

**Fekih Firas**

1. **http://localhost:8888/tree/Desktop/Projet-AnalyseNmrq**

**2) On va utiliser le code Python de la méthode rectangle à gauche :**

* **Valeurs approchées :**
* pour n=5

import numpy as np

f = lambda t : np.cos(t)

a = 0

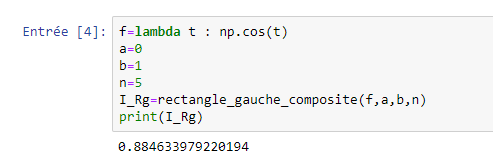
b = 1

n = 5

f1 = \_\_init\_\_(self,a,b,n,f) ;

Integ\_f1 = integrate (self, f1) ;

Print(Integ\_fcos ) ;



f = lambda t : np.sin(t)

a = 0

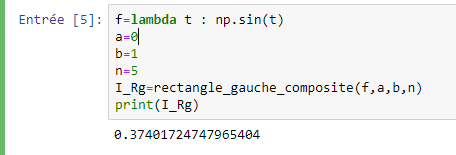
b = 1

n = 5

f2 = \_\_init\_\_(self,a,b,n,f) ;

Integ\_f2 = integrate (self, f2) ;

Print(Integ\_f2 ) ;



Import sympy as sp

t = sp.Symbol(‘t’) ;

f = lambda t : (t\*\*2-t+5)

a=0

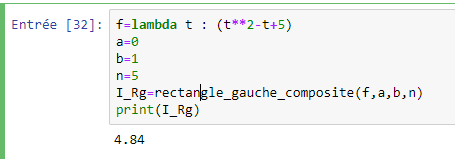
b = 1

n = 5

f3 = \_\_init\_\_(self,a,b,n,f) ;

Integ\_f3 = integrate (self, f3) ;

Print(Integ\_f3) ;



f = lambda t : 1/(1+t\*\*2)

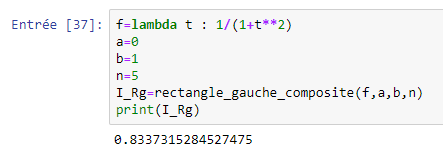
a=0

b = 1

n = 5

f4 = \_\_init\_\_(self,a,b,n,f) ;

Print(Integ\_f4) ;



* **pour n = 25 :**

import numpy as np

f = lambda t : np.cos(t)

a = 0

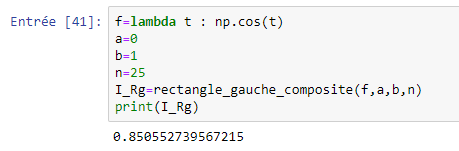
b = 1

n = 25

f1 = \_\_init\_\_(self,a,b,n,f) ;

Integ\_f1 = integrate (self, f1) ;

Print(Integ\_fcos ) ;



f = lambda t : np.sin(t)

a = 0

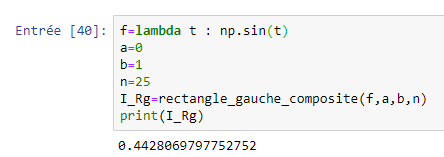
b = 1

n = 25

f2 = \_\_init\_\_(self,a,b,n,f) ;

Integ\_f2 = integrate (self, f2) ;

Print(Integ\_f2 ) ;



Import sympy as sp

t = sp.Symbol(‘t’) ;

f = lambda t : (t\*\*2-t+5)

a=0

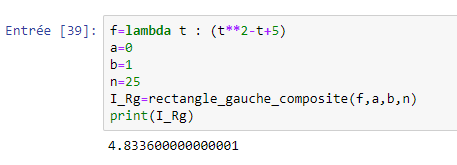
b = 1

n = 25

f3 = \_\_init\_\_(self,a,b,n,f) ;

Integ\_f3 = integrate (self, f3) ;

Print(Integ\_f3) ;



f = lambda t : 1/(1+t\*\*2)

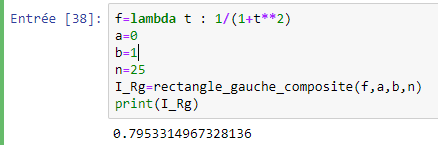
a=0

b = 1

n = 25

f4 = \_\_init\_\_(self,a,b,n,f) ;

Print(Integ\_f4) ;



* **n=50**

import numpy as np

f = lambda t : np.cos(t)

a = 0

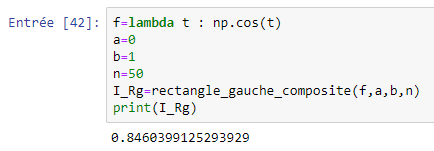
b = 1

n = 50

f1 = \_\_init\_\_(self,a,b,n,f) ;

Integ\_f1 = integrate (self, f1) ;

Print(Integ\_fcos ) ;



f = lambda t : np.sin(t)

a = 0

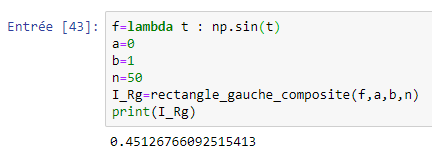
b = 1

n = 50

f2 = \_\_init\_\_(self,a,b,n,f) ;

Integ\_f2 = integrate (self, f2) ;

Print(Integ\_f2 ) ;



Import sympy as sp

t = sp.Symbol(‘t’) ;

f = lambda t : (t\*\*2-t+5)

a=0

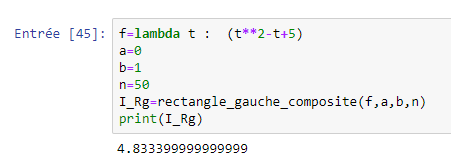
b = 1

n = 50

f3 = \_\_init\_\_(self,a,b,n,f) ;

Integ\_f3 = integrate (self, f3) ;

Print(Integ\_f3) ;



f = lambda t : 1/(1+t\*\*2)

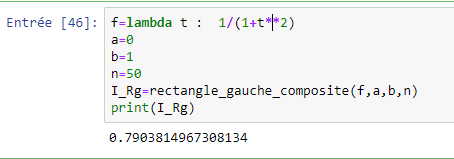
a=0

b = 1

n = 50

f4 = \_\_init\_\_(self,a,b,n,f) ;

Print(Integ\_f4) ;



* **pour n= 100 :**

import numpy as np

f = lambda t : np.cos(t)

a = 0

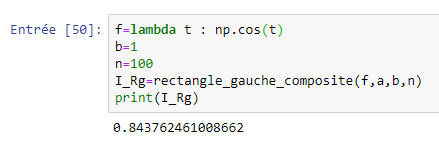
b = 1

n = 100

f1 = \_\_init\_\_(self,a,b,n,f) ;

Integ\_f1 = integrate (self, f1) ;

Print(Integ\_fcos ) ;



f = lambda t : np.sin(t)

a = 0

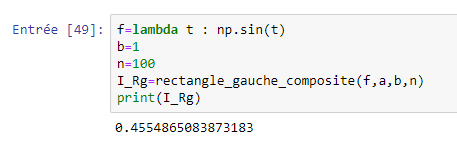
b = 1

n = 100

f2 = \_\_init\_\_(self,a,b,n,f) ;

Integ\_f2 = integrate (self, f2) ;

Print(Integ\_f2 ) ;



Import sympy as sp

t = sp.Symbol(‘t’) ;

f = lambda t : (t\*\*2-t+5)

a=0

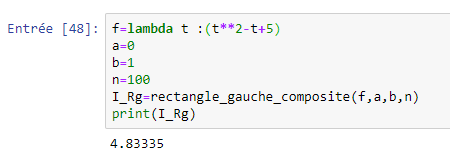
b = 1

n = 100

f3 = \_\_init\_\_(self,a,b,n,f) ;

