



Faculté des sciences de Montpellier
Rapport séance 1

Étudiant: Giscard Leonel Zouakeu

Partie 1

Réolvons l'équation différentielle suivante:

$$\begin{aligned}u'(x) &= -\lambda u(x) \\ u(0) &= 1\end{aligned}\tag{1}$$

avec λ une constate et $x \in [0, 1]$

une solution de cette équation différentielle est : $u(x) = e^{-\lambda x}$

Résolution de l'équation avec Python pour $\lambda = 1$

On peut écrire(1) de la façon suivante: $y'(t) = f(t, y(t))$ avec $y(t_0) = y_0$.
Ce problème est appelé problème de Cauchy.

```
1 import numpy as np
2 #import math
3 #from scipy.integrate import odeint
4 from scipy.integrate import solve_ivp
5 import matplotlib.pyplot as plt
6 #import scipy
7 #from scipy.integrate import quad
8
9 #
10 # -----
11 # on a par exemple l'équation différentielle du 1ere ordre:
12 # -----
13 #          # u'(t)+ a.u(t)= b avec a,b des constantes
14 #          # u(t_0)= c avec c une constante
15 # -----
16 # pour resoudre cette equation différentielle nous la mettons
17 # sous la forme:
18 # u'(t)= -a u(t) + b
19 #
20 # -----
21
22 # definition de l'équation différentielle du 1ere ordre
23
24 def equation (t,u):
25     lamb= -1
26     return lamb*u
27
28 t_initial = 0 # temps initial
29 t_final =1   # temps final
30 u_0= 1 # condition initiale
31 t= np.linspace(t_initial,t_final, 100)
```

```

32 #solution python scipy
33
34 sol_solve_ivp=solve_ivp(equation,[t_initial,t_final],[u_0],
    dense_output=True)
35
36 z=sol_solve_ivp.sol(t)
37 plt.plot(t,z.T)
38
39
40 # solution de l'equation
41 def U_exact (s,nn):
42     u=np.zeros(nn)
43     s= np.linspace(t_initial,t_final, nn)
44     for i in range(nn):
45         u[i]= np.exp(-s[i])
46     return u
47 resultat= U_exact(t,100)
48
49
50 #solution_odeint= odeint(equation,u_0,t)
51 #plt.plot(t,solution_odeint,label="scipy")
52 plt.plot(t,resultat)
53 plt.xlabel("Temps")
54 plt.ylabel("")
55 plt.show
56
57 # definition de la methode de euleur explicite
58
59 def Euleur_explicite(fonction,t_i,t_f,n,u_i):
60     y= np.zeros(n+1) # vecteur de y initialise a 0.
61     y[0]= u_i # condition initiale
62     h= (t_f - t_i)/n # pas du temps
63     t1= np.linspace(t_initial,t_final, n+1)
64     for i in np.arange(n):
65         y[i+1]= y[i]+ h*fonction(t1[i],y[i])
66     return t1, y
67
68 t2,solution_Euleur= Euleur_explicite(equation, t_initial,
    t_final,99,u_0)
69
70 plt.plot(t2,solution_Euleur)
71 plt.legend(["Solution_scipy","Solution_exacte",
    Euleur_explicite"])
72 plt.show()

