T.D. n°6 Analyse Numérique

EMA

Intégration numérique
Méthodes aléatoires
et déterministes.

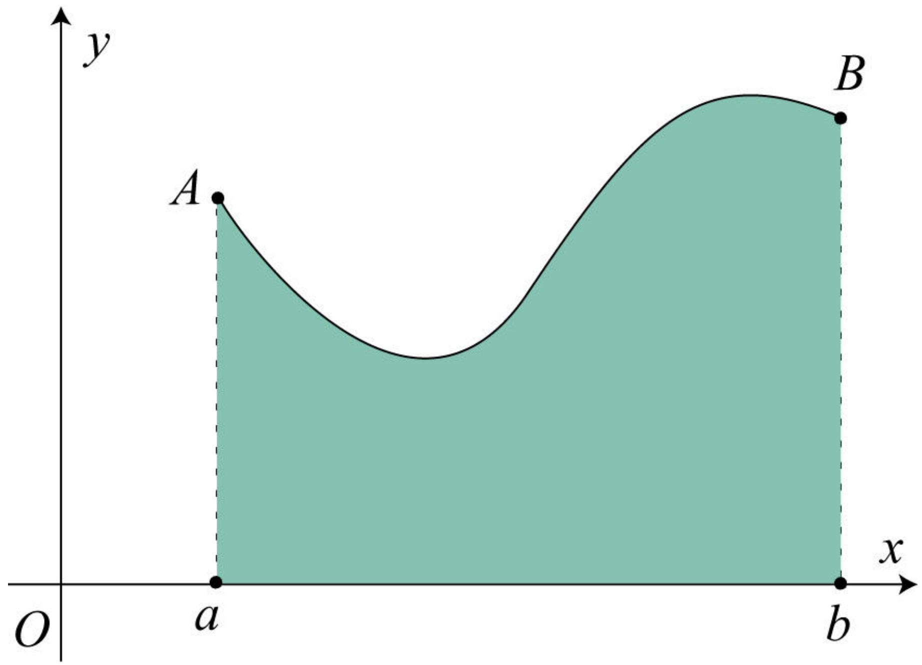
EMA

Introduction :
Calcul d’une intégrale

On cherche à calculer une valeur approchée de :

$$I = \int_a^b f(x)\,\mathrm{d}x$$

pour f suffisamment régulière sur le segment $[a, b]$.



Intégration numérique

Première partie : Méthodes aléatoires

dites « de Monte-Carlo »

Il s'agit de méthodes basées sur l'idée que l'Espérance d'une variable aléatoire se calcule comme une intégrale d'une fonction.

Et que l'espérance peut être estimée par la fréquence d'un échantillon.

I) Méthodes aléatoires

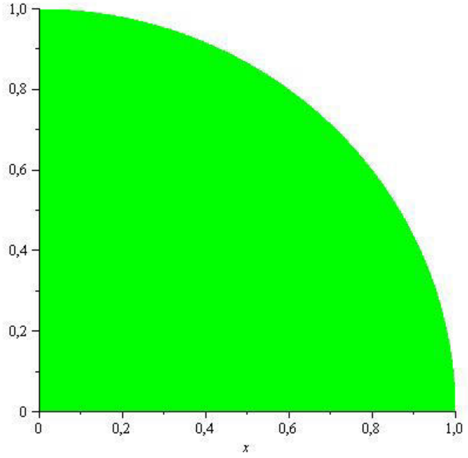
dites « de Monte-Carlo »

- Idée : pour f positive, on inclut la surface sous la courbe, $I = \int_a^b f(x)\,\mathrm{d}x$, dans un pavé $[a,b] \times [c,d]$, de surface $Surf_0$, avec $Surf_0 = (b-a)(d-c)$.

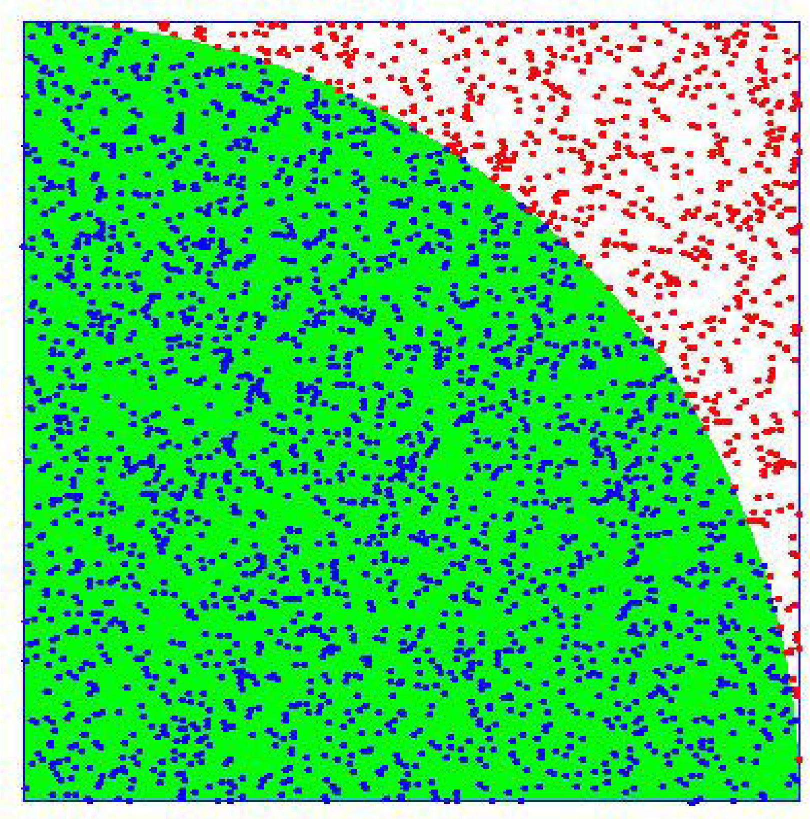
On choisit *Ntirages* points $(x_i, y_i)_{1 \le i \le Nt}$ aléatoirement équirépartis dans ce pavé. On teste si ces points vérifient $f(x_i) \le y_i$ et on compte les *Nchoisis* points sous la courbe parmi les points tirés. Alors la surface-intégrale I est approchée par la proportion

$$Surf_1 = \frac{Nchoisis}{Ntirages} \times Surf_0$$

I) Méthodes aléatoires



- Pour la fonction $y=\sqrt{1-x^2}$ sur $[0,1]$: on l'inclut dans le pavé $[0,1]\times[0,1]$, et on propose *Ntirages* points équirépartis aléatoirement dans ce pavé.



- On compte parmi ces points, ceux qui véri--fient $y\le\sqrt{1-x^2}$. Il y en a *Nchoisis*.
- Alors :

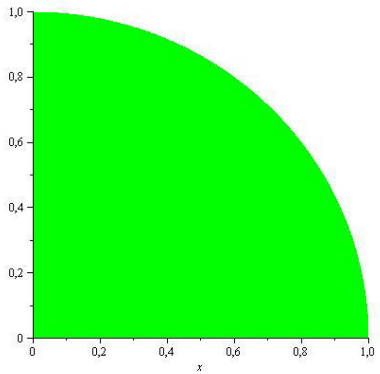
$$\frac{Nchoisis}{Ntirages}\approx\frac{intégrale}{aire\ du\ \ pavé}$$

I) Méthodes aléatoires

- Pour la fonction

$y = \sqrt{1 - x^2}$

sur $[0,1]$:



- $\int_0^1 \sqrt{1 - x^2}\, dx = \frac{\pi}{4}$

```
def Monte_Carlo_1(n):
    Compteur = 0
    for i in range(n):
        x=np.random.rand()
        y=np.random.rand()
        if x**2+y**2<1:
            Compteur += 1
    resul =  Compteur / n
    return resul
n = 1000000
Approx =4* Monte_Carlo_1(n)
print('Approx_pi =', Approx)
print('Erreur = ',abs(np.pi - Approx))
```

Programme TD6_I_1

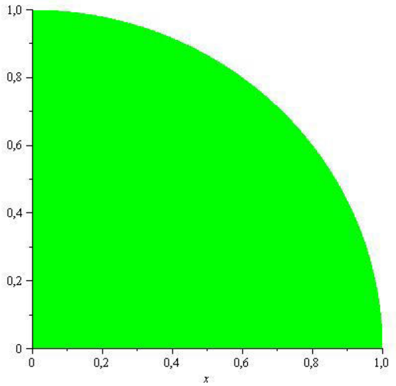
I) Méthodes aléatoires

Variante

- Pour la fonction

  $y=\sqrt{1-x^2}$

  sur $[0,1]$:



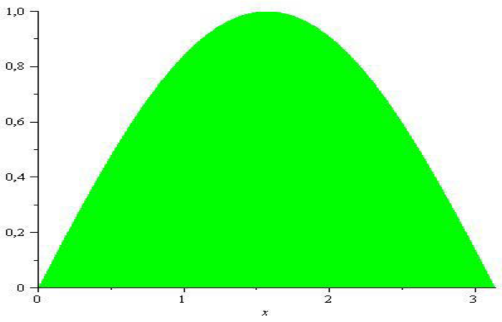
- $\int_0^1 \sqrt{1-x^2}\,\mathrm{d}x = \dfrac{\pi}{4}$

```
def Monte_Carlo_2(n):
    x=np.random.rand(n)
    y=np.random.rand(n)
    Compteur=np.sum(x**2+y**2<1)
    resul = Compteur / n
    return resul
n = 1000000
Approx =4* Monte_Carlo_2(n)
print('Approx_pi =', Approx)
print('Erreur = ',abs(np.pi - Approx))
```

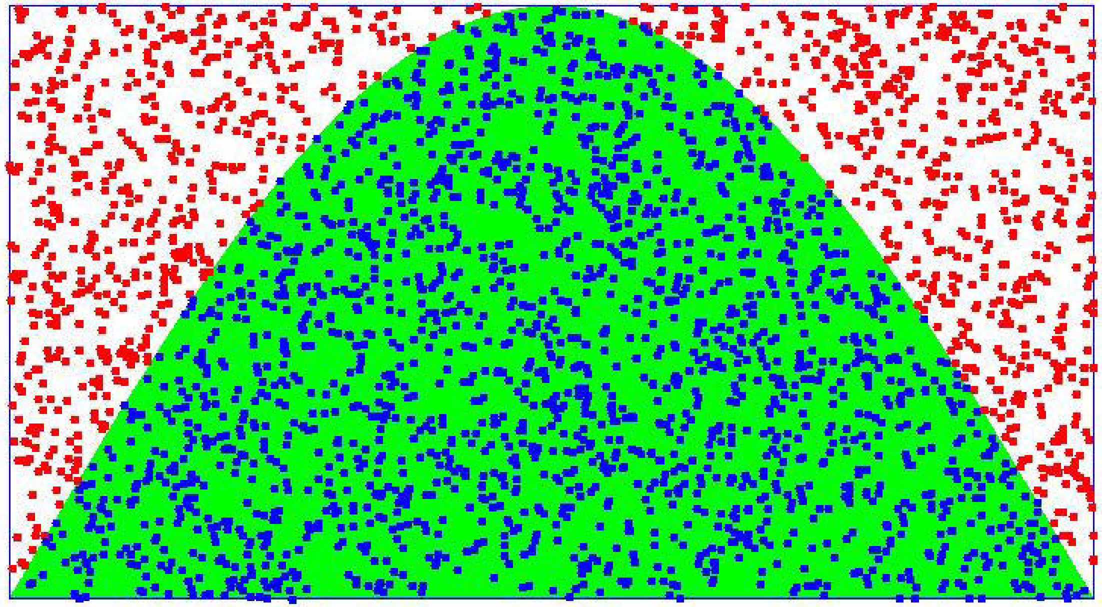
x.^2+y.^2<1 est un booléen qui vaut 1 si c'est vrai et 0 sinon. sum(x.^2+y.^2<1) calcule donc la somme de ces 1.

Programme TD6_I_2

I) Méthodes aléatoires



- Pour la fonction $y = \sin(x)$ sur $[0,\pi]$:
  on l'inclut dans le pavé $[0,\pi]\times[0,1]$, et on propose *Ntirages* points équirépartis aléatoirement dans ce pavé.



- On compte parmi ces points, ceux qui véri-
  -fient $y \le \sin(x)$.
  Il y en a *Nchoisis*.

- Alors :

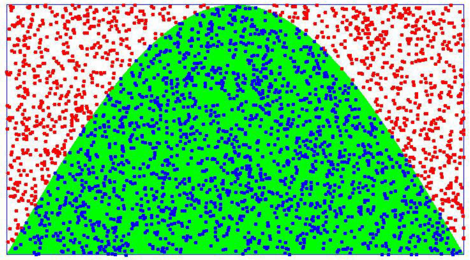
$$\frac{Nchoisis}{Ntirages} \approx \frac{intégrale}{aire\ du\ \ pavé}$$

I) Méthodes aléatoires

- Pour la fonction

  $y = \sin(x)$

  sur $[0, \pi]$:



- $\int_0^{\pi} \sin(x)\,\mathrm{d}x = 2$

```
def Monte_Carlo(f , a , b , c , d , n):
    Surf0=(b-a)*(d-c)
    x=a+(b-a)*np.random.rand(n)
    y=c+(d-c)*np.random.rand(n)
    Nchoisis=np.sum(y-f(x)<0)
    SurfApprox = Surf0*Nchoisis / n
    return SurfApprox
# Donnees
f=lambda x:np.sin(x)
a , b , c , d = 0. , np.pi , 0. , 1.
n = 3000
Approx = Monte_Carlo(f,a,b,c,d,n)
print( 'Integrale = ' , Approx)
```

Programme TD6_I_3

I) Méthodes aléatoires

Ces méthodes ont une convergence assez lente, en $O\left(\frac{1}{\sqrt{n}}\right)$.

Par contre, elles sont très efficaces en dimensions supérieures à 1, car on peut se dispenser de délimiter les domaines pour appliquer un Théorème de Fubini, pour les intégrations successives.

Et on peut travailler en (très) grandes dimensions.

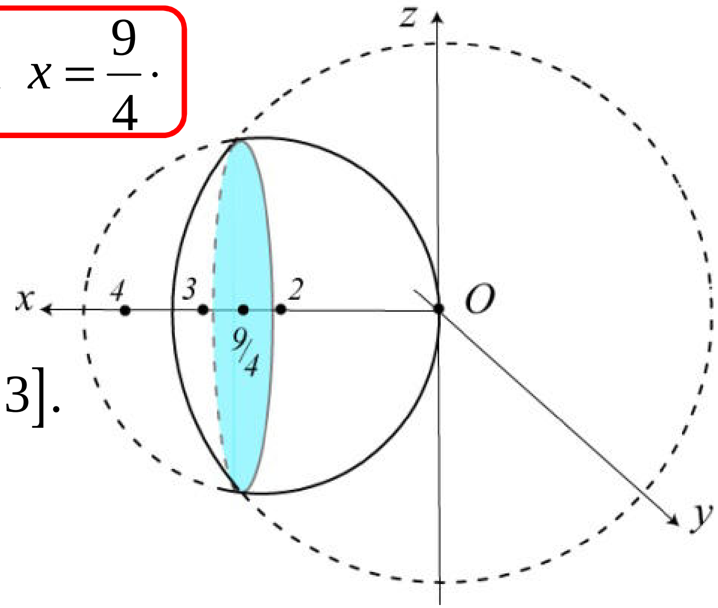
II) En dim 3 Exemple de méthode de Monte-Carlo

Démonstration non exigible.

- On considère le volume intérieur à deux sphères :

$$x^2+y^2+z^2=9 \text{ et } (x-2)^2+y^2+z^2=4$$

$$\boxed{\text{Leur intersection est un cercle dans le plan } x=\frac{9}{4}.}$$



- Le domaine $x^2+y^2+z^2\le 9$

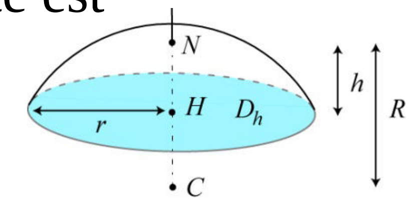
  et $(x-2)^2+y^2+z^2\le 4$

  est inclus dans le pavé $[0,4]\times[-3,3]\times[-3,3]$.

- Ce volume est constitué de la réunion de deux "calottes sphériques".

- Le volume d'une calotte est

$$V(h)=\frac{\pi h^2}{3}(3R-h).$$



- Et $V=\left(\frac{1215}{3\times 64}+\frac{99}{64}\right)\pi=\frac{1512}{192}\pi$

  $\approx 24{,}7400421$

II) Méthodes aléatoires

```
def MonteCarlo3D(n):
    Compteur , Vol0 = 0 , 7*6*6
    for k in range(n):
        x = -3 + 7*np.random.rand()
        y = -3 + 6*np.random.rand()
        z = -3 + 6*np.random.rand()
        if x**2+y**2+z**2<=9 and (x-2)**2+y**2+z**2<=4:
            Compteur+=1
    return Vol0*Compteur/n
n = 1000000
Approx = MonteCarlo3D(n)
print(' Volume = ', Approx)
VolExact=1512*np.pi/192
print('Erreur = ',abs(VolExact - Approx))
```

Programme TD6_II

Intégration numérique

Deuxième partie :

Méthodes déterministes.

