Chapitre 11
Analyse numérique

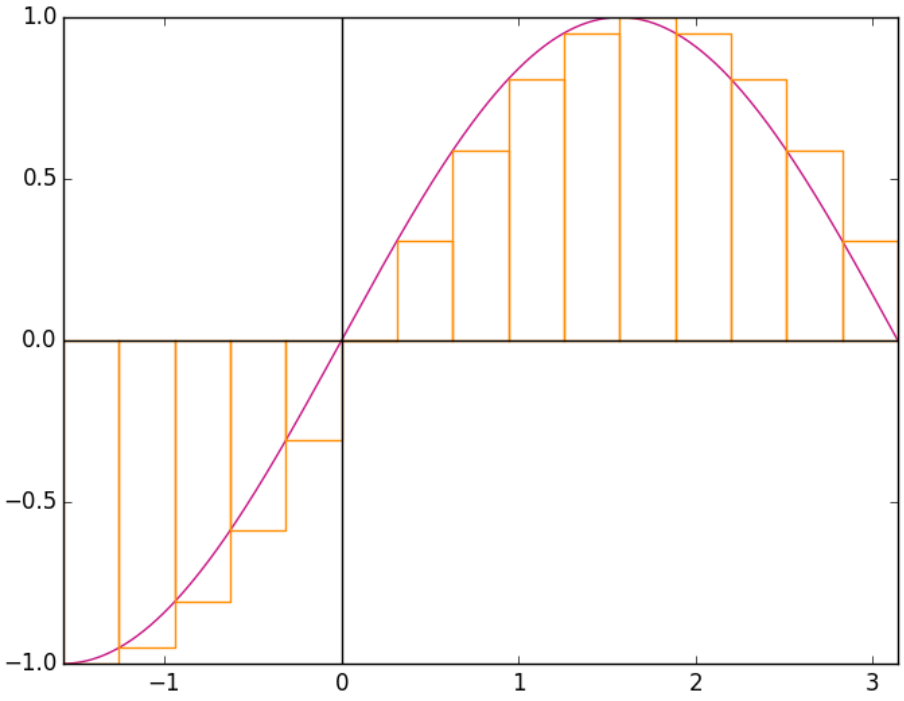
En sciences expérimentales, l'étude de la plupart des phénomènes mène à des calculs que l'on ne sait pas faire ou des équations que l'on ne sait pas résoudre de manière exacte. On se tourne alors vers un calcul ou une résolution approchée, à l'aide d'une méthode d'analyse numérique.

1 Calcul approché d'intégrales : méthode des rectangles

Soit $f : [a; b] \longrightarrow \mathbb{R}$ une fonction continue.

Dans cette partie, on souhaite calculer une valeur approchée de l'intégrale $\displaystyle\int_a^b f(x)\mathrm{d}x$.

Pour ce faire, on applique la **méthode des rectangles** : on approche l'aire sous la courbe de la fonction f entre les points a et b par une somme d'aires de rectangles (de largeur petite), comme sur la figure ci-dessous.



Cette approximation se justifie par le théorème des sommes de Riemann, vu en cours de maths.

Théorème. (Riemann)
Soit f une fonction continue sur le segment $[a;b]$. Alors

$$\lim_{n\to+\infty} \frac{b-a}{n}\sum_{k=0}^{n-1} f\left(a+k\frac{b-a}{n}\right) = \int_a^b f(x)\mathrm{d}x.$$

Plus l'entier n est grand (c'est-à-dire plus le pas de la subdivision est petit), plus le réel $\frac{b-a}{n}\sum_{k=0}^{n-1} f\left(a+k\frac{b-a}{n}\right)$ fournit une bonne approximation de l'intégrale $\int_a^b f(x)\mathrm{d}x$.

```
def rectangles(f, a, b, n):
    """
    Entrées :
        - une fonction f continue ;
        - deux flottants a et b tels que a < b ;
        - un entier naturel n non nul.
    Sortie : la n-ième somme de Riemann de f sur [a ; b],
             qui approche l'intégrale de f sur [a ; b].
    """
    h = (b - a) / n # h est le pas de la subdivision.
    somme = 0
    for k in range(n): # L'indice k va de 0 à n - 1.
        somme += f(a + k * h)
    return h * somme
```

2 Résolution approchée d'équations : algorithme de dichotomie

Dans cette section, on s'intéresse à la recherche d'une solution approchée d'une équation (que l'on ne saurait pas résoudre mathématiquement de manière exacte).

Il est toujours possible, en passant toute l'équation d'un seul côté de l'égalité, de se ramener à une équation de la forme

$$f(x) = 0,$$

d'inconnue x réelle, où f est une fonction.

La résolution se ramène donc à la recherche de **zéros d'une fonction**, c'est-à-dire des points où la fonction s'annule.

Soit $f\colon [a;b] \longrightarrow \mathbb{R}$ une fonction continue telle que $f(a)$ et $f(b)$ soient de signes opposés.

Le théorème des valeurs intermédiaires assure alors que l'équation $f(x) = 0$ admet au moins une solution dans l'intervalle $[a;b]$.

L'**algorithme de dichotomie**, que l'on va voir en cours de maths, en recherche une solution approchée de la manière suivante.

- On considère le milieu c de l'intervalle $[a;b]$.
  - ⋆ Si $f(a)$ et $f(c)$ sont de signes opposés, alors on restreint l'intervalle dans lequel on cherche une solution à l'intervalle $[a;c]$.
  - ⋆ Sinon, $f(c)$ et $f(b)$ sont alors de signes opposés et on restreint l'intervalle à $[c;b]$.

  Dans les deux cas, l'existence d'une solution dans le nouvel intervalle est garantie par le théorème des valeurs intermédiaires.

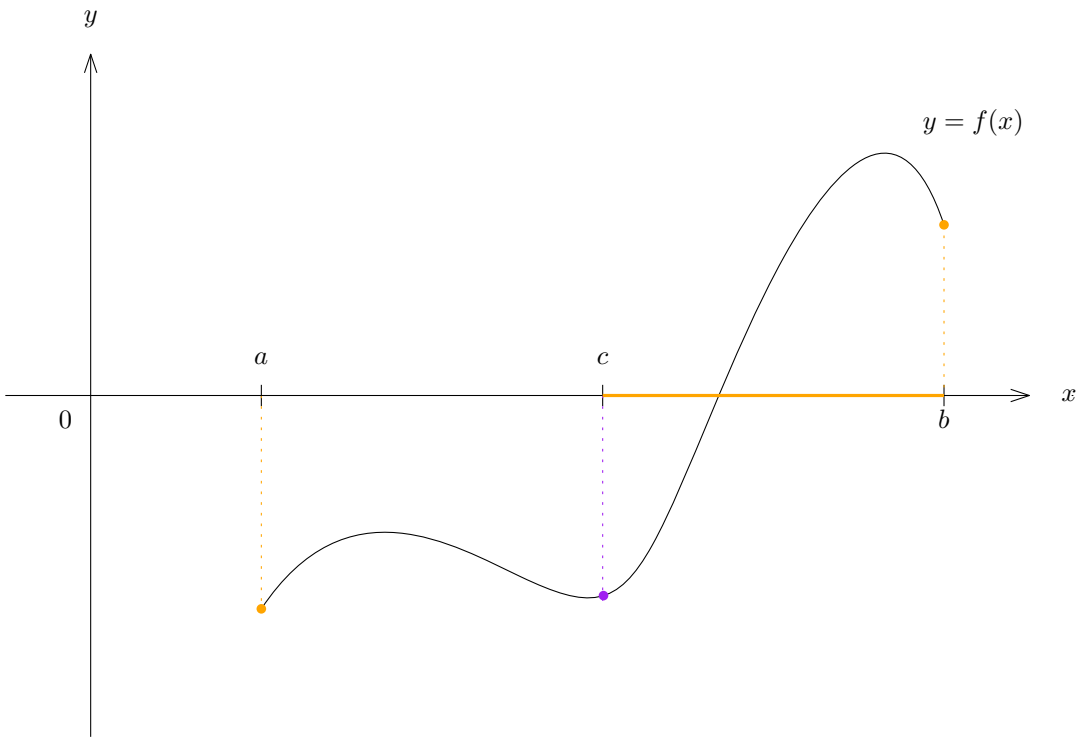


FIGURE 1 – Première étape de dichotomie

- On recommence dans ce petit intervalle (voir la figure 2).

  On définit ainsi des suites $(a_n)_{n\in\mathbb{N}}$, $(b_n)_{n\in\mathbb{N}}$ et $(c_n)_{n\in\mathbb{N}}$ telles que : $\begin{cases} a_0 = a \\ b_0 = b \\ c_0 = \dfrac{a+b}{2}, \end{cases}$

et pour tout $n \in \mathbb{N}$, si $f(a_n)$ et $f(c_n)$ sont de signes opposés, alors

$$\begin{cases} a_{n+1} = a_n \\ b_{n+1} = c_n \\ c_{n+1} = \dfrac{a_{n+1} + b_{n+1}}{2}, \end{cases}$$

et sinon,

$$\begin{cases} a_{n+1} = c_n \\ b_{n+1} = b_n \\ c_{n+1} = \dfrac{a_{n+1} + b_{n+1}}{2}. \end{cases}$$

À l'étape n, on recherche une solution de l'équation $f(x) = 0$ dans l'intervalle $[a_n; b_n]$.

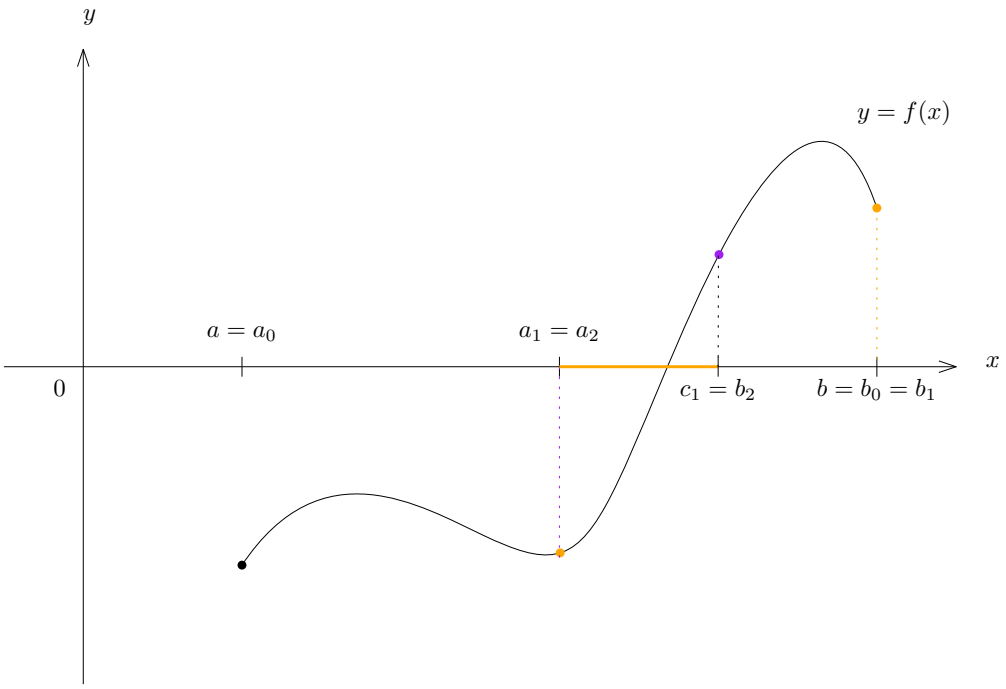
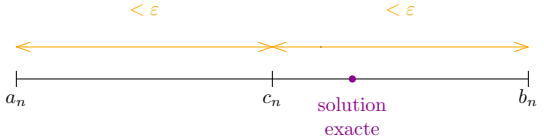


