TP n°2 Corrigé : Méthodes de quadratures pour l'intégration numérique

1. **Méthodes de quadrature à 1 point :**
   (a) Définir la fonction f

```
function [y] = f(x)
y=%pi/2*sin(%pi/2*x);
endfunction
```

   (b) Implémenter la méthode de quadrature des rectangles à gauche pour une fonction `g` sur une subdivision régulière de `N` points dans l'intervalle $[0,1]$.

```
function I=rectG(g,N)
dx=1/N;
X=(0:dx:1-dx);
I=dx*sum(g(X));
endfunction
```

   Remarque : L'évaluation du vecteur `g(X)` impose une attention particulière à la manière dont on définit `g`. Par exemple pour `f(X)` Scilab réalise le calcul du `sin` d'un vecteur, ce qu'il interprète comme le vecteur des sinus des composantes de `X`. Si `g` est la fonction $x \mapsto x^2$, il faut utiliser la multiplication terme à terme du vecteur `X` dans la définition de `g`, par `y=x.*x` ou par `y=x.^2`.
   Pour le calcul exact de l'intégrale, on obtient à la main (ou éventuellement à l'aide de la commande `integrate('f(x)','x',0,1)` de scilab)

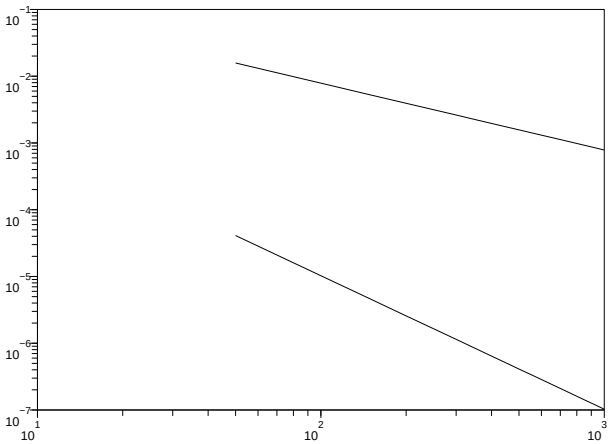
$$\int_0^1 f(t)\,dt = 1$$

   (c) Implémenter la méthode de quadrature du point milieu pour une fonction `g` sur une subdivision régulière de `N` points dans l'intervalle $[0,1]$.

```
function I=rectM(g,N)
dx=1/N;
X=(0:dx:1-dx)+dx/2;
I=dx*sum(g(X));
endfunction
```

   (d) Comparer ces deux méthodes - tracé de l'erreur en échelle log-log.

```
N=[50:50:1000];
IntG=[];
IntM=[];
for n=N
 IntG=[IntG,rectG(f,n)];
 IntM=[IntM,rectM(f,n)];
end;
// Calcul de l'erreur
errG=IntG-1;
errM=IntM-1;
// Graphe log-log de l'erreur
scf(0);
plot2d(N,abs(errG),logflag="ll")
plot2d(N,abs(errM),logflag="ll")
```



(e) La vitesse de convergence de chacune des deux méthodes, lorsque `N` tend vers l'infini est obtenue à partir de la pente des droites obtenues dans le tracé de l'erreur en échelle logarithmique. Par exemple :

```
disp('convergence - rectangle a gauche')
num=length(N);
-(log(errG(num))-log(errG(num-1)))/(log(num)-log(num-1))
disp('convergence - point milieu')
-(log(errM(num))-log(errM(num-1)))/(log(num)-log(num-1))
```

Qui retourne :

```
convergence - rectangle a gauche
ans  =
   1.0002686

convergence - point milieu
ans  =
   2.0000001
```

La méthode des rectangle à gauche fourni donc un résultat dont l'erreur décroit ici en $1/N$ tandis que la méthode du point milieu permet d'obtenir un résultat dont l'erreur décroit en $1/N^2$.

2. **Méthode de quadrature à 2 points :** On cherche ici à construire une méthode à deux points

(a) Lorsque $\theta = 0$, la quadrature élémentaire consiste à remplacer l'intégrale de f sur $[x_i, x_{i+1}]$ par la quantité $(x_{i+1} - x_i)f(x_i)$, elle correspond donc à la quadrature élémentaire de la méthode des rectangles à gauche. De même lorsque $\theta = 1$ on retrouve la méthode des rectangles à droites avec pour quadrature élémentaire $(x_{i+1} - x_i)f(x_{i+1})$.

(b) Implémenter la méthode proposée, en fonction de θ.

```
function I=DeuxPts(g,theta,N)
dx=1/N;
X=(0:dx:1-dx);
I=dx*sum(theta*g(X+dx)+(1-theta)*g(X));
endfunction
```