Elles sont basées sur le calcul des intégrales de polynômes d'interpolation (Lagrange), de différents degrés, sur différents supports :

$$\int_a^b f(t)\,\mathrm{d}t \quad \text{approchée par} \int_a^b P(t)\,\mathrm{d}t$$

III) Méthodes déterministes

On dispose donc de multiples méthodes de calcul d'approximation d'intégrales :

Méthodes des trapèzes, de Simpson, de Newton,
de Cotes, de Hardy, adaptatives, etc.

Python les utilise dans des fonctions natives, pour des segments ou des domaines en tranches verticales :

quad(f,a,b) pour $\int_a^b f(t)\,\mathrm{d}t$ éventuellement avec une tolérance : tol

dblquad(f,a,b,c,d) avec des **fonctions** c et d,

pour $\int_a^b \left(\int_{c(x)}^{d(x)} f(x,y)\,\mathrm{d}y \right) \mathrm{d}x$

III) Méthodes déterministes 1)

```
from scipy.integrate import quad
# Fonctions
def f(x):
    return x**4*np.log(x+np.sqrt(x**2+1))
# Programme
xmin, xmax, ymin, ymax = 0.0 , 2.0 , -0.5 , 25.0
res , err = quad(f, xmin, xmax)
print('Integrale =', res)
print('Erreur evaluee =',err)
```

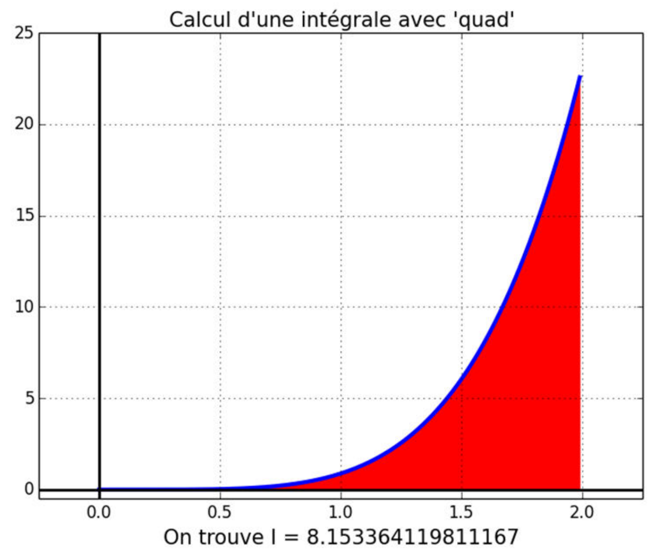
Programme TD6_III_1

Résultats : Integrale = 8.153364119811167
Erreur evaluee = 9.052052573966981e-14

III) Méthodes déterministes 1)

Programme
TD6_III_1 (suite)

```
# Visualisation graphique
bornes=[xmin-0.25,xmax+0.25,ymin,ymax]
xpas=0.01
x=np.arange(xmin, xmax, xpas)
fig=plt.figure(1)
ax=fig.add_subplot(111)
ax.grid(True)
plt.plot(x, f(x),'b',linewidth=3)
plt.axis(bornes)
ax.fill_between(x, 0, f(x), color='r')
plt.title("Calcul d'une intégrale ",fontsize=15)
plt.xlabel('On trouve I = '+str(res),fontsize=15)
```



III) Méthodes déterministes 2)

```
from scipy.integrate import dblquad
# Fonctions
def f(x,y):
	return np.exp(-x**2-y**2)
Ymin=lambda x:ymin
Ymax=lambda x:ymax
xmin , xmax , ymin , ymax = 0.0 , 5.0 , 0.0 ,5.0
res , err = dblquad(f, xmin, xmax, Ymin, Ymax)
print('Integrale calculee =\n', res)
print('Erreur estimee =\n',err)
```

Programme TD6_III_2

Résultats : Integrale = 0.7853981633950333

Erreur estimee = 2.092348318287337e-14

Un peu de mathématiques …

Rappel : $I_0 = \iint_{\mathbb{R}^2} \exp\left(-x^2 - y^2\right) \mathrm{d}x\,\mathrm{d}y = \int_{-\pi}^{\pi}\int_0^{\infty} \exp\left(-\rho^2\right)\rho\,\mathrm{d}\rho\,\mathrm{d}\theta$

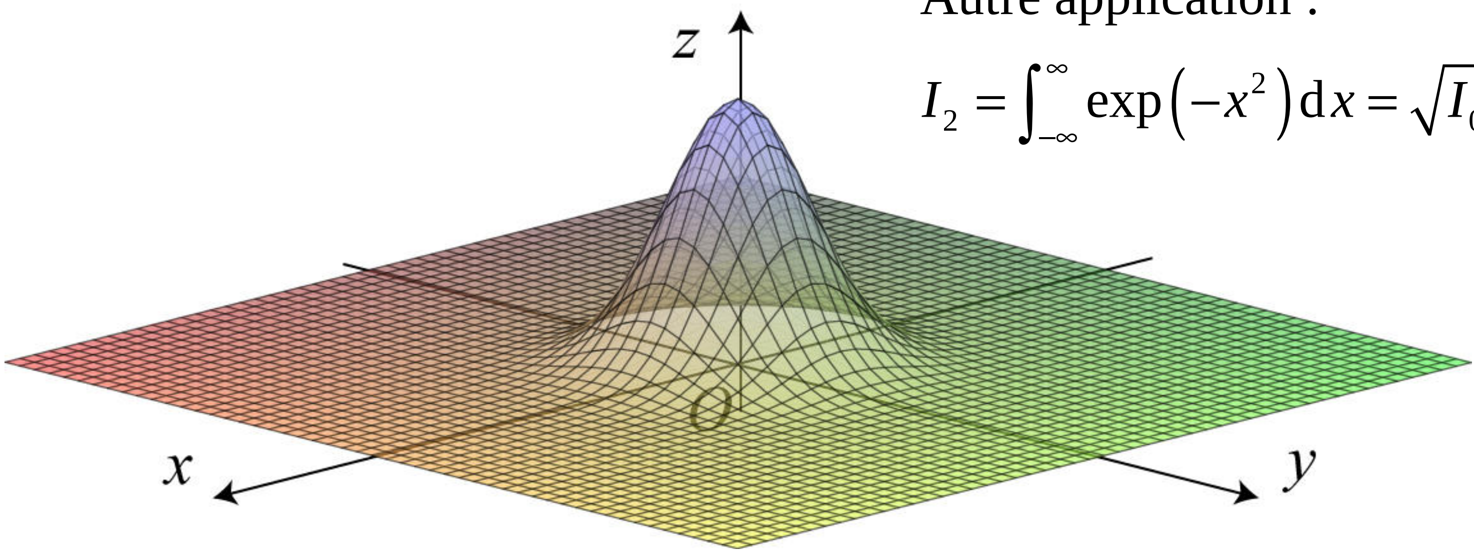
en passant en polaires.

$$I_0 = \int_{-\pi}^{\pi}\left[-\frac{1}{2}\exp\left(-\rho^2\right)\right]_{\rho=0}^{\rho\to\infty}\mathrm{d}\theta = (2\pi)\times\left(\frac{1}{2}\right) = \pi$$

Donc : $I_1 = \int_0^{\infty}\int_0^{\infty} \exp\left(-x^2 - y^2\right)\mathrm{d}x\,\mathrm{d}y = \frac{\pi}{4}$

Autre application :

$I_2 = \int_{-\infty}^{\infty} \exp\left(-x^2\right)\mathrm{d}x = \sqrt{I_0} = \sqrt{\pi}$

