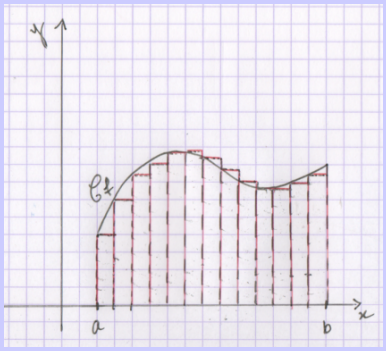
On note $(x_i)_{i\in\{0,\dots,n\}}$ une subdivision de l'intervale $[a,b]$

Avec $a = x_0 < x_1 < \dots < x_{n-1} < x_n = b$ et pour tout $i \in \{0,\dots,n\}$, $x_i = a + i\dfrac{b-a}{n}$

I Méthode des rectangles

Méthode des Rectangles



Elle consiste à approximer $\displaystyle\int_{x_i}^{x_{i+1}} f(t)dt$ par l'aire d'un rectangle de dimensions : $\dfrac{b-a}{n}$ et $f(x_i)$

On note $R_n(f) = \dfrac{b-a}{n}\displaystyle\sum_{k=0}^{n-1} f(x_i)$

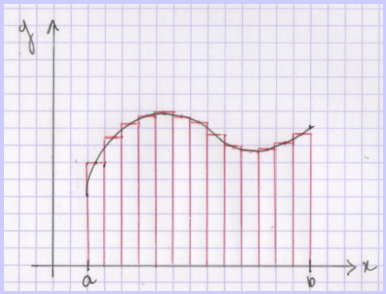
Une valeur approchée de $\displaystyle\int_a^b f(t)dt$ est $R_n(f)$

Propriété : Si f est monotone sur $[a,b]$ alors $\left|\displaystyle\int_a^b f(t)dt - R_n(f)\right| \leqslant \dfrac{b-a}{n}\left|f(b)-f(a)\right|$

Propriété : Si f est de classe C^1 sur $[a,b]$ alors $\left|\displaystyle\int_a^b f(t)dt - R_n(f)\right| \leqslant \dfrac{\|f'\|_\infty (b-a)^2}{2n}$

II Méthode des points milieux

Méthode des Points Milieux



Elle consiste à approximer $\displaystyle\int_{x_i}^{x_{i+1}} f(t)dt$ par l'aire d'un rectangle de dimensions : $\dfrac{b-a}{n}$ et $f\left(\dfrac{x_i+x_{i+1}}{2}\right)$

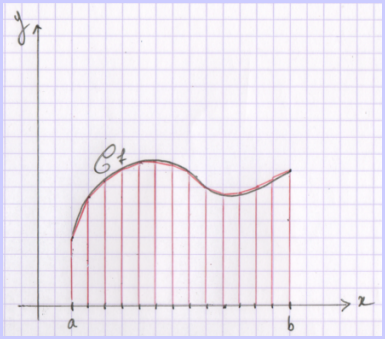
On note $M_n(f) = \dfrac{b-a}{n}\displaystyle\sum_{k=0}^{n-1} f\left(\dfrac{x_i+x_{i+1}}{2}\right)$

Une valeur approchée de $\displaystyle\int_a^b f(t)dt$ est $M_n(f)$

Propriété : Si f est de classe C^2 sur $[a,b]$ alors $\left|\displaystyle\int_a^b f(t)dt - M_n(f)\right| \leqslant \dfrac{\|f^{(2)}\|_\infty (b-a)^3}{24n^2}$

III Méthode des trapèzes

Méthode des trapèzes



Elle consiste à approximer $\int_{x_i}^{x_{i+1}} f(t)dt$ par l'aire d'un trapèze de dimensions : $h = \frac{b-a}{n}$, $b_1 = f(x_i)$ et $b_2 = f(x_{i+1})$

On note $T_n(f) = \frac{b-a}{2n} \sum_{k=0}^{n-1} (f(x_i) + f(x_{i+1}))$

Une valeur approchée de $\int_a^b f(t)dt$ est $T_n(f)$

Propriété : Si f est de classe C^2 sur $[a,b]$ alors $\left| \int_a^b f(t)dt - T_n(f) \right| \leqslant \frac{\|f^{(2)}\|_\infty (b-a)^3}{12n^2}$

IV Méthode de Simpson

Méthode de Simpson

Elle consiste à approximer f sur $[x_i; x_{i+1}]$ par la fonction polynômiale de degré au plus 2 prenant les mêmes valeurs que f en x_i, $\frac{x_i + x_{i+1}}{2}$ et x_{i+1}

On note $R_n(f) = \frac{b-a}{6n} \sum_{k=0}^{n-1} \left(f(x_i) + 4f\left(\frac{x_i + x_{i+1}}{2}\right) + f(x_{i+1}) \right)$

Une valeur approchée de $\int_a^b f(t)dt$ est $R_n(f)$

Propriété : Si f est de classe C^4 sur $[a,b]$ alors $\left| \int_a^b f(t)dt - R_n(f) \right| \leqslant \frac{\|f^{(4)}\|_\infty (b-a)^5}{2880n^4}$