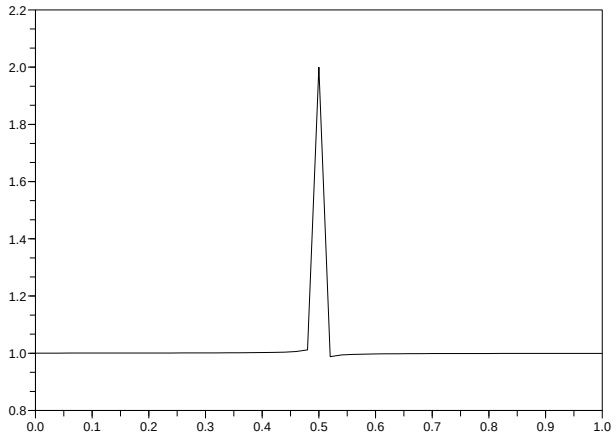
(c) Calcul de l'erreur en fonction de θ pour deux grandes valeurs de N.

```
Theta=[0:0.02:1];
Err1=[]; Err2=[];
n1=500;    n2=600;
for theta=Theta
 Err1=[Err1,DeuxPts(f,theta,n1)];
 Err2=[Err2,DeuxPts(f,theta,n2)];
end
```

(d) Tracé de la vitesse de convergence approximative déduite de ces erreurs (toujours vue comme l'opposé de la pente de la droite) et de ce fait évaluée via

$$\texttt{ordre}(\theta) \simeq -\frac{\log(\texttt{Err2}(\theta)) - \log(\texttt{Err1}(\theta))}{\log(\texttt{n2}) - \log(\texttt{n1})}$$

```
scf(1);
errDeuxPts1=abs(Resul1-1);
errDeuxPts2=abs(Resul2-1);
plot2d(Theta,(log(errDeuxPts2)-log(errDeuxPts1))/(log(n1)-log(n2)));
```



On remarque que la convergence est pour la plupart des valeurs de θ en $1/N$, sauf lorsque $\theta = 1/2$, pour lequel la convergence semble être en $1/N^2$.

(e) La valeur $\theta = 1/2$ correspond à la méthode des trapèzes, issue d'une quadrature élémentaire basée sur une interpolation P_1 (affine) de la fonction sur chaque intervalle :

$$f(x) \simeq f(x_i) + \frac{x - x_i}{x_{i+1} - x_i}(f(x_{i+1}) - f(x_i))$$

donnant une intégration approchée

$$\int_{x_i}^{x_{i+1}} f(t)\, dt \simeq (x_{i+1} - x_i)\left(\frac{f(x_{i+1}) + f(x_i)}{2}\right)$$

3. **Méthode de quadrature à 3 points :** La méthode de Simpson d'ordre 3

$$\int_{x_i}^{x_{i+1}} f(t)\, dt \simeq \frac{1}{6}(x_{i+1} - x_i)\left(f(x_i) + 4f\left(\frac{x_i + x_{i+1}}{2}\right) + f(x_{i+1})\right)$$

(a)
```
function I=Simpson(g,N)
dx=1/N;
X=(0:dx:1-dx);
I=sum(g(X)+4*g(X+dx/2)+g(X+dx))*dx/6;
endfunction
```

(b) Évaluation de la méthode pour des polynômes, d'intégrale 1 sur l'intervalle $[0, 1]$.

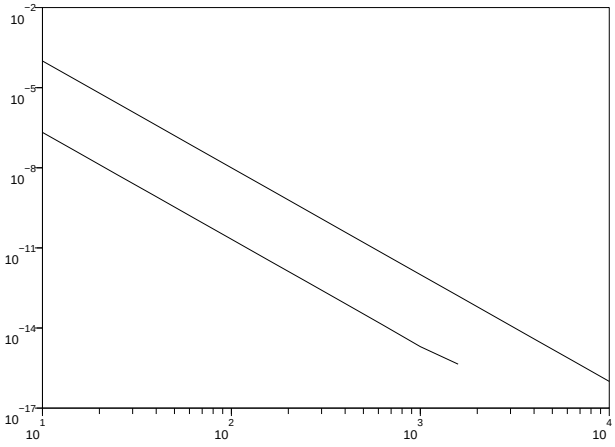
```
deff("y=x0(x)","y=1*x.^0");
deff("y=x1(x)","y=2*x.^1");
deff("y=x2(x)","y=3*x.^2");
deff("y=x3(x)","y=4*x.^3");
deff("y=x4(x)","y=5*x.^4");

disp('degre 0');
Simpson(x0,10)
disp('degre 1');
Simpson(x1,10)
disp('degre 2');
Simpson(x2,10)
disp('degre 3');
Simpson(x3,10)
disp('degre 4');
Simpson(x4,10)
```

```
degre 0
ans  =
   1.

degre 1
ans  =
   1.

degre 2
ans  =
   1.

degre 3
ans  =
   1.

degre 4
ans  =
   1.0000042
```

Le résultat est exact pour les polynômes `x0, x1, x2, x3` de degré inférieur ou égal à 3, donc par linéarité, ces polynômes formant une base de $\mathcal{P}_3$, elle est exacte pour tous les polynômes de degré inférieur ou égal à 3. En revanche, la valeur obtenue pour `x4`, polynôme de $\mathcal{P}_4$ présente une erreur. La méthode est donc bien d'ordre 3.

(c) Quelle est la vitesse de convergence ? On trace les courbes d'erreur et de $N \mapsto 1/N^4$ sur un même graphique, en échelle log-log.

```
scf(2);
N=int(10^[1:0.2:3])
IntS=[];
for n=N
 IntS=[IntS,Simpson(f,n)];
end;

errS=abs(IntS-1);
plot2d(N,errS,logflag="ll")
plot2d(N,N.^(-4),logflag="ll")
```



L'erreur décroit donc en $1/N^4$.