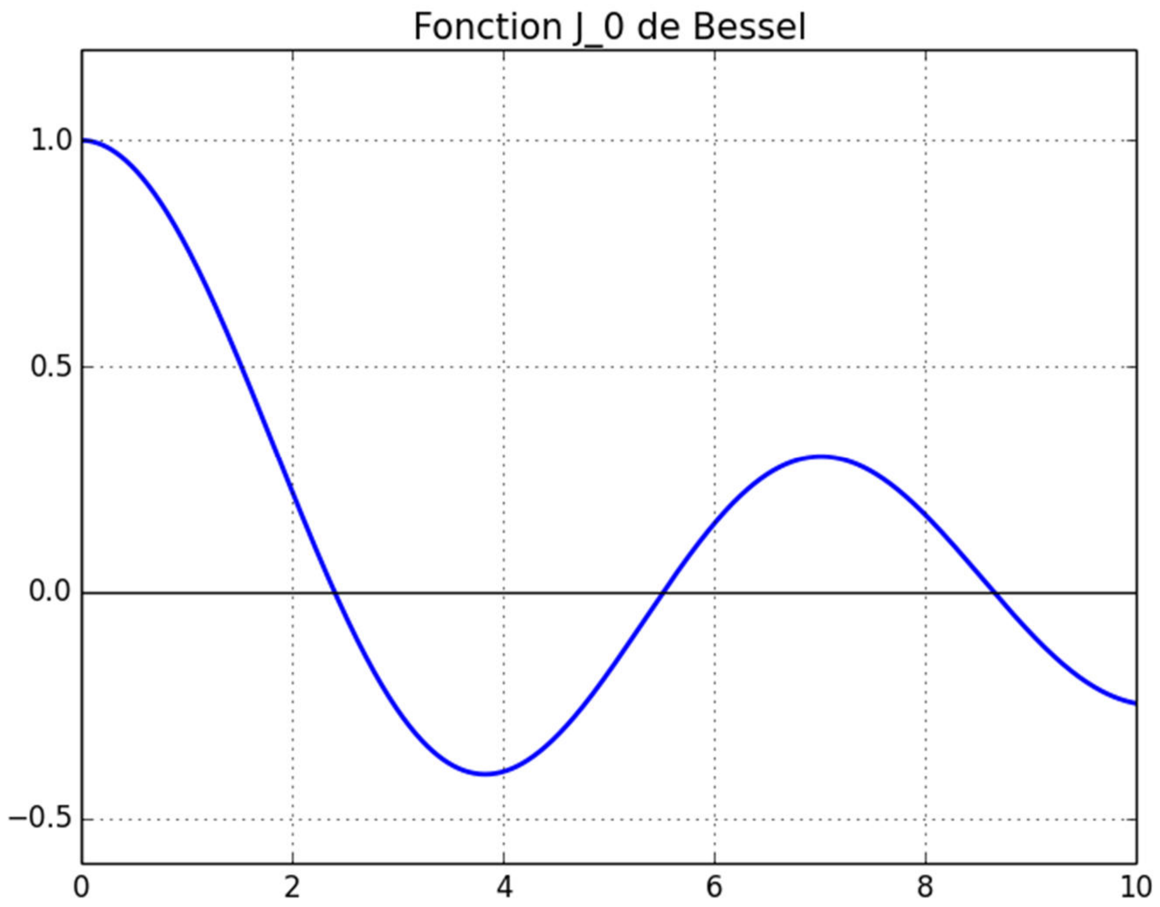


III) Méthodes déterministes 3)

Programme TD6_III_3

```
# Fonctions
def f(x,u):
    return np.cos(x*np.sin(u))/np.pi
xmin,xmax,xpas=0.0,10.0,0.01
umin,umax=0.0,np.pi
xx=np.arange(xmin,xmax,xpas)
y=[]
for x in xx:
    y.append(quad(lambda u: f(x,u),umin,umax))
plt.figure(1)
yy=[y[k][0] for k in range(len(xx))]
plt.plot(xx,yy,'r',linewidth=2)
```

III) Méthodes déterministes 3)

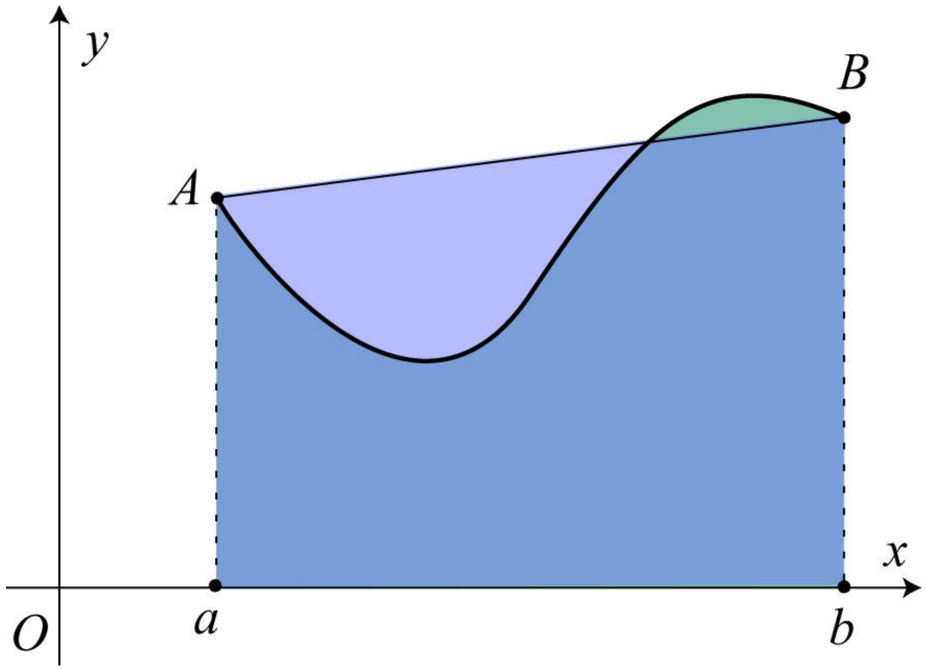


Programme TD6_III_3

IV) Méthodes déterministes du Trapèze

On utilise le polynôme de Lagrange de degré 1 (droite).
On peut approcher l'intégrale par la surface du trapèze *AabB*.

$$I_T = \frac{b-a}{2}\left[f(a) + f(b) \right]$$



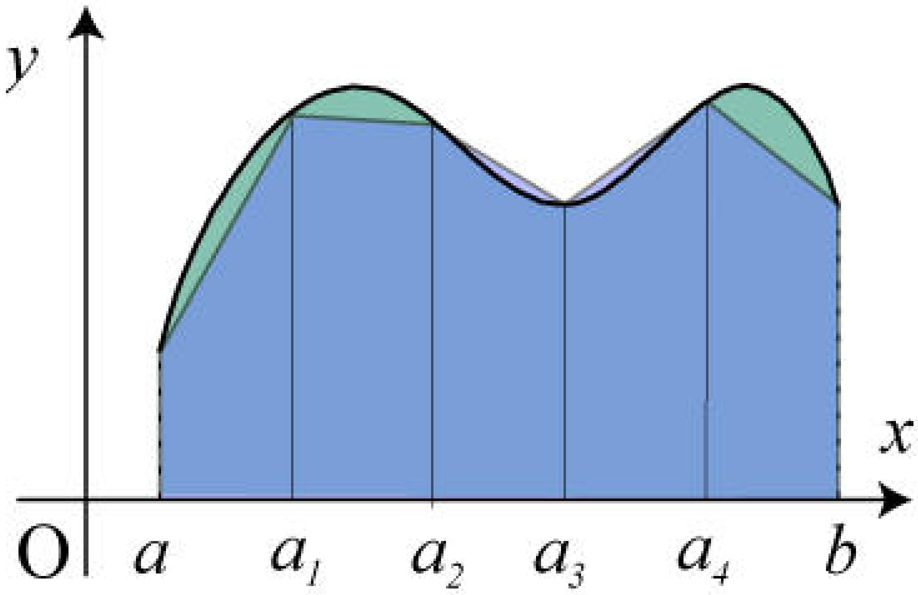
Incertitude de méthode de la méthode du Trapèze :

Si f est C^2, il existe $c \in [a,b]$ tel que $I - I_T = -\frac{(b-a)^3}{12} f''(c).$

IV) Intégration par arcs

Idée générale :

On partage $[a, b]$ en n sous-segments égaux et on applique une méthode sur chaque $[a_j, a_{j+1}]$ où $a_j = a + j\,\frac{b-a}{n}$



- Pour les trapèzes par arcs :

$$I_{T,n} = \frac{b-a}{2n}\left[f(a) + f(b) + 2\sum_{j=1}^{j=n-1} f\left(a_j\right)\right]$$

- Incertitude de méthode : si f est C^2,

il existe $c \in [a,b]$ tel que $I_{T,n} - I = \frac{(b-a)^3}{12n^2} f''(c)$.

