Cours

Révisions : Méthodes numériques

Lycée Thiers - Physique-Chimie - MPI/MPI*- 2023-2024

Contenu du programme officiel :

Notions et contenus	Capacités exigibles
1. Outils graphiques Représentation graphique d'un nuage de points. Représentation graphique d'une fonction. Courbes planes paramétrées.	Utiliser les fonctions de base de la bibliothèque `matplotlib` pour représenter un nuage de points. Utiliser les fonctions de base de la bibliothèque `matplotlib` pour tracer la courbe représentative d'une fonction. Utiliser les fonctions de base de la bibliothèque `matplotlib` pour tracer une courbe plane paramétrée.
2. Équations algébriques Résolution d'une équation algébrique ou d'une équation transcendante : méthode dichotomique.	Déterminer, en s'appuyant sur une représentation graphique, un intervalle adapté à la recherche numérique d'une racine par une méthode dichotomique. Mettre en œuvre une méthode dichotomique afin de résoudre une équation avec une précision donnée. Utiliser la fonction `bisect` de la bibliothèque `scipy.optimize` (sa spécification étant fournie).
3. Intégration – Dérivation Calcul approché d'une intégrale sur un segment par la méthode des rectangles. Calcul approché du nombre dérivé d'une fonction en un point.	Mettre en œuvre la méthode des rectangles pour calculer une valeur approchée d'une intégrale sur un segment. Utiliser un schéma numérique pour déterminer une valeur approchée du nombre dérivé d'une fonction en un point.
4. Équations différentielles Équations différentielles d'ordre 1. Équations différentielles d'ordre supérieur ou égal à 2	Mettre en œuvre la méthode d'Euler explicite afin de résoudre une équation différentielle d'ordre 1. Transformer une équation différentielle d'ordre n en un système différentiel de n équations d'ordre 1. Utiliser la fonction `odeint` de la bibliothèque `scipy.integrate` (sa spécification étant fournie).
5. Probabilité – statistiques Variable aléatoire. Régression linéaire.	Utiliser les fonctions de base des bibliothèques `random` et/ou `numpy` (leurs spécifications étant fournies) pour réaliser des tirages d'une variable aléatoire. Utiliser la fonction `hist` de la bibliothèque `matplotlib.pyplot` (sa spécification étant fournie) pour représenter les résultats d'un ensemble de tirages d'une variable aléatoire. Déterminer la moyenne et l'écart-type d'un ensemble de tirages d'une variable aléatoire. Utiliser la fonction `polyfit` de la bibliothèque `numpy` (sa spécification étant fournie) pour exploiter des données. Utiliser la fonction `random.normal` de la bibliothèque `numpy` (sa spécification étant fournie) pour simuler un processus aléatoