Fiche de révision — Méthode des rectangles

Objectif. Approcher numériquement une intégrale $\displaystyle\int_a^b f(t)\,dt$ lorsqu'on ne sait pas *(ou ne veut pas)* calculer une primitive de f.

Cadre. f est continue sur $[a,b]$. On découpe $[a,b]$ en n sous-intervalles de même longueur à l'aide de la *subdivision régulière*

$$x_k = a + k\,h, \qquad h = \frac{b-a}{n}, \qquad k \in [\![0, n]\!].$$

Le *pas* h est la largeur commune des sous-intervalles $[x_k, x_{k+1}]$.

Les trois variantes.

Méthode. On approche l'aire sous la courbe par une somme d'aires de rectangles :

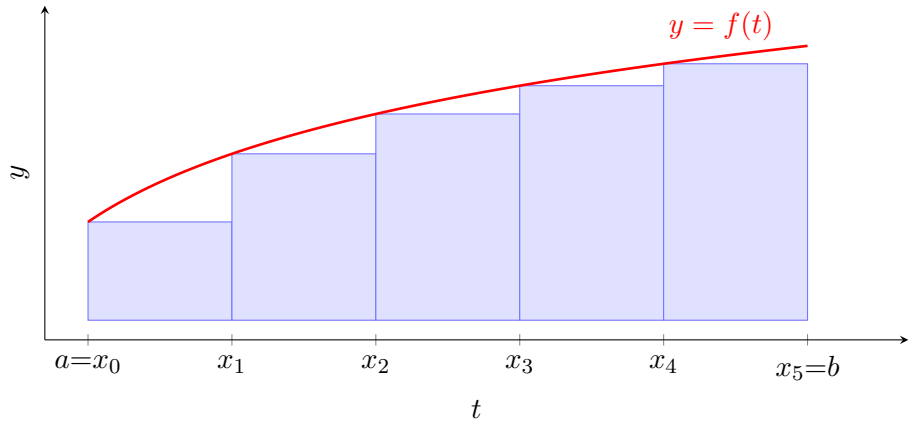
$$\underbrace{R_n^g = h\sum_{k=0}^{n-1} f(x_k)}_{\text{rectangles à gauche}} \qquad \underbrace{R_n^d = h\sum_{k=1}^{n} f(x_k)}_{\text{rectangles à droite}} \qquad \underbrace{R_n^m = h\sum_{k=0}^{n-1} f\left(\frac{x_k + x_{k+1}}{2}\right)}_{\text{point milieu}}$$

Chacune de ces sommes est une **somme de Riemann** de f sur $[a,b]$.

Comme f est continue, ses sommes de Riemann convergent vers l'intégrale lorsque le pas tend vers 0 : c'est ce qui justifie la méthode.

$$\lim_{n\to+\infty} h\sum_{k=0}^{n-1} f(x_k) = \int_a^b f(t)\,dt$$

Interprétation graphique (rectangles à gauche, $n = 5$) pour une fonction positive.



On approche l'aire sous la courbe par la somme des aires des rectangles.

En Python.

```
def rect_gauche(f, a, b, n):
    h = (b - a)/n                          # pas de la subdivision
    S = 0
    for k in range(n):                     # k = 0, 1, ..., n-1
        S = S + f(a + k*h)                 # hauteur au bord gauche
    return h*S
```

Revoir la feuille_ info_ 2.

Fiche de révision — Méthode d'Euler

Objectif. Approcher la solution du *problème de Cauchy*

$$y'(t) = F\big(t, y(t)\big), \qquad y(t_0) = y_0,$$

sur un intervalle $[t_0, t_f]$, lorsqu'on ne sait pas résoudre l'équation explicitement (typiquement les équations autonomes : modèles logistique, de Gompertz… *(voir annexe D du cours de maths)*).

Idée : suivre la tangente. Pour un petit pas h, la formule de Taylor donne

$$y(t+h) = y(t) + h\,y'(t) + o(h) = y(t) + h\,F\big(t, y(t)\big) + o(h).$$

On **néglige** le reste $o(h)$: on remplace localement la courbe par sa tangente.

Le schéma.

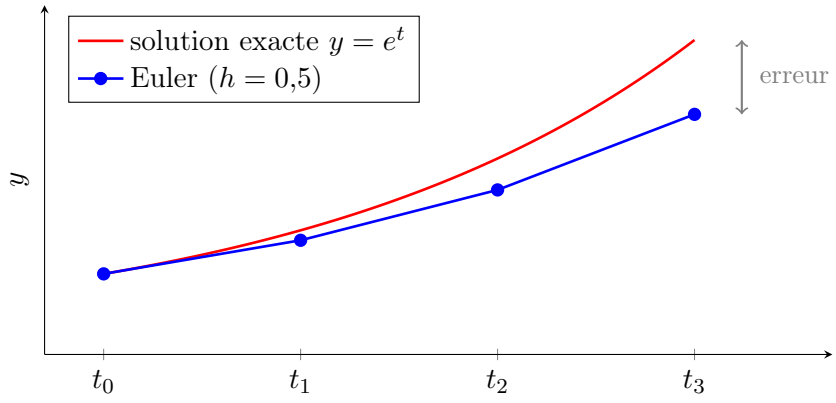
Méthode. Avec la subdivision régulière $t_k = t_0 + k\,h$ et $h = \frac{t_f - t_0}{n}$, on construit une suite (y_k) par la récurrence

$$y_{k+1} = y_k + h\,F(t_k,\, y_k), \qquad k \in [\![0, n-1]\!].$$

Le terme y_k est une **approximation** de la vraie valeur $y(t_k)$.

Interprétation graphique. On part de (t_0, y_0) et, à chaque pas, on avance en ligne droite selon la pente $F(t_k, y_k)$ pendant une durée h : la solution approchée est une **ligne polygonale**.

Ci-dessous sur $y' = y,\ y(0) = 1$ (solution exacte $y = e^t$), avec $h = 0{,}5$.



Extensions au programme. *(Voir feuille 13 et 15 d'informatique)*

- *Systèmes* : si $Y(t) = (x(t), y(t))$, la récurrence vectorielle $Y_{k+1} = Y_k + h\,F(t_k, Y_k)$ s'écrit composante par composante.
- *Ordre* 2 : une équation $y'' = g(t, y, y')$ se ramène à un système d'ordre 1 en posant $z = y'$, puis on applique Euler au couple (y, z).

En Python. *(Une version différente de ce qu'on a fait dans la feuille 15)*

```
    h = (tf - t0)/n                    # pas de la subdivision
    t, y = t0, y0
    T, Y = [t0], [y0]
    for k in range(n):                 # n pas
        y = y + h*F(t, y)              # un pas d'Euler
        t = t + h                      # Attention à l'ordre de ces deux lignes.
        T.append(t)
        Y.append(y)
    plt.plot(T, Y)
    plt.show()
```

Revoir les feuille_info_ 13 et 15.