IV) Intégration par arcs 1)

```
f=lambda x:np.sin(x)
def Trapeze( xmin , xmax , nbx , f ) :
    xpas=(xmax-xmin)/(nbx-1)
    x=np.linspace(xmin,xmax,nbx)
    y=f(x)
    I=0
    for k in range(nbx-1):
        I+=xpas*(y[k]+y[k+1])/2
    return I
xmin , xmax , nbx = 0.0 , np.pi , 30
I1=Trapeze( xmin , xmax , nbx , f )
print(' Trapèzes =\n ' , I1 )
```

Programme TD6_IV_1

Résultats : Trapèzes =
1.99804369097

IV) Intégration par arcs 2)

Python fournit une fonction implémentée : **trapz**(y,x) qu'on peut comparer à notre programmation.

```
f=lambda x:np.sin(x)
xmin , xmax , nbx = 0.0 , np.pi , 30
x=np.linspace(xmin,xmax,nbx)
y=f(x)
I2=np.trapz(y,x)
print(' Intégrale avec trapz de numpy =\n ' , I2)
```

Programme TD6_IV_2

Résultats :

Intégrale avec trapz de numpy = 1.99804369097

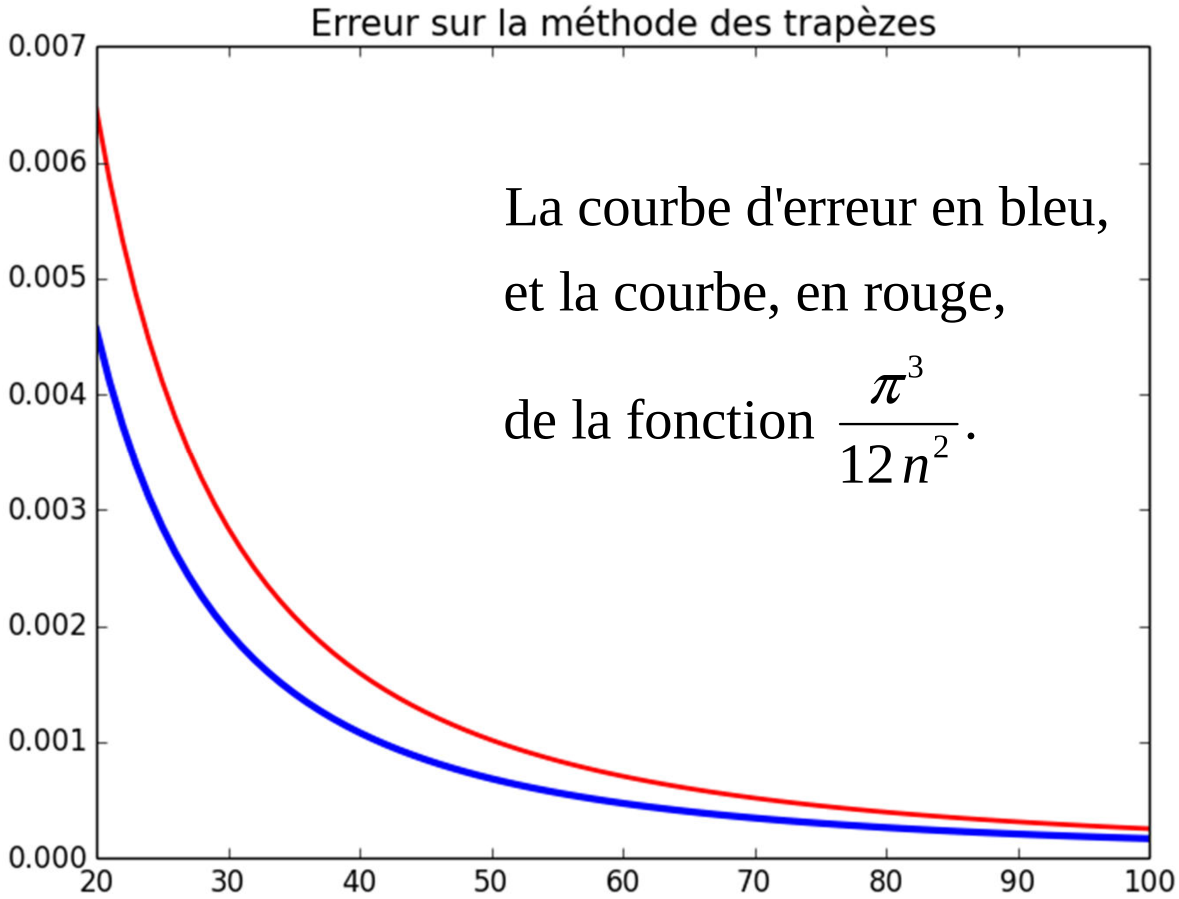
IV) Intégration par arcs 3)

Programme TD6_IV_3

```
xmin , xmax = 0.0 , np.pi
nmin , nmax = 20 , 101
Xn = range(nmin,nmax)
IT=[]
for nbx in Xn :
     x=np.linspace(xmin,xmax,nbx)
     y=f(x)
     IT.append(np.trapz(y,x))
plt.figure(1)
plt.plot( Xn , abs(2.0-np.array(IT)), 'b', linewidth=3)
plt.title('Erreur des trapèzes', fontsize=15)
```

IV) Intégration par arcs 3)

Programme TD6_IV_3

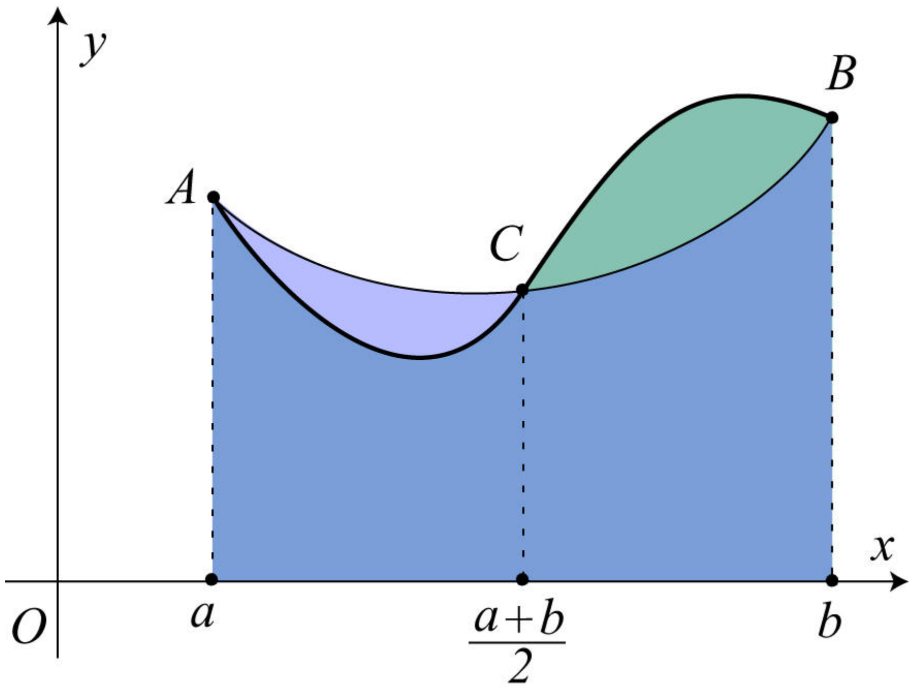


La courbe d'erreur en bleu,
et la courbe, en rouge,
de la fonction $\dfrac{\pi^3}{12\,n^2}$.

IV) Méthodes de Simpson

On utilise le polynôme de Lagrange de degré 2 (parabole).
On peut approcher l'intégrale par la surface sous la parabole interpolant f aux points A, B et C.

$$I_S = \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b) \right]$$



Incertitude de méthode de la méthode de Simpson :

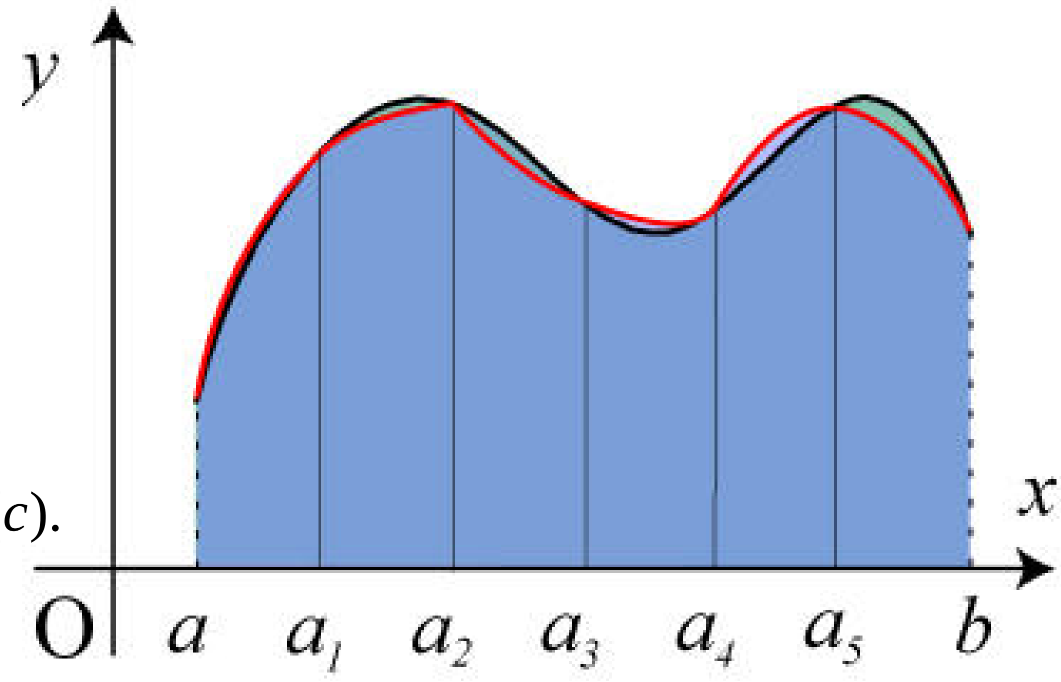
Si f est C^4, il existe $c \in [a,b]$ tel que $I - I_S = -\frac{(b-a)^5}{2^5 \times 90} f^{(4)}(c)$.

IV) Intégration par arcs

- Pour Simpson par arcs, on découpe $[a,b]$ en $n=2p$ (donc pair) :

$$I_{S,n}=\frac{b-a}{3\times(2p)}\left[f(a)+f(b)+2\sum_{j=1}^{j=p-1}f\left(a_{2j}\right)+4\sum_{j=0}^{j=p-1}f\left(a_{2j+1}\right)\right]$$

- On considère p paraboles, interpolant f en trois points successifs, a_{2k}, a_{2k+1} et a_{2k+2}.



- Incertitude de méthode :

si f est C^4, il existe $c\in[a,b]$

tel que $\quad I_{S,n}-I=\dfrac{(b-a)^5}{2880\ n^4}\,f^{(4)}(c).$

IV) Intégration par arcs 4)

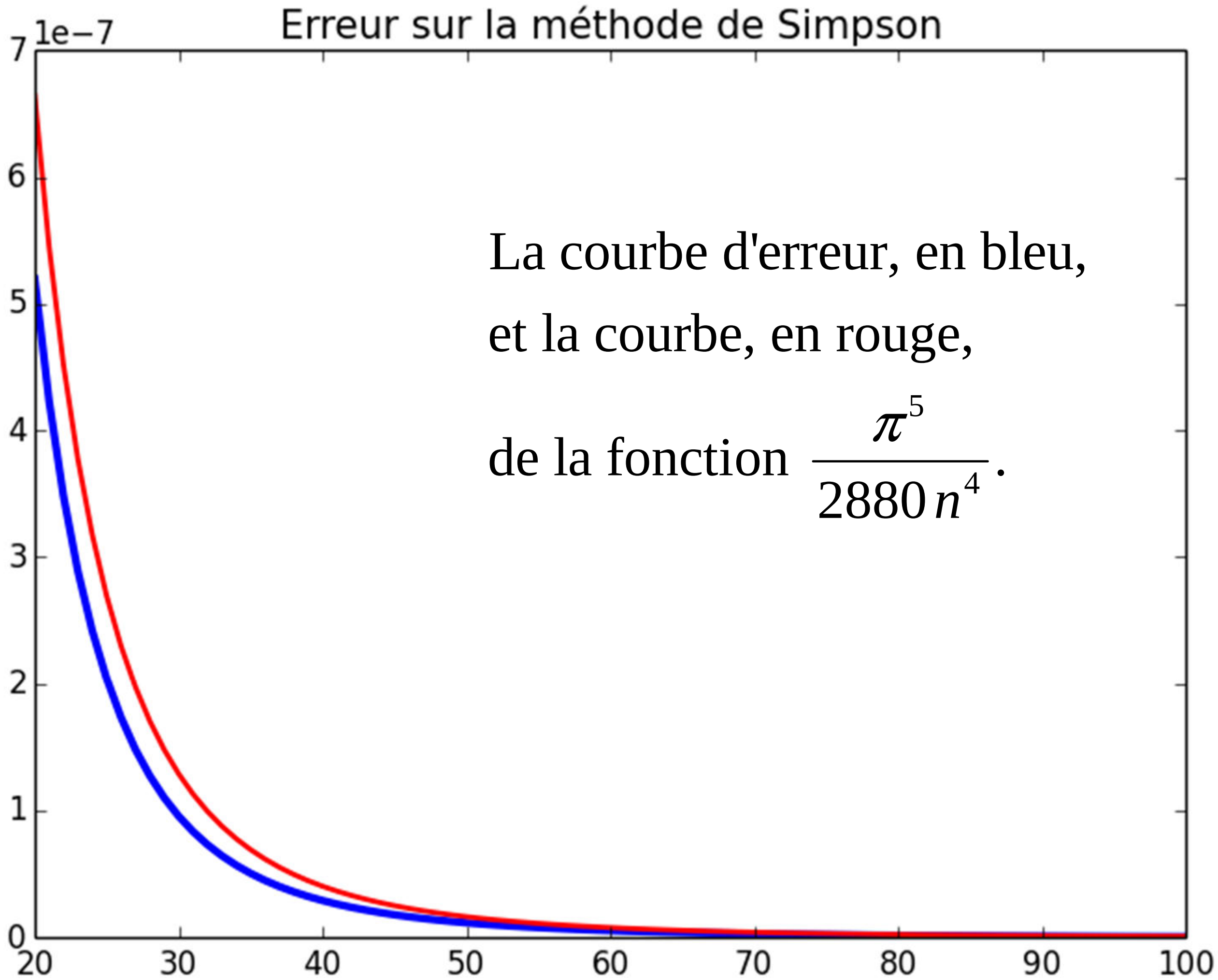
```
f=lambda x:np.sin(x)
def Simpson(xmin,xmax,nbx,f):
    xpas=(xmax-xmin)/(nbx-1)
    x=np.linspace(xmin,xmax,nbx)
    y=f(x)
    I=0
    for k in range(nbx-1):
        yy=f((x[k]+x[k+1])/2)
        I+=xpas*(y[k]+4*yy+y[k+1])/6
    return I
xmin , xmax , nbx = 0.0 , np.pi , 30
I3=Simpson( xmin , xmax , nbx , f )
print(' Simpson =\n ' , I3 )
```