FIGURE 2 – Deuxième étape de dichotomie

- L'algorithme s'arrête lorsque la taille de l'intervalle $[a_n; b_n]$ est suffisamment petite. Plus précisément, pour obtenir une solution à une précision $\varepsilon > 0$ donnée près, on s'arrête lorsque la taille de l'intervalle $[a_n; b_n]$ est inférieure à 2ε, c'est-à-dire lorsque

$$b_n - a_n < 2\varepsilon.$$

Le milieu c_n de l'intervalle diffère alors d'au plus ε d'une solution exacte et fournira notre solution approchée à ε près.



```
def dichotomie(f, a, b, epsilon):
    """
    Entrées :
        - une fonction f ;
        - deux flottants a et b tels que
          f(a) et f(b) soient de signes opposés ;
        - une précision epsilon strictement positive.
    Sortie : valeur approchée à epsilon près d'une solution
             de l'équation f(x) = 0 sur [a ; b].
    """
    while b - a >= 2 * epsilon:
        c = (a + b) / 2
        if f(a) * f(c) <= 0:
        # Si f(a) et f(c) sont de signes opposés,
        # on recherche une solution dans [a ; c].
            b = c
        else:
        # Sinon, f(c) et f(b) sont de signes opposés,
        # on recherche alors une solution dans [c ; b].
            a = c
    return (a + b) / 2
```

Testons le programme sur l'équation $x^2 = 1$, d'inconnue x réelle.

```
def fonction(x):
    return x**2 - 1

dichotomie(fonction, 0.6, 4.1, 0.001)
# Solution approchée à un millième près.
dichotomie(fonction, -9.3, 4.1, 0.001)
# Solution approchée à un millième près (mais pas la même qu'avant).
```

3 Résolution approchée d'équations différentielles : méthode d'Euler

On s'intéresse à présent à la résolution de l'équation différentielle non (nécessairement) linéaire suivante, avec condition initiale :

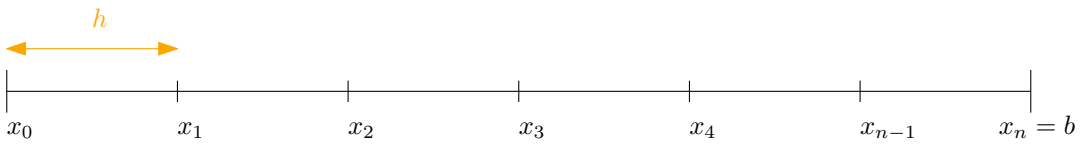
$$\begin{cases} \forall x \in [x_0; b], y'(x) = F(y(x)) \\ y(x_0) = y_0, \end{cases}$$

où I est un intervalle, x_0, b sont des réels, $y_0 \in I$, et $F : I \longrightarrow \mathbb{R}$ une fonction.

On supposera que l'équation différentielle avec condition initiale admet une unique solution (ce qui arrive lorsque la fonction F est régulière).

Appelons y cette unique solution. On cherche alors à approcher la solution y sur l'intervalle $[x_0; b]$.

Pour cela, on subdivise le segment $[x_0; b]$ en n petits sous-segments $[x_k; x_{k+1}]$ de longueur $h = \dfrac{b - x_0}{n}$ en posant, pour tout $k \in [\![0; n]\!]$, $x_k = x_0 + kh$.



On cherche alors à donner une valeur approchée de la solution y aux points de la subdivision, c'est-à-dire à donner des valeurs y_0, y_1, ..., y_n telles que pour tout $k \in [\![0; n]\!]$, le nombre y_k soit une valeur approchée de $y(x_k)$.

Avec ces valeurs approchées, on pourra alors tracer le graphe de la solution approchée sur $[x_0; b]$.

La méthode de résolution approchée que nous allons employer est la **méthode d'Euler**.