Integ\_f3 = integrate (self, f3) ;

Print(Integ\_f3) ;



f = lambda t : 1/(1+t\*\*2)

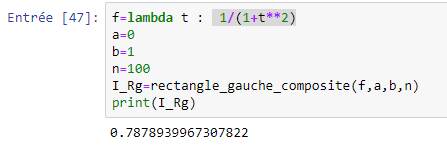
a=0

b = 1

n = 100

f4 = \_\_init\_\_(self,a,b,n,f) ;

Print(Integ\_f4) ;



**3) Valeurs exactes des**

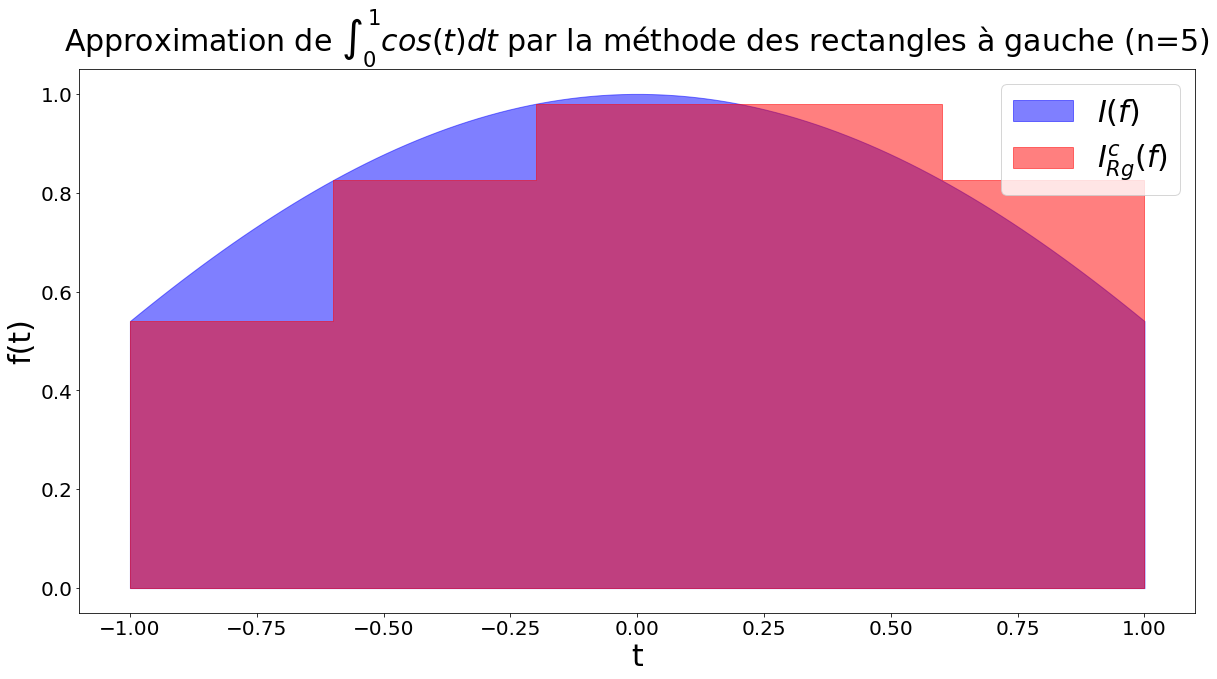
sympy.evalf() : Renvoie l'expression mathématique évaluée

Import numpy as np

import sympy as sp

t = sp.Symbol(‘t’) ;

I1 = sp.integrate(np.cos(t),(t,0,1)).evalf() ;