Programme TD6_IV_4_a

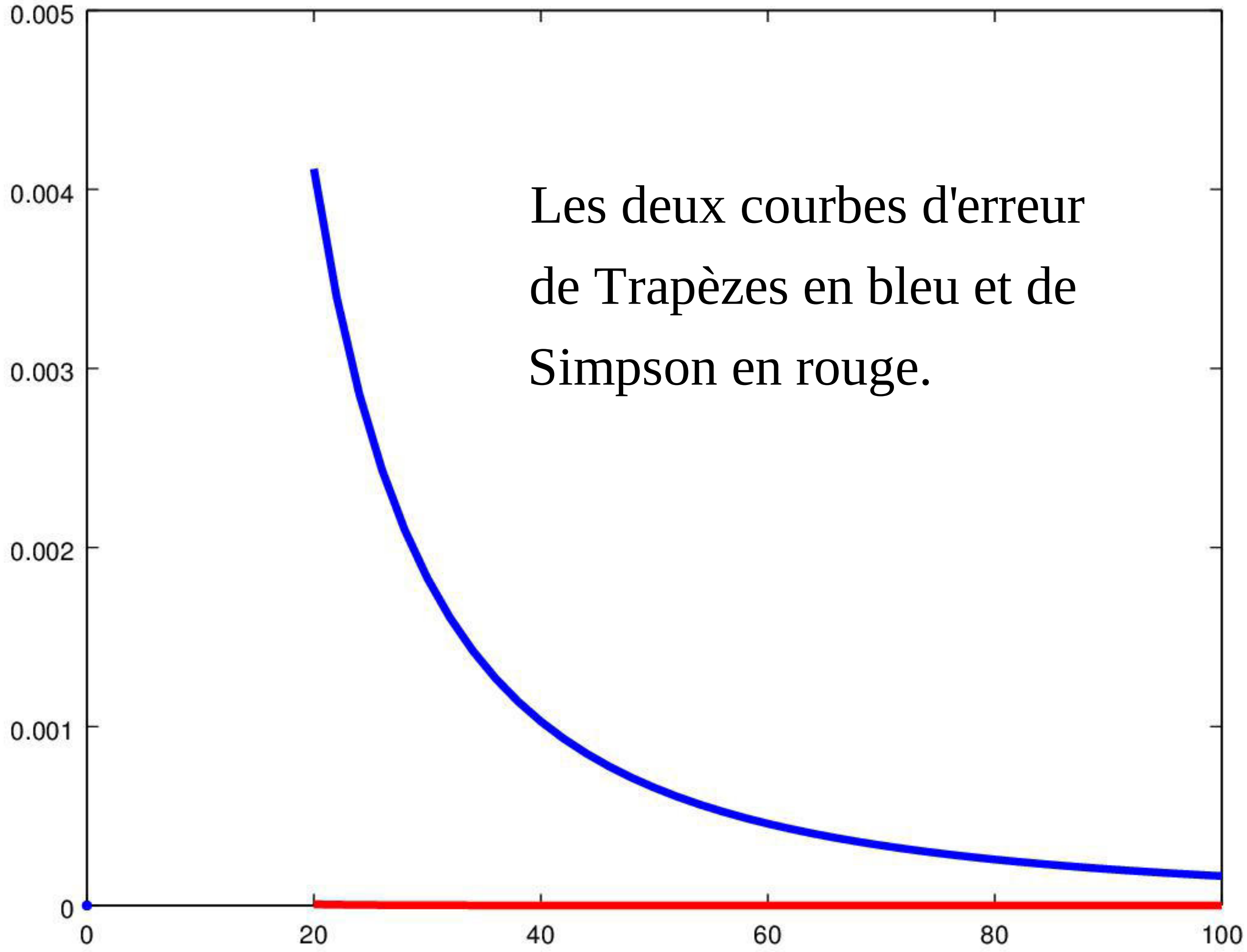
Résultats : Simpson = 2.00000009567

IV) 4) Intégration par arcs

Programme
TD6_ExoIV4



IV) Intégration par arcs

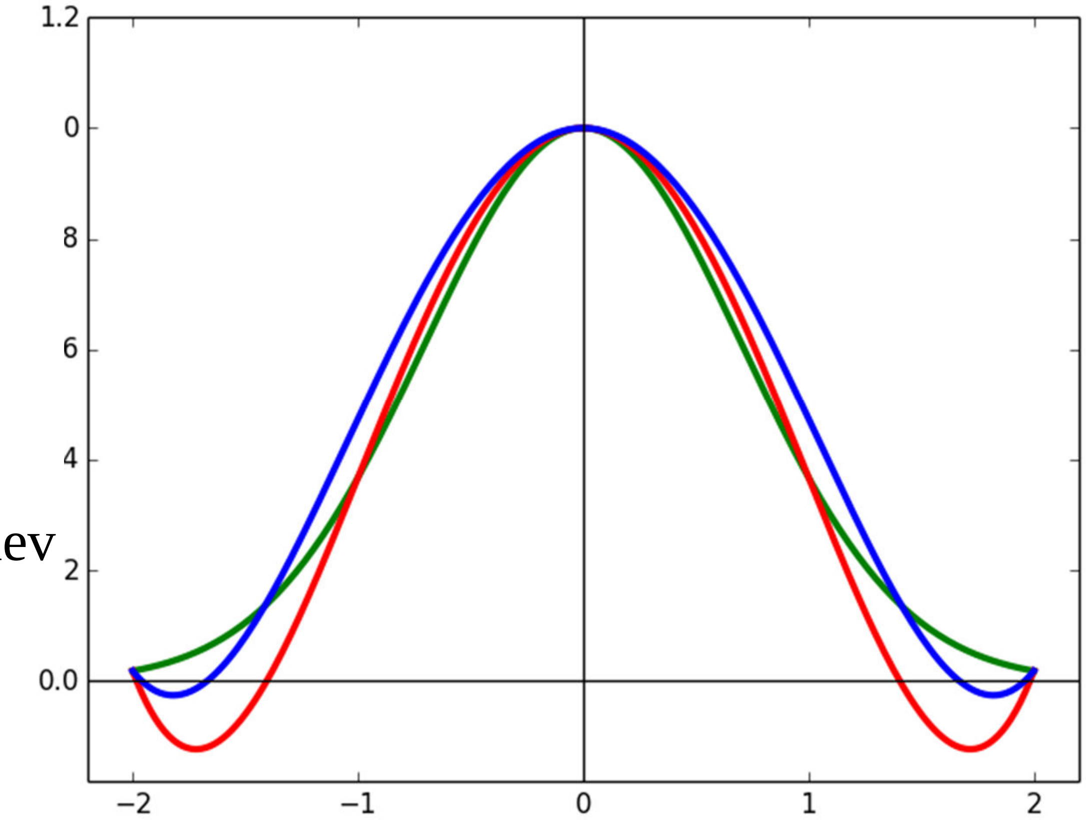


V) Intégration par arcs 1) et 2)

Autre exemple d'interpolations en utilisant 5 points, (polynôme de degré 4), soit avec un support équidistant, qualifié ici de "Lagrange", soit un support réparti selon "les abscisses de Tchebychev".

Les trois courbes :

- la fonction $f : x \mapsto e^{-x^2}$
  en vert,
- le polynôme de Lagrange
  en rouge,
- le polynôme de Tchebychev
  en bleu.



V) Intégration par arcs 1) et 2)

```
# Fonctions
f=lambda x:np.exp(-x**2)
# Lagrange,pas régulier
xmin,xmax,nbx=-2.0,2.0,5
XE=np.linspace(xmin,xmax,nbx)
YE=f(XE)
Pol1=np.polyfit(XE,YE,nbx-1)
print(' Lagrange = ' , Pol1 )
PolInt=np.polyint(Pol1)
Iregulier=np.polyval(PolInt,xmax)-np.polyval(PolInt,xmin)
print('Intégrale =',Iregulier)
```

Résultats : Lagrange =

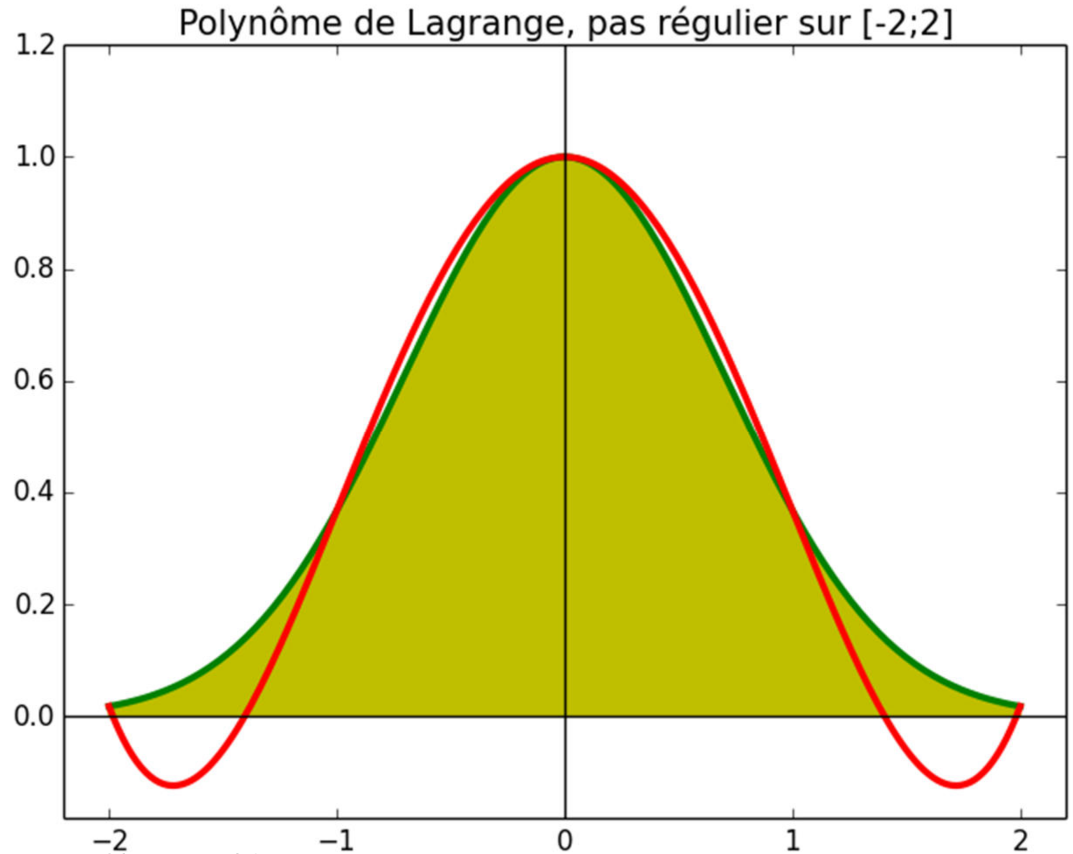
```
[ 9.34556341e-02
 -3.70074342e-17
  -6.19243626e-01
   9.27261682e-17
  1.00000000e+00]
Intégrale = 1.59114236353
```

V) Intégration par arcs 1) et 2)

Les trois courbes : la fonction $f : x \mapsto e^{-x^2}$ en vert,

le polynôme de Lagrange en rouge,

le polynôme de Tchebychev en bleu.