- La condition initiale $y(x_0) = y_0$ est donnée, donc on dispose de la première valeur y_0 :

$$y(x_0) = y_0.$$

- Pour approcher $y(x_1)$, on réalise l'approximation suivante de la dérivée $y'(x_0)$ (qui est bonne lorsque le pas h de la subdivision est petit) :

$$y'(x_0) \simeq \frac{y(x_0 + h) - y(x_0)}{h} = \frac{y(x_1) - y_0}{h}.$$

  Cela fournit l'approximation suivante de $y(x_1)$:

$$y(x_1) \simeq y_0 + hy'(x_0).$$

  Or la fonction y est solution de (E), donc on a

$$y'(x_0) = F(y(x_0)) = F(y_0),$$

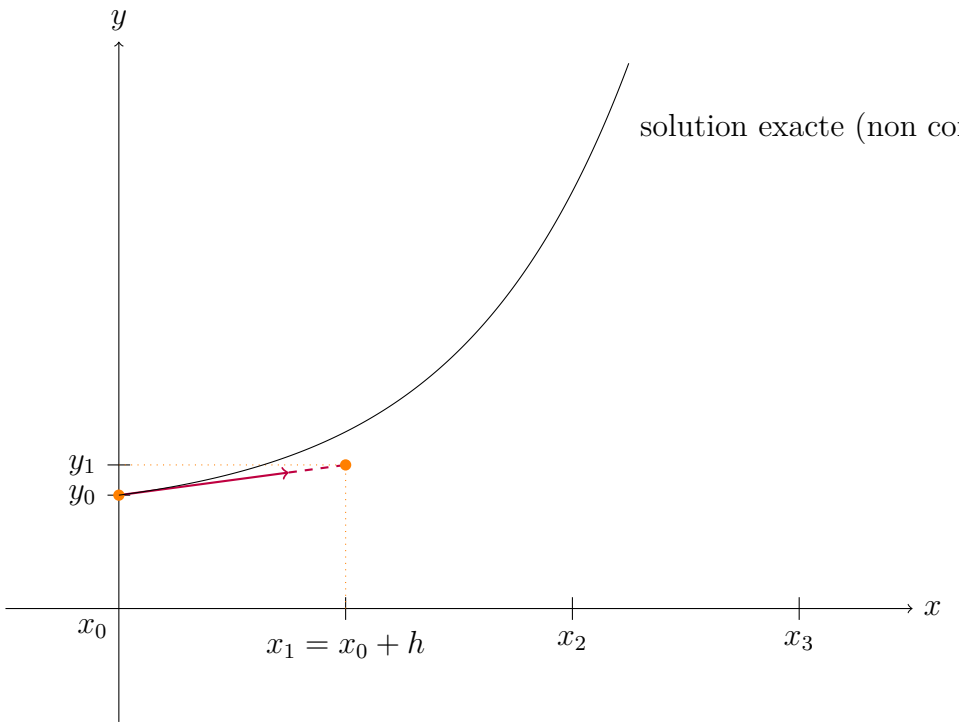
  donc on obtient l'approximation suivante de $y(x_1)$:

$$y(x_1) \simeq y_0 + hF(y_0).$$

  On pose alors

$$y_1 = y_0 + hF(y_0).$$

  Autrement dit, cette approximation consiste à approcher sur le segment $[x_0; x_1]$ la fonction y par sa tangente en x_0.



- On réalise la même approximation pour estimer $y(x_2)$:

$$y'(x_1) \simeq \frac{y(x_1+h)-y(x_1)}{h} = \frac{y(x_2)-y(x_1)}{h}.$$

En utilisant l'approximation de y_1 trouvée précédemment, on obtient alors l'approximation de $y(x_2)$ suivante :

$$y(x_2) \simeq y_1 + hy'(x_1).$$

Or la fonction y est solution de (E), donc on a

$$y'(x_1) = F(y(x_1)) \simeq F(y_1),$$

toujours en utilisant l'approximation précédente.
Ainsi, on obtient l'approximation suivante de $y(x_2)$:

$$y(x_2) \simeq y_1 + hF(y_1),$$

et on pose

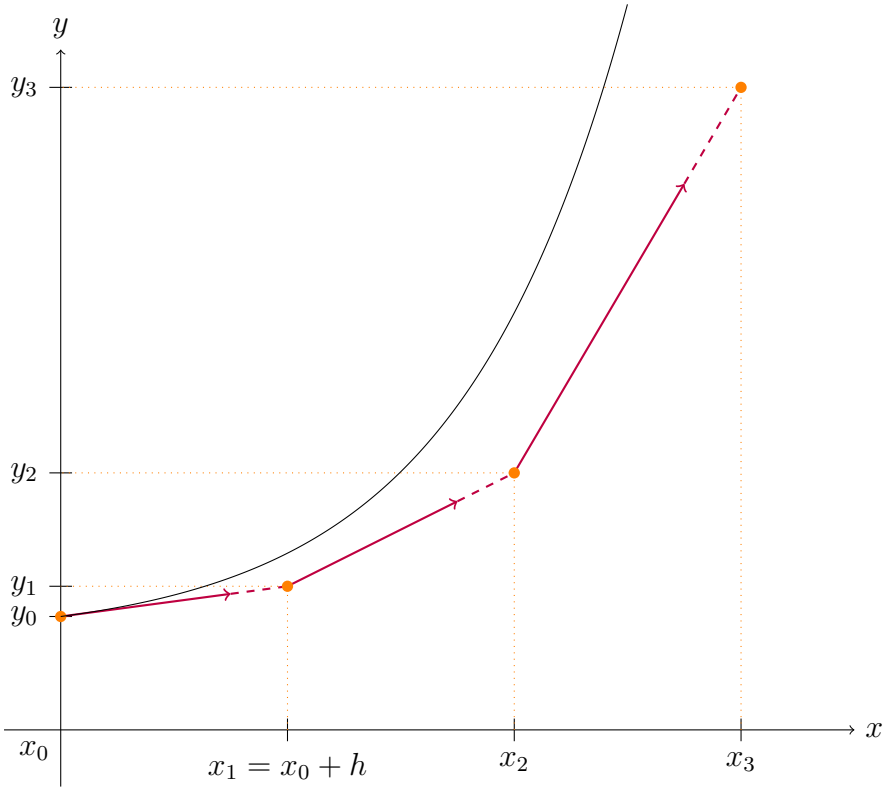
$$y_2 = y_1 + hF(y_1).$$

- On procède de la même manière pour les points suivants, en réutilisant à chaque étape l'approximation obtenue à l'étape précédente.

On définit alors la suite finie $(y_k)_{k\in[\![0;n]\!]}$ par :

$$\begin{cases} y_0 = y_0 \\ \forall k \in [\![0; n-1]\!], y_{k+1} = y_k + hF(y_k). \end{cases}$$

Notons alors que les approximations sont de moins en moins bonnes, puisqu'elles s'appuient sur les approximations précédentes. Il est donc très important de choisir un pas h très petit pour la subdivision, afin de minimiser l'erreur commise à chaque étape.




```python
def Euler(F, x_0, y_0, b, n):
    """
    Entrées :
        - une fonction F ;
        - trois flottants x_0, y_0 et b ;
        - un entier naturel n non nul.
    Sortie : les listes des valeurs des x_k et des y_k obtenues
             avec la méthode d'Euler.
    """
    h = (b - x_0) / n # h est le pas de la subdivision.
    x = x_0 # Au départ, x vaut x_0.
    y = y_0 # Au départ, y vaut y_0.
    abscisses = [x_0]
    ordonnees = [y_0]
    for k in range(n): # L'indice k va de 0 à n - 1.
        y = y + h * F(y) # Maintenant, y vaut y_{k + 1}.
        x = x + h # Maintenant, x vaut x_{k + 1}.
        abscisses.append(x)
        ordonnees.append(y)
    return (abscisses, ordonnees)
```