Informatique

**Intégration numérique**

***Cours***

[Intégration numérique 3](#_Toc67919509)

[1.I. Contexte 3](#_Toc67919510)

[1.II. Valeur à gauche – Méthode des rectangles 4](#_Toc67919511)

[1.III. Valeur à droite - Méthode des rectangles 4](#_Toc67919512)

[1.IV. Valeur centrée – Méthode des trapèzes 5](#_Toc67919513)

[1.IV.1 Principe 5](#_Toc67919514)

[1.IV.2 Remarques 6](#_Toc67919515)

[1.V. Remarque sur les trois méthodes 6](#_Toc67919516)

[1.VI. Fonction python préprogrammée 6](#_Toc67919517)

###### Intégration numérique

# Contexte

Soit le signal échantillonné suivant contenant valeurs à partir du temps :

On veut :

Appelons et les temps de part et d’autre de chaque intervalle.

On va sommer les aires rectangles sur chaque intervalle , de largeur et de hauteur, soit :

* Valeur à gauche :
* Valeur à droite :
* Valeur centrée :

# Valeur à gauche – Méthode des rectangles

Attention à ne bien prendre que valeurs !

On voit qu’il y a sous-estimation de la courbe lorsqu’elle est croissante et surestimation lorsqu’elle est décroissante.

# Valeur à droite - Méthode des rectangles

Attention à ne bien prendre que valeurs !

On voit qu’il y a surestimation de la courbe lorsqu’elle est croissante et sous-estimation lorsqu’elle est décroissante.

# Valeur centrée – Méthode des trapèzes

## Principe

Le calcul d’aires en tenant compte de la valeur centrée sur le segment de largeur revient calculer les aires de trapèzes, d’où le nom de méthode des trapèzes :

Les deux surfaces ci-dessus ont des aires égales.

## Remarques

Attention à ne bien prendre que valeurs !

On voit que cette méthode compense à peu près la surestimation et la sous-estimation de sur chacun des intervalles de largeur . Elle sera donc privilégiée. Elle sous estimera ou surestimera les intégrales en fonction de la courbure (concavité/convexité) de la courbe intégrée.

On privilégiera le calcul avec les valeurs centrées plutôt que de faire la somme des aires des rectangles et des triangles supérieurs…

Lorsque l’on a bien des intervalles réguliers de largeur , on a aussi :

Soit finalement :

# Remarque sur les trois méthodes

Plus sera petit, soit plus il y aura de points, plus les résultats seront proches de la réalité. Mais évidemment, la complexité étant en , le temps de calcul augmentera avec la précision :

# Fonction python préprogrammée

|  |
| --- |
| **from** scipy**.**integrate **import** quad  **def** f**(**x**):**  **return** x  a **=** 1  b **=** 2  Res**,**Precision **=** quad**(**f**,**a**,**b**)**  **print(**Res**)** |

Remarque : Infini possible via « numpy**.**inf ».