Algorithmes d'approximation des intégrales

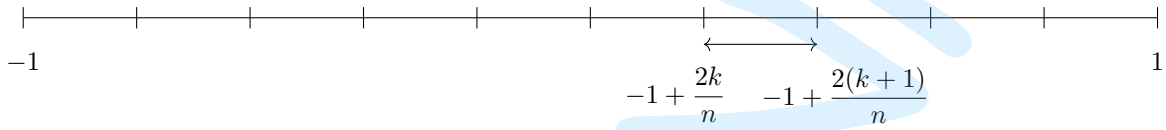
Nous considérons la fonction définie sur $\mathbb{R}$ par $f(x) = \dfrac{8}{x^2+2x+5}$.

Nous souhaitons calculer une valeur approchée de $\displaystyle\int_{-1}^{1} f(x)\,\mathrm{d}x$.

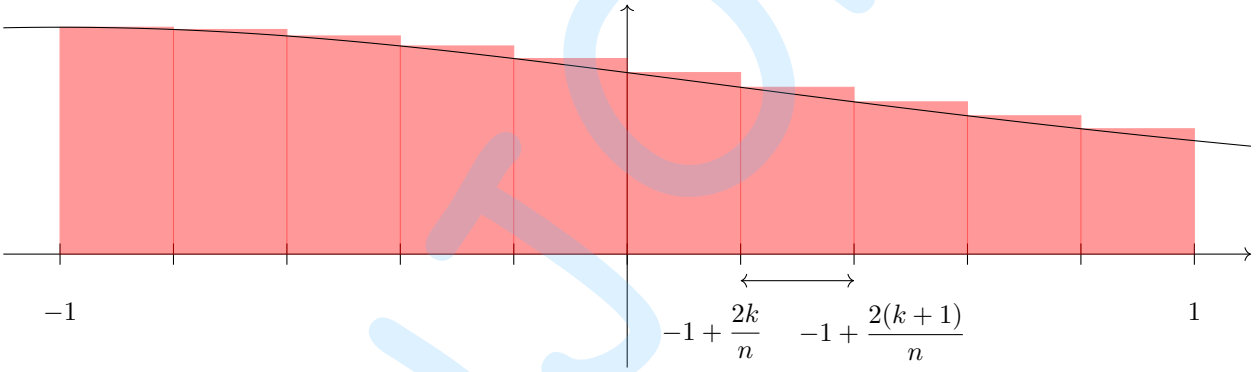
I Méthode des rectangles à gauche

Méthode

- Découper l'intervalle $[a\ ;\ b]$ en n intervalles de même amplitude $\dfrac{2}{n}$: $-1\ ;\ -1+\dfrac{2}{n}\ ;\ -1+\dfrac{4}{n}\ ;\ -1+\dfrac{6}{n}\ ;\ \cdots$



- Faire la somme des rectangles de largeur $\dfrac{2}{n}$ et de hauteur $f\left(-1+\dfrac{2k}{n}\right)$.



Exercice 1

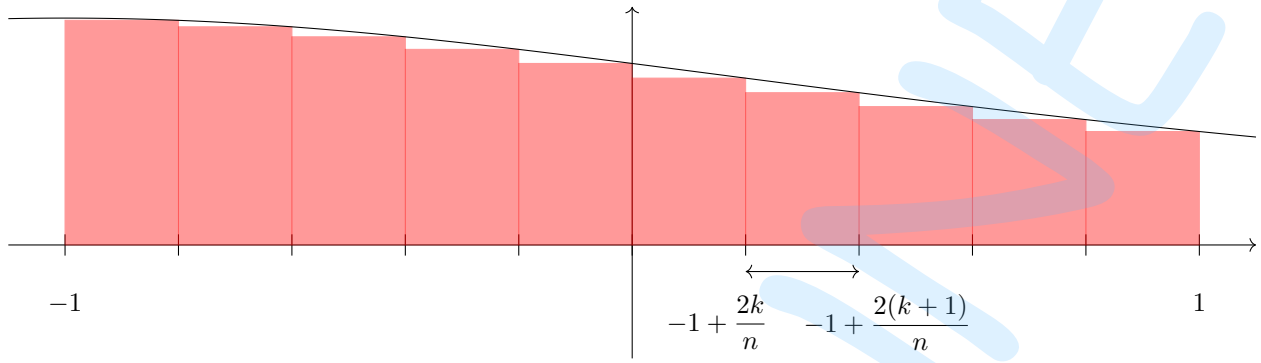
Mettre en œuvre le programme suivant écrit en Python ou l'adapter sous TI :

```
def f(x):
    return 8/(x**2+2*x+5)

def rectangles_gauche(n):
    s=0
    x=-1
    for i in range(n):
        s=s+2/n*f(x)
        x=x+2/n
    return s

print(rectangles_gauche(2))
print(rectangles_gauche(10))
print(rectangles_gauche(100))
```

II Méthode des rectangles à droite

Méthode

- Découper l'intervalle $[a\ ;\ b]$ en n intervalles de même amplitude $\frac{2}{n}$: $-1\ ;\ -1+\frac{2}{n}\ ;\ -1+\frac{4}{n}\ ;\ -1+\frac{6}{n}\ ;\ \cdots$
- Faire la somme des rectangles de largeur $\frac{2}{n}$ et de hauteur $f\left(-1+\frac{2(k+1)}{n}\right)$.