Propriété : Si $f : [a,b] \to \mathbb{R}$ de classe C^4 alors $S_n(f) = \frac{T_n(f) + 2M_n(f)}{3}$

V Méthode d'accélération de Richardson-Romberg

Méthode d'accélération de Richardson-Romberg

On pose $T_{n,o} = T_{2^n}$, $T_{n,1} = \frac{4T_{n,0} - T_{n-1,0}}{3}$ et $T_{n,k} = \frac{2^{2k} T_{n,k-1} - T_{n-1,k-1}}{2^{2k} - 1}$

Propriété : $T_{n,p} - \int_a^b f(t)dt = O\left(\frac{1}{2^{2np}}\right)$

VI Méthode de Monte-Carlo

Méthode de Monte Carlo

Si (X_i) est une suite de variables aléatoires de loi uniforme sur $[a,b]$ alors d'après la loi des grands nombres :

$$\frac{1}{n}\sum_{i=0}^{n-1} f(X_i) \to E(f(X_i))$$

Une valeur approchée de $\int_a^b f(t)dt$ est $\frac{1}{n}\sum_{i=0}^{n-1} f(X_i)$

$\epsilon_n = \frac{1}{n}\sum_{i=0}^{n-1} f(X_i) - E(f(X_i))$ est avec une quasi certitude de 95% inférieur à $\sigma\frac{1,96}{\sqrt{n}}$

Pour calculer σ :

$$\frac{1}{n}\sum_{i=0}^{n-1} (f(X_i))^2 - \left(\frac{1}{n}\sum_{i=0}^{n-1} f(X_i)\right)^2 \to \sigma^2$$

La méthode de Monte-Carlo est intéressante pour des intégrales doubles. Exemple : $\int_{[a,b]}\int_{[c,d]} f(x,y)dxdy$

VII Comparaison

Comparaison

```
import math
import random

def f(x):
        return 4/(1+x*x);

def Rectangle(a,b,n):
        S=0;
        for i in range(n):
                x=a+i*(b-a)/n;
                S=S+f(x);
        S=((b-a)/n)*S;
        return S;
#****************METHODE DES RECTANGLES  ********************
def Trapeze(a,b,n):
        S=0;
        for i in range(n):
                x=a+i*(b-a)/n;
                y=a+(i+1)*(b-a)/n;
                S=S+f(x)+f(y);
        S=((b-a)/(2*n))*S;
        return S;
#*************METHODE DES POINST MILIEUX ********************
def Milieu(a,b,n):
        S=0
        for i in range(n):
                x=a+i*(b-a)/n;
                y=a+(i+1)*(b-a)/n;
                x=(x+y)/2;
                S=S+f(x);
        S=((b-a)/n)*S;
        return S;
```

```
#*************METHODE DE SIMPSON ********************
def Simpson(a,b,n):
        S=0;
        for i in range(n):
                x=a+i*(b-a)/n;
                y=a+(i+1)*(b-a)/n;
                z=(x+y)/2;
                S=S+f(x)+4*f(z)+f(y);
        S=((b-a)/(6*n))*S;
        return S;
#*************METHODE DE ROMBERG ********************
def Romberg(a,b,n):
        S=(4*Trapeze(a,b,2*n)-Trapeze(a,b,n))/(3);
        return S;
#*************METHODE DE MONTE CARLO ********************
def MonteCarlo(a,b,n):
        S=0;
        for i in range(n):
                x=random.uniform(0,1);
                S=S+f(x);
        S=(1/n)*S;
        return S;
#*************COMPARAISON ********************
def Integrale(a,b,n):
        S=Rectangle(a,b,n);
        print('Rectangle : ',S,'| Reste= ',abs(math.pi-S));
        S=Trapeze(a,b,n);
        print('Trapeze : ',S,'| Reste= ',abs(math.pi-S));
        S=Milieu(a,b,n);
        print('Milieu : ',S,'| Reste= ',abs(math.pi-S));
        S=Simpson(a,b,n);
        print('Simpson : ',S,'Reste= ',abs(math.pi-S));
        S=Romberg(a,b,n);
        print('Romberg : ',S,'| Reste= ',abs(math.pi-S));
        S=MonteCarlo(a,b,n);
        print('Monte Carlo : ',S,'| Reste= ',abs(math.pi-S));
```

```
Python 3.3.4 (v3.3.4:7ff62415e426, Feb 10 2014, 18:12:08) [MSC v.1600 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
>>> Integrale(0,1,10000)
Rectangle :  3.14169265192314 | Reste=  9.999833334672914e-05
Trapeze :  3.1415926519231303 | Reste=  1.6666628077643963e-09
Milieu :  3.141592654423134 | Reste=  8.3334095180021e-10
Simpson :  3.1415926535897754 Reste=  1.7763568394002505e-14
Romberg :  3.1415926535897647 | Reste=  2.842170943040401e-14
Monte Carlo :  3.1510562328508414 Reste=  0.009463579261048238
>>> math.pi
3.141592653589793
>>>
```