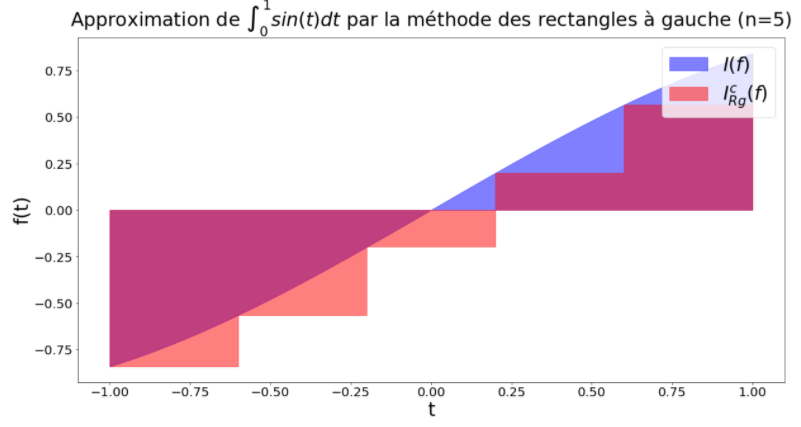


Import numpy as np

import sympy as sp

t = sp.Symbol(‘t’) ;

I2 = sp.integrate(np.sin(t),(t,0,1)).evalf() ;

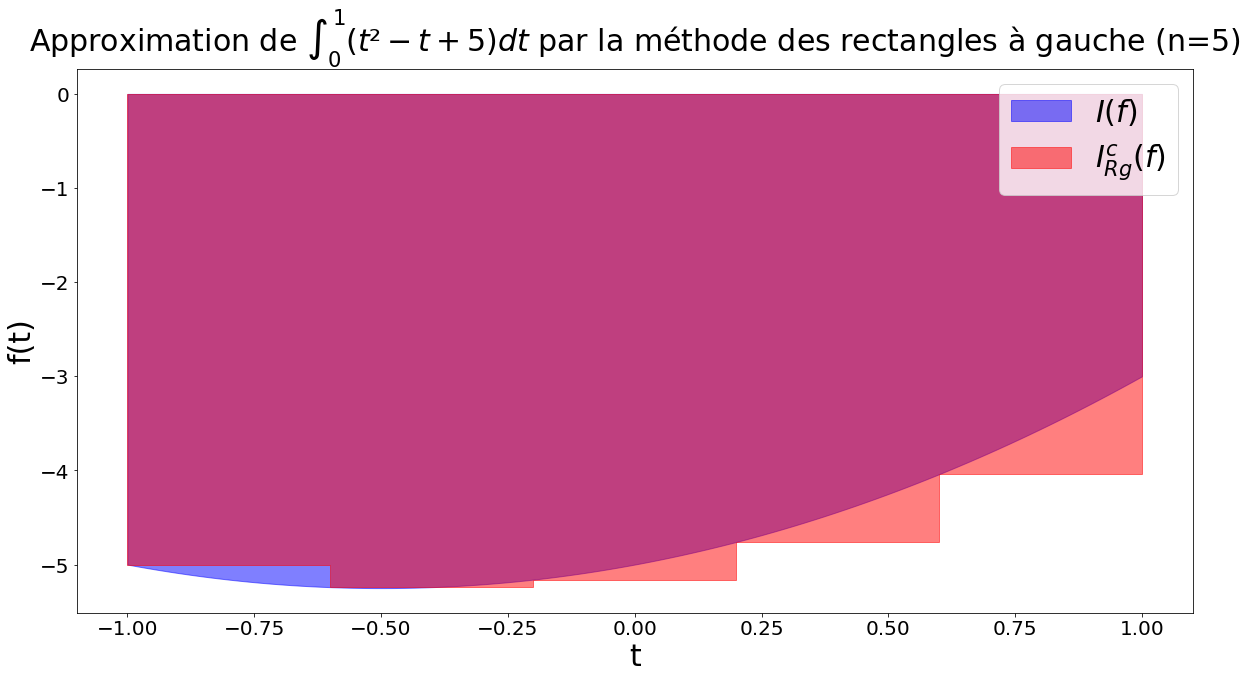


Import numpy as np

import sympy as sp

t = sp.Symbol(‘t’) ;