Exercice 2

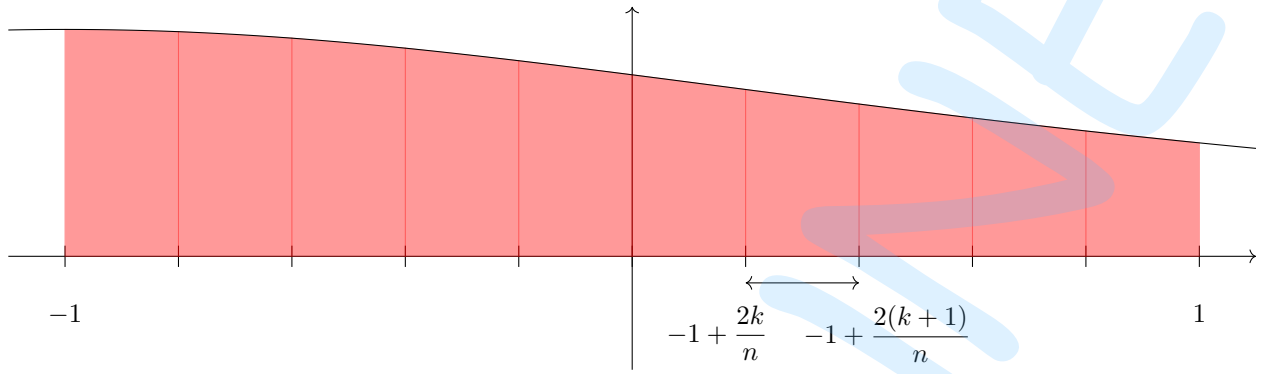
Mettre en œuvre le programme suivant écrit en Python ou l'adapter sous TI :

```
def f(x):
    return 8/(x**2+2*x+5)

def rectangles_droite(n):
    s=0
    x=-1
    for i in range(n):
        x=x+2/n
        s=s+2/n*f(x)
    return s

print(rectangles_droite(2))
print(rectangles_droite(10))
print(rectangles_droite(100))
```

III Méthode des trapèzes

Méthode

- Découper l'intervalle $[a\ ;\ b]$ en n intervalles de même amplitude $\frac{2}{n}$: $-1\ ;\ -1+\frac{2}{n}\ ;\ -1+\frac{4}{n}\ ;\ -1+\frac{6}{n}\ ;\ \cdots$
- Faire la somme des trapèzes de hauteur $\frac{2}{n}$ et de bases $f\left(-1+\frac{2k}{n}\right)$ et $f\left(-1+\frac{2(k+1)}{n}\right)$.



Exercice 3

Mettre en œuvre les programmes suivants écrits en Python ou les adapter sous TI :

```
def f(x):
    return 8/(x**2+2*x+5)

def trapezes(n):
    s=0
    x=-1
    for i in range(n):
        s=s+2/n*(f(x)+f(x+2/n))/2
        x=x+2/n
    return s

print(trapezes(2))
print(trapezes(10))
print(trapezes(100))
```

```
def f(x):
    return 8/(x**2+2*x+5)

def trapezes(n):
    s=0
    h=2/n
    x=-1
    for i in range(n-1):
        x=x+h
        s+=f(x)
    s=s+(f(-1)+f(1))/2
    s=s*h
    return s

print(trapezes(2))
print(trapezes(10))
print(trapezes(100))
```

Que peut-on remarquer ?

Comparer l'efficacité des méthodes précédentes sachant que : $\int_{-1}^{1} f(x)\,\mathrm{d}x = \pi$.

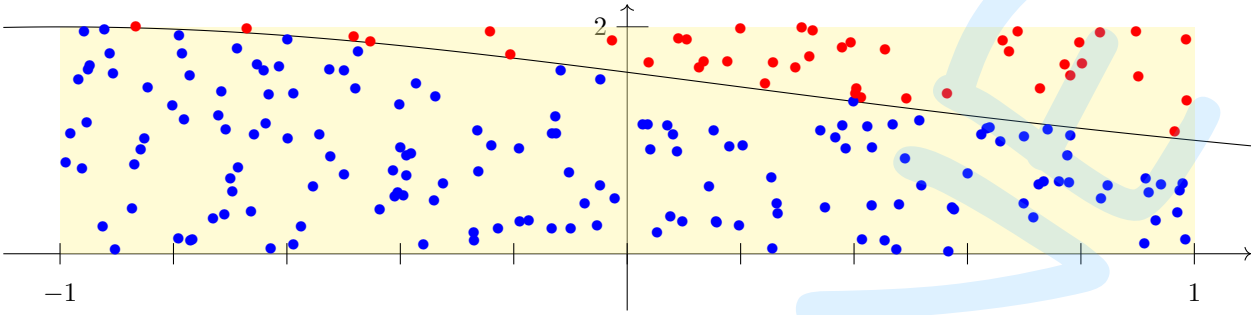
Remarque

Votre calculatrice possède une fonction $\boxed{\int f(x)\mathrm{d}x}$ permettant de donner une valeur approchée d'une intégrale.

IV Méthode de Monte-Carlo

Méthode

- Prendre n points au hasard dans le rectangle $[-1\ ;\ 1] \times [0\ ;\ 2]$.



- Déterminer la fréquence du nombre de points sous la courbe représentative de f.
- Multiplier cette fréquence par l'aire du rectangle $[-1\ ;\ 1] \times [0\ ;\ 2]$ c'est-à-dire par 4.

Exercice 4

Mettre en œuvre le programme suivant écrit en Python ou l'adapter sous TI :

```
from random import *

def f(x):
    return 8/(x**2+2*x+5)

def montecarlo(n):
    effectif=0
    for i in range(n):
        x=-1+2*random()
        y=2*random()
        if y<f(x):
            effectif=effectif+1
    frequence=effectif/n
    return frequence*4

print(montecarlo(2))
print(montecarlo(10))
print(montecarlo(100))
print(montecarlo(1000))
print(montecarlo(10000))
```

Que pensez-vous de son efficacité ?