V) Intégration par arcs 3)

```
XE=(xmin+xmax)/2-(xmax-xmin)/
                    2*np.cos(np.pi*np.arange(nbx)/(nbx-1))
YE=f(XE)
Pol2=np.polyfit(XE,YE,nbx-1)
print(' Tchebychev = ' , Pol2 )
PolInt2=np.polyint(Pol2)
ITchebychev =np.polyval(PolInt2,xmax)-
np.polyval(PolInt2,xmin)
print('Intégrale =',ITchebychev)
print('La valeur exacte est ',
np.sqrt(np.pi))
```

Résultats : Tchebychev =
[9.34556341e-02
 -3.70074342e-17
 -6.19243626e-01
  9.27261682e-17
  1.00000000e+00]
Intégrale = 1.89359944128

$$I_{exacte} = \int_{-\infty}^{\infty} \exp\left(-x^2\right) \mathrm{d}x = \sqrt{I_0} = \sqrt{\pi}$$

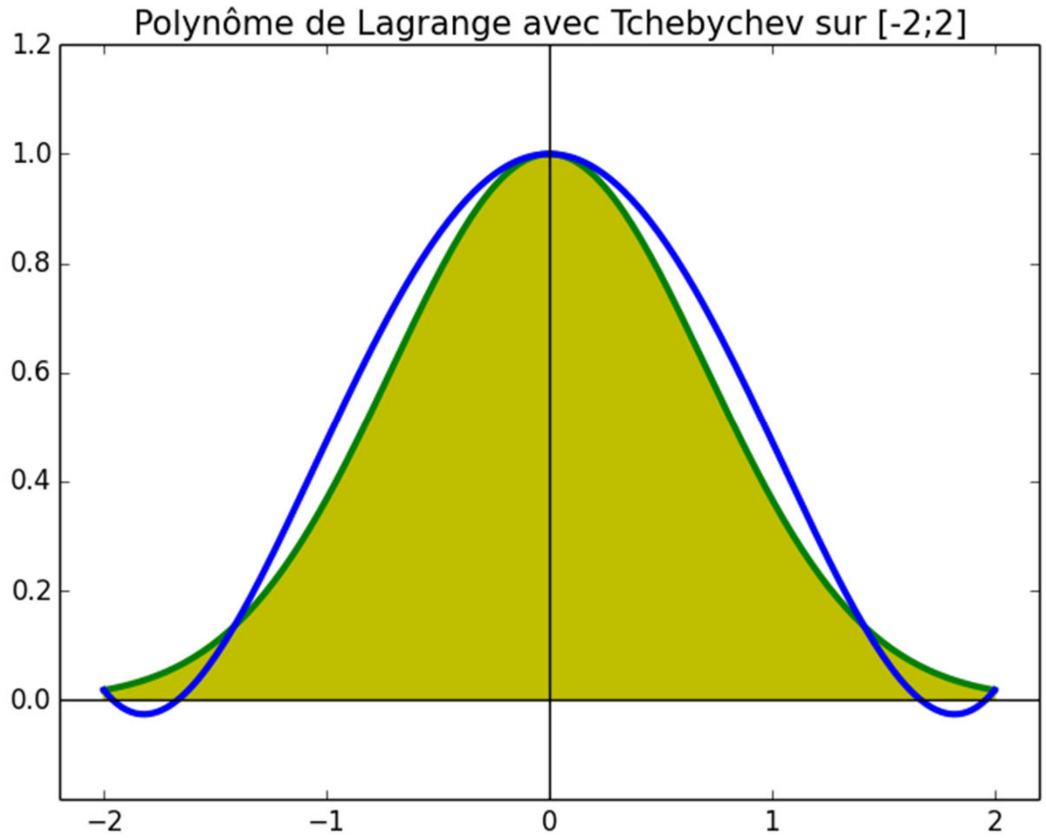
V) Intégration par arcs 3)

Les trois courbes : la fonction $f : x \mapsto e^{-x^2}$ en vert,

le polynôme de Lagrange en rouge,

le polynôme de Tchebychev en bleu.

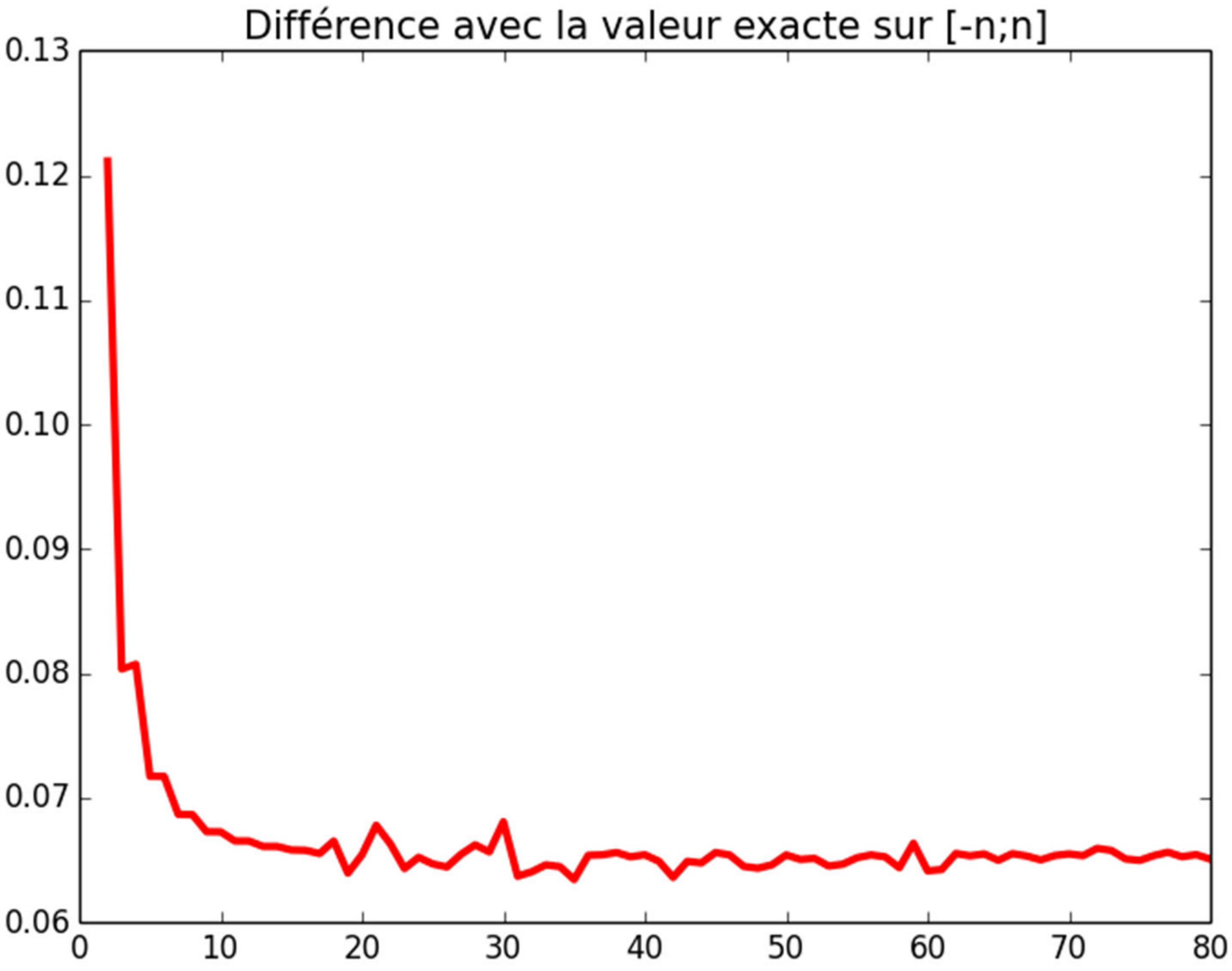
Interpolations en 5 points, équidistants (Lagrange) ou répartis selon Tchebychev.



V) Intégration par arcs 4)

```
nbxmin,nbxmax=2,51
I=[]
Xn=range(nbxmin,nbxmax)
for n in Xn:
    xmin,xmax,nbx=-n,n,2*n+1
    XE=(xmin+xmax)/2-(xmax-xmin)/
                        2*np.cos(np.pi*np.arange(nbx)/(nbx-1))
    YE=f(XE)
    Pol=np.polyfit(XE,YE,nbx-1)
    PolInt=np.polyint(Pol)
    I.append(np.polyval(PolInt,xmax)-np.polyval(PolInt,xmin))
ValeurExacte=np.sqrt(np.pi)
plt.figure(4)
plt.plot(Xn,np.array(I)-ValeurExacte,'r',linewidth=3)
plt.title('Différence avec la valeur exacte sur [-n;n]',fontsize=15)
```

V) Intégration par arcs 4)



Annexe : Méthodes de Newton-Cotes

Généralisation de Trapèzes et de Simpson, avec des polynômes de degré p :

On peut approcher l'intégrale, par la surface sous le polynôme de degré p

interpolant f aux points $\left(P_k\right)_{0\le k\le p}$ $\quad I_{N,p} = \dfrac{b-a}{p\times\beta}\left[\sum_{k=0}^{k=p}\alpha_k\, f\left(a+k\dfrac{(b-a)}{p}\right)\right]$

Formule	p	β	α_0	α_1	α_2	α_3	α_4	α_5	α_6
Trapèze	1	2	1	1					
Simpson	2	3	1	4	1				
Boole	3	8	3	9	9	3			
Villarceau	4	45	14	64	24	64	14		
Hardy	6	140	41	216	27	272	27	216	41