I3 = sp.integrate(t\*\*2-t+5,(t,0,1)).evalf() ;

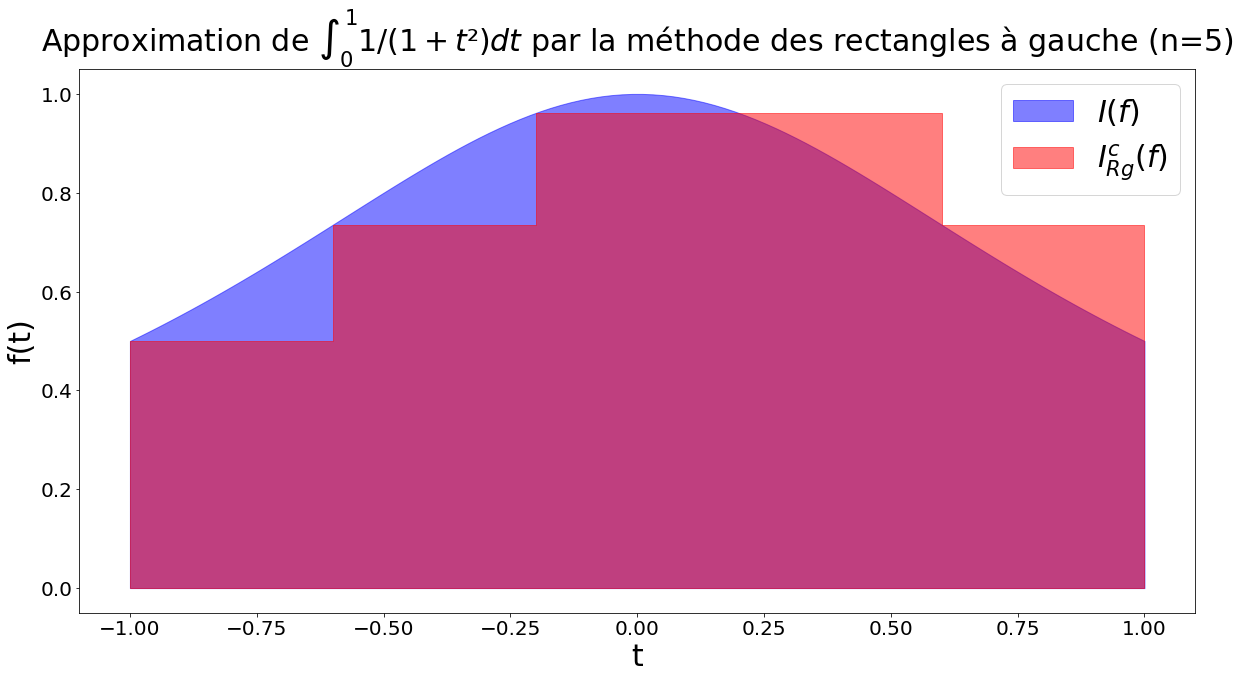


Import numpy as np

import sympy as sp

t = sp.Symbol(‘t’) ;

I4 = sp.integrate(1/(1+t\*\*2),(t,0,1)).evalf() ;



**4) Erreurs numériques**

N = 5

Print('n=',5)

Print(" Erreur commise par f1 : " , abs (Integ\_f1 – I1)) ;

Print(" Erreur commise par f2 : " , abs (Integ\_f2 – I2)) ;

Print(" Erreur commise par f3 : " , abs (Integ\_f3 – I3)) ;

Print(" Erreur commise par f4 : " , abs (Integ\_f4 – I4)) ;

**5)**

Quand on augmente le nombre des sous – intervalles (n), l’erreur tend à diminuer.