U.A.B.T. 2016/2017
Département de Mathématiques Master E. D. P.

Corrigé de l'examen final de langages de programmation
Sujet A

Exercice 1

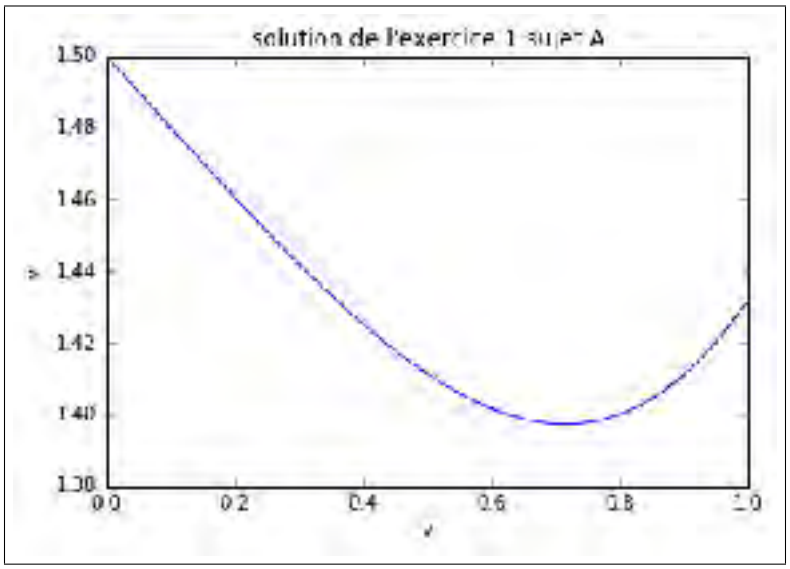
En utilisant **odeint** de scipy.integrate, résoudre le problème suivant:

$y'' + xy' \square x^2 y = 0 \qquad x \in [0, 1]$

avec les conditions initiales $y(0) = 1, y'(0) = 2$ et représenter y en fonction de x.

Corrigé

```
from scipy.integrate import odeint
import numpy as np
import pylab as plt
def f(y,x):
 return np.array([y[1],-x*y[1]+x**2*y[0]])
x=np.linspace(0,1,100)
w=odeint(f,[1.5,-.2],x)
plt.plot(x,w[:,0])
plt.title('solution de l exercice 1 sujet A')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```



Exercice 2

Utiliser les fonctions **simps** et **quad** de scipy.integrate pour calculer:

$\int_{1.0}^{3.5}(1 + x^2)dx$

Corrigé

```
from scipy.integrate import quad,simps
import numpy as np
x=np.linspace(1.0,3.5,100)
def y(x):
```

```
 return 1+x**2
i=quad(y,1.0,3.5)
print('L\'integrale avec quad',i[0])
z=y(x)
i1=simps(z,x)
print('L\'integrale avec simps',i1)
```

L'exécution du programme donne:

L'intégrale avec quad 16.458333333333332
L'intégrale avec simps 16.4583360172

Exercice 3

On considère l'équation de Poisson 1D suivante:

$$\frac{d^2\phi}{dx^2}(x) = \square\rho(x) \qquad x \in [0,1]$$

avec $\phi(0) = \phi(1) = 0$ et $\rho(x) = 2x^2 \square 1$.

1- Utiliser la discrétisation de $[0,1]$, $0 = x_0 < x_1 < \cdots < x_N = 1$ avec un pas constant h.

Puis l'approximation: $\frac{d^2\phi}{dx^2}(x_i) \approx \frac{\phi(x_{i+1}) \square 2\phi(x_i) + \phi(x_{i\square 1})}{h^2}$ pour résoudre le problème précédent.

Construire la matrice du système linéaire avec la fonction **diag** de numpy.

2- Comparer avec la solution exacte: $\phi(x) = \square\frac{x}{3} + \frac{x^2}{2} \square \frac{x^4}{6}$, pour cela:

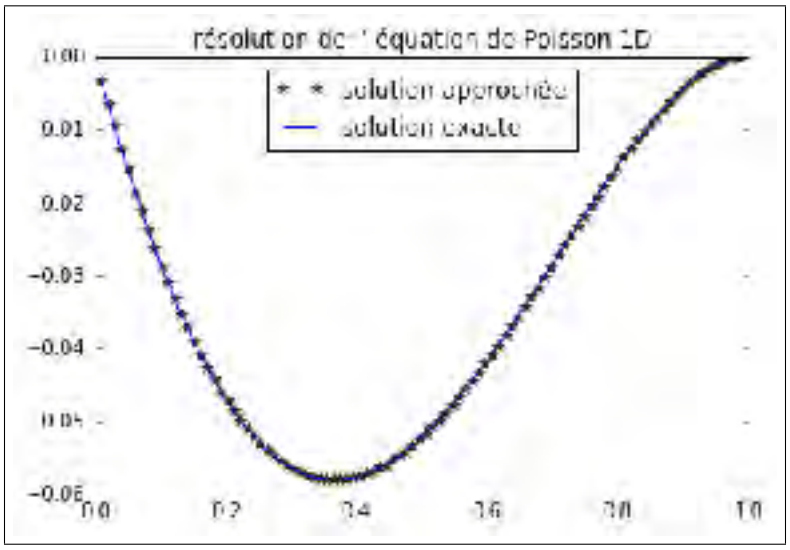
Tracer dans une même figure la solution exacte et la solution approchée en rouge *
Puis calculer $erreur = \|\phi(x_i) \square \phi_i\|_\infty$

Corrigé

```
import numpy as np
import numpy.linalg as ln
import matplotlib.pyplot as plt
f=lambda x: 2*x**2-1
Nx=100
x=np.linspace(0,1,Nx)
xx=x[1:(Nx-1)]
h=1/(Nx-1)
A=-2*np.diag(np.ones(Nx-2))+np.diag(np.ones(Nx-3),k=-1)+np.diag(np.ones(Nx-3),k=1)
b=-h**2*f(xx)
phi=ln.solve(A,b)
#solution exacte
sol=lambda x: -x/3+x**2/2-x**4/6
phiexa=sol(xx)
print('erreur=',ln.norm(phiexa-phi,np.inf))
plt.plot(xx,phi,'r*',xx,phiexa)
plt.legend(['solution approchée','solution exacte'],loc=9)
plt.title('résolution de l\' équation de Poisson 1D')
plt.show()
```

L'exécution du programme donne:

erreur= 4.25083311872e-06


résolut on de l équation de Poisson 1D
solution approchée
solution exacte
0.01
0.02
−0.03
−0.04
−0.06
0.0
0.2
0.4
0.6
0.8
1.0

U.A.B.T. 2016/2017
Département de Mathématiques Master E. D. P.

Corrigé de l'examen final de langages de programmation
Sujet B

Exercice 1

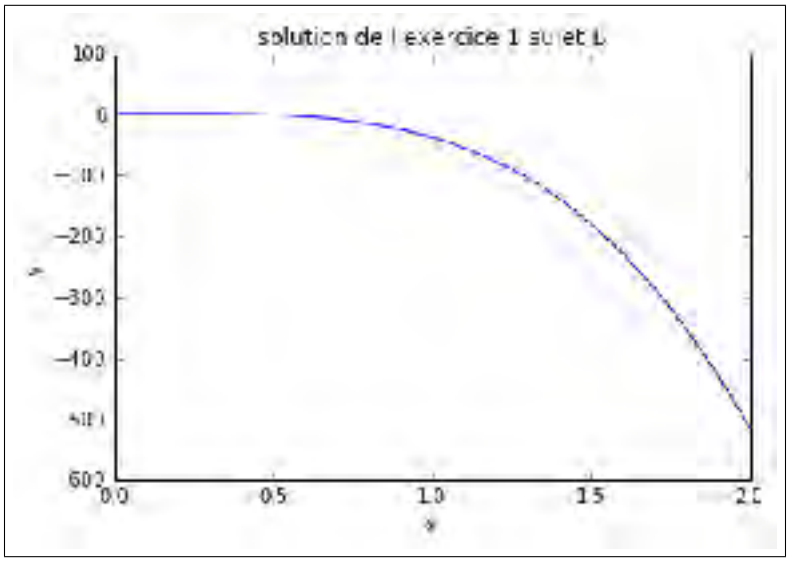
En utilisant **odeint** de scipy.integrate, résoudre le problème suivant:

$(3t \square 1)y'' \square (3t+2)y' + (6t \square 8)y = 0 \qquad t \in [0,2]$

avec les conditions initiales $y(0) = 2, y'(0) = 3$ et représenter y en fonction de t.

Corrigé

```
from scipy.integrate import odeint
import numpy as np
import matplotlib.pyplot as plt
def f(y,t):
 return np.array([y[1],((3*t+2)*y[1]-(6*t-8)*y[0])/(3*t-1)])
t=np.linspace(0,2,100)
w=odeint(f,[2,3],t)
plt.plot(t,w[:,0])
plt.title('solution de l\'exercice 1 sujet B')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```



Exercice 2

Utiliser la fonction **interp1d** de scipy.interpolate pour interpoler les données suivantes:
x = [0, 0.5, 1, 1.5, 2]
y = [0, 0.1250, 1.0000, 3.3750, 8.0000]
puis calculer l'intégrale entre 0.25 et 1.75 de la fonction trouvée avec **quad** de scipy.integrate.

Corrigé

```
from scipy.integrate import quad
from scipy.interpolate import interp1d
```

x = [0, 0.5, 1, 1.5, 2]
y = [0, 0.1250, 1.0000, 3.3750, 8.0000]
f=interp1d(x,y)
i=quad(f,.25,1.75)
print('L\'integrale demandée',i[0])
L'exécution du programme donne:
L'integrale demandée 2.5312499999999987

Exercice 3

On considère l'équation de Poisson 1D suivante:

$$\frac{d^2\phi}{dx^2}(x) = \square\rho(x) \qquad x \in [0,1]$$

avec $\phi(0) = \phi(1) = 0$ et $\rho(x) = \square x(x+3)e^x$

1- Utiliser la discrétisation de $[0,1]$, $0 = x_0 < x_1 < \cdots < x_N = 1$ avec un pas constant h.

Puis l'approximation: $\frac{d^2\phi}{dx^2}(x_i) \approx \frac{\phi(x_{i+1}) \square\ 2\phi(x_i) + \phi(x_{i\square 1})}{h^2}$ pour résoudre le problème précédent.

Construire la matrice du système linéaire avec la fonction **diag** de numpy.

2- Comparer avec la solution exacte: $\phi(x) = x(x \square\ 1)e^x$, pour cela:

Tracer dans une même figure la solution exacte et la solution approchée en rouge *

Puis calculer $erreur = \|\phi(x_i) \square\ \phi_i\|_\infty$

Corrigé

```
import numpy.linalg as ln
import matplotlib.pyplot as plt
f=lambda x: x*(x+3)*np.exp(x)
Nx=100
x=np.linspace(0,1,Nx)
xx=x[1:(Nx-1)]
h=1/(Nx-1)
A=-2*np.diag(np.ones(Nx-2))+np.diag(np.ones(Nx-3),k=-1)+np.diag(np.ones(Nx-3),k=1)
b=h**2*f(xx)
phi=ln.solve(A,b)
#solution exacte
sol=lambda x: x*(x-1)*np.exp(x)
phiexa=sol(xx)
print('erreur=',ln.norm(phiexa-phi,np.inf))
plt.plot(xx,phi,'r*',xx,phiexa)
plt.legend(['solution approchée','solution exacte'],loc=9)
plt.title('résolution de l équation de Poisson 1D')
plt.show()
```

L'execution du programme donne:
erreur= 2.21014588745e-05


résolution de l équation de Poisson 1D
solution approchée
solution exacte
0.00
−0.05
−0.10
0.15
−0.20
−0.25
−0.30
0.35
−0.40
−0.45
0.0
0.2
0.4
0.6
0.8
1.0