```

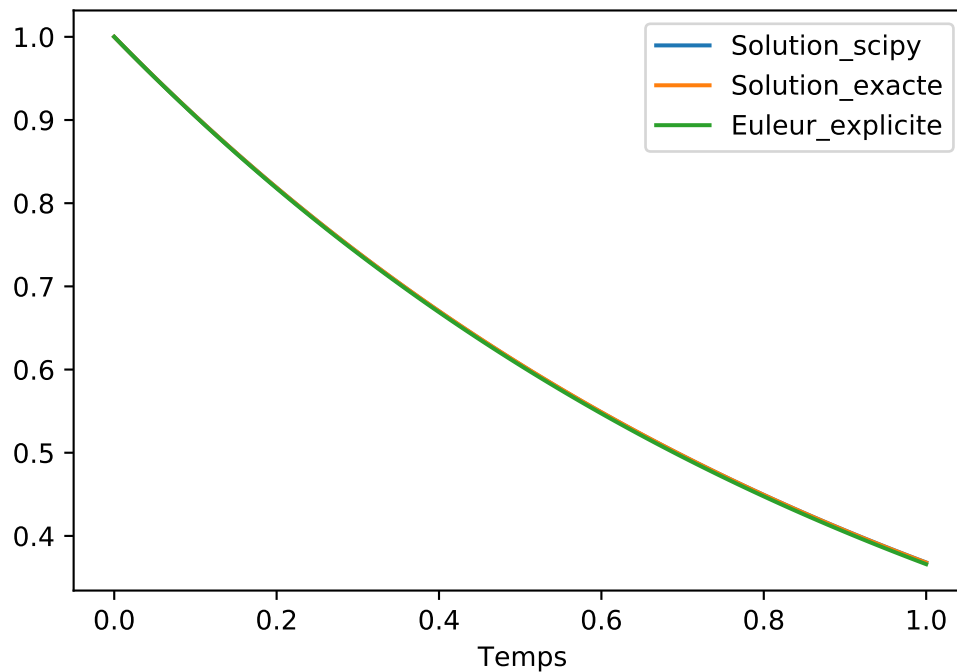


Figure 1: comparaison des Solutions

Traçons la courbe d'erreur L_2

On définit l'erreur L_2 :

$$L_2 = \sqrt{\sum_i^n (U^{app} - U^{exac})^2}$$

```

1 # definition de l'erreur L2
2 f_erreur=np.zeros(Nb)
3 dt=np.linspace(0,1,Nb)
4 Erreur_L2=np.zeros(Nb)
5 for i in range (Nb):
6     f_erreur[i]=np.sum((solution_Euleur[i]-resultat[i])**2)
7     Erreur_L2[i]=np.sqrt(f_erreur[i])
8
9 plt.loglog(dt,Erreur_L2)
10 plt.xlabel("pas Temps")
11 plt.ylabel("function_erreur")
12 plt.legend(["Erreur"])
13 plt.savefig('Erreur.pdf', dpi=300, bbox_inches='tight')
14 plt.show()

```

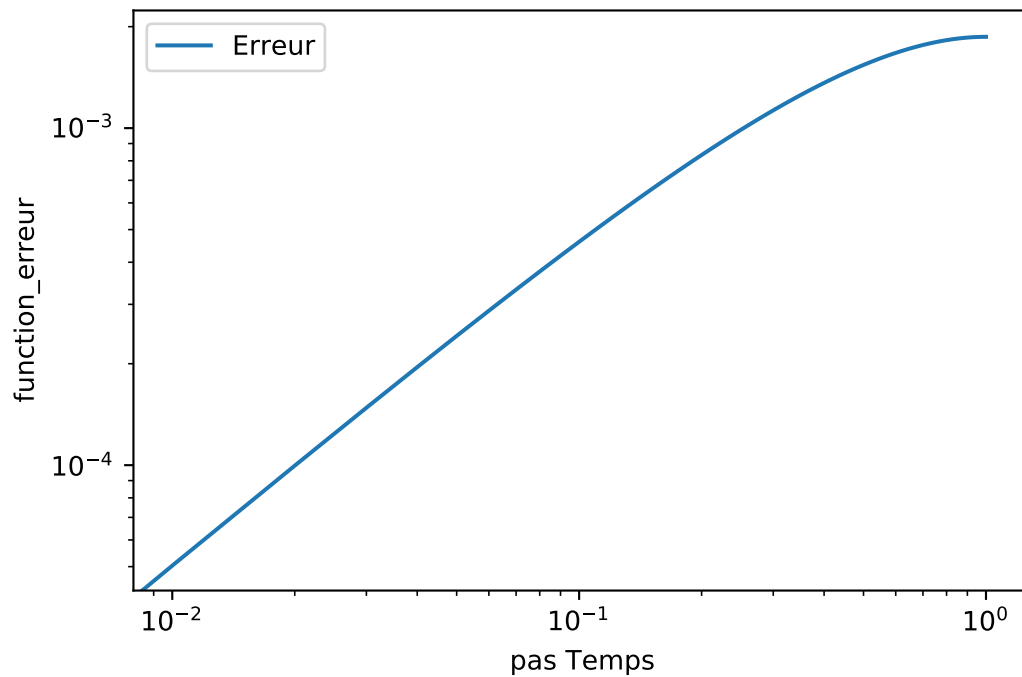


Figure 2: Courbe de l'erreur

Partie 2

On pose: $j'(x) = \sin(x)$, $j''(x) = \cos(x)$ $x \in [0, \pi]$

Méthode de différence finie

soit $L_h u$ L'opérateur approché:

$$L_h u = \frac{u(x+h) - 2u(x) + u(x-h)}{h^2} \text{ pour la dérivée seconde}$$

$$L_h u = \frac{u(x+h) - u(x-h)}{2h} \text{ pour la dérivée première}$$

```

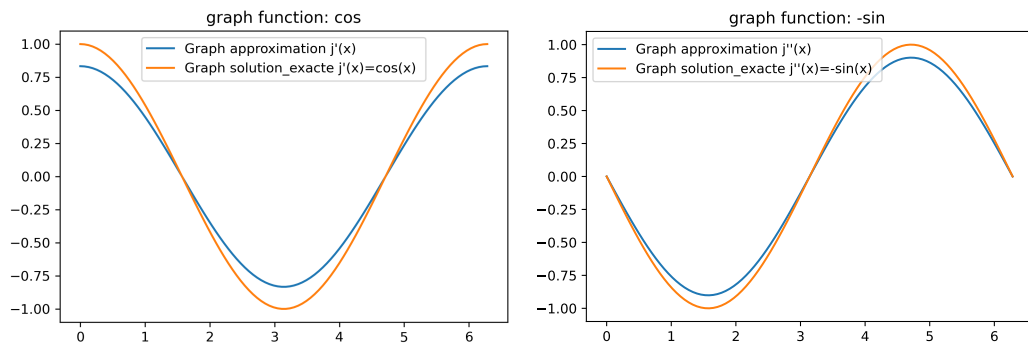
1 # -----
2 # j(x)=sin(x) calculons j'(x) et j''(x) par la methode:
3
4
5 # Methode de la difference finie
6
7 N=100
8 h= 1/(N+1)*[1.0E-01,5.0E-02,1.0E-02,5.0E-03,1.0E-03 ]
9 x=np.linspace(0,2*np.pi,N)
10 J=lambda x : np.sin(x)
11
12 Y=np.cos(x) # fonction exacte
13 YY= -1*np.sin(x) # fonction exacte

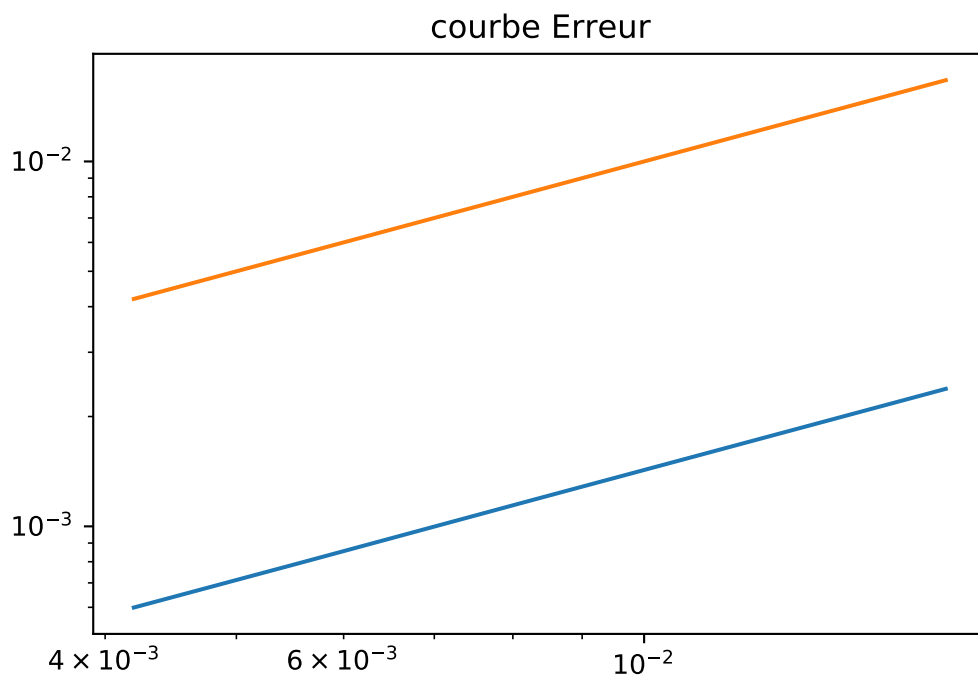
```

```

14 J_diff_1 =np.zeros(N)
15 J_diff_2 =np.zeros(N)
16 fun_sin =np.zeros(N)
17
18 for i in range(N):
19     J_diff_2[i]=( J(x[i]+1) -2*J(x[i]) +J(x[i]-1))/(h**2) #
20     J_diff_1[i]=( J(x[i]+1) - J(x[i]-1))/(h*2) # la derive
21     premiere
22 plt.plot(x,J_diff_1)
23 plt.plot(x,Y)
24 plt.legend(["Graph approximation j'(x)","Graph solution_exacte
25 j'(x)=cos(x) "])
26 plt.title("graph function: cos")
27 plt.savefig('approx_j_diff', dpi=300, bbox_inches='tight')
28 plt.show()
29
30 plt.plot(x,J_diff_2)
31 plt.plot(x,YY)
32 plt.legend(["Graph approximation j''(x)","Graph solution_exacte
33 j''(x)=-sin(x) "])
34 plt.title("graph function: -sin ")
35 plt.savefig('approx_j_diffdiff.pdf', dpi=300, bbox_inches='
36 tight')
37 plt.show()

