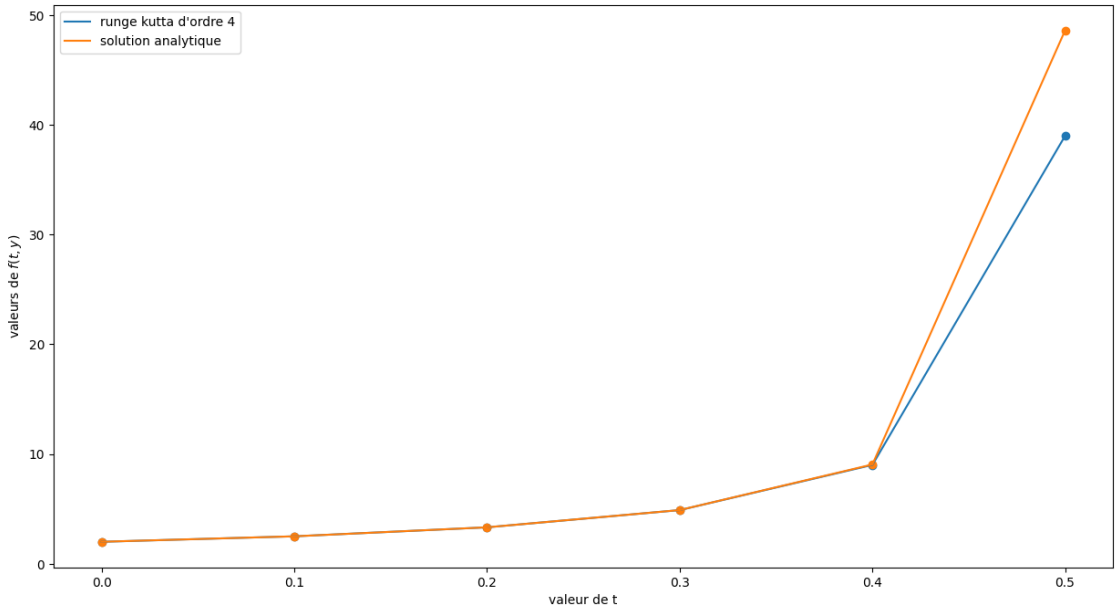
```
k1= 0.4 K2= 0.4833951260311638 K3= 0.501892776261382 K4= 0.6228196199011622 t=
0.1 y= 2.4988992374143755

k1= 0.6213300921824194 K2= 0.7805014266070494 K3= 0.8253457520541692 K4=
1.083032838041526 t= 0.2 y= 3.318242118672106

k1= 1.0791249213651326 K2= 1.4419990433574046 K3= 1.5808264484022516 K4=
2.2928909580486314 t= 0.3 y= 4.887853262494286

k1= 2.28240486860213 K2= 3.414574025710192 K3= 4.085883833270609 K4=
7.417116054672819 t= 0.4 y= 9.004592702700378

k1= 7.468210280975647 K2= 14.611953374222933 K3= 23.955015116643896 K4=
95.33493084562718 t= 0.5 y= 38.994105720756465
```




```
les solutions analytiques:  [ 2.          2.49895929  3.31861341  4.89045878
9.04309106 48.60394548]
Erreur absolue : [0.00000000e+00 6.00500418e-05 3.71292430e-04 2.60551425e-03
 3.84983567e-02 9.60983976e+00]
```

0.3 Observation et discussion

À partir des deux graphiques tracés plus haut nous constatons que la méthode de Runge Kutta d'ordre 4 se rapproche beaucoup plus du résultat analytique que la méthode de Runge Kutta d'ordre 2. autrement dit, la méthode de Runge kutta d'ordre 4 converge plus rapidement que la méthode de Runge kutta d'ordre 2

Après la valuer de $t = 0.4$ la méthode de Runge Kutta d'ordre 2 diverge complement alors que la méthode de Runge Kutta d'ordre 4 après cette valeur se rapproche de la solution analytique.

En conclusion la méthode de Runge Kutta d'ordre 4 a une meilleure précision sur le résultat que la méthode de Runge Kutta d'ordre 2.