```





```

1 def func(x):
2
3     T = np.sin(x)
4     Txe = np.cos(x)
5     Txxe = -np.sin(x)
6
7     return T,Txe,Txxe
8
9 #####
10 def derivatives(NX):
11
12     NX=int(NX)
13     L=2*math.pi
14     dx = L/(NX-1)
15
16     x = np.linspace(0.0,L,NX)
17
18     T,Txe,Txxe=func(x)
19
20     Tx = np.zeros(NX)
21     Txx = np.zeros(NX)
22
23     for j in range (1, NX-1):
24         Tx[j] = (T[j+1]-T[j-1])/(dx*2)
25         Txx[j] = (T[j-1]-2*T[j]+T[j+1])/(dx**2)
26
27     Tx[0] = (T[1]-T[0])/dx

```

```

28 Tx[NX-1] = (T[NX-1]-T[NX-2])/dx
29 Txx[0] = 2*Txx[1]-Txx[2]
30 Txx[NX-1] = 2*Txx[NX-2]-Txx[NX-3]
31
32 errTx=0
33 errTxx=0
34 for j in range (0, NX):
35     errTx+=abs(Tx[j]-Txe[j])*dx
36     errTxx+=abs(Txx[j]-Txxe[j])*dx
37
38 #complex variable method
39 coef=100
40 dx/=coef
41 errcTx=0
42 errcTxx=0
43 for j in range (0, NX):
44     eps = complex(0,dx)
45     Tc0 = np.sin(x[j])
46     Tc = np.sin(x[j]+eps)
47     Tcx=Tc.imag/eps.imag
48     Tcxx=(Tc0-Tc).real/(eps.imag)**2*2
49     errcTx+=abs(Tcx-Txe[j])*dx*coef
50     errcTxx+=abs(Tcxx-Txxe[j])*dx*coef
51
52     return errTx,errTxx,errcTx,errcTxx
53 #####
54
55 Nsample=20
56 DNX=round(300/Nsample)
57
58 sample=np.zeros((Nsample))
59 terrTx=np.zeros((Nsample))
60 terrTxx=np.zeros((Nsample))
61 terrcTx=np.zeros((Nsample))
62 terrcTxx=np.zeros((Nsample))
63
64 NX=10
65 for j in range (0, Nsample):
66     sample[j]=NX
67     NX+=DNX
68     terrTx[j],terrTxx[j],terrcTx[j],terrcTxx[j]=derivatives(NX)
69 #     print(NX,terrTx[j],terrTxx[j],terrcTx[j],terrcTxx[j])
70
71 #
72 plt.figure(1)
73 plt.plot(sample,np.log10(terrTx))
74 plt.plot(sample,np.log10(terrTxx))
75 plt.figure(2)
76 plt.title(u'L1 ERROR ON TX & TXX by CVM')
77 plt.xlabel(u'$NX$', fontsize=26)
78 plt.ylabel(u'$|Tx - Tx_{cvm}|$', fontsize=26, rotation=90)
79 plt.plot(sample,np.log10(terrcTx))
80 plt.plot(sample,np.log10(terrcTxx))
81 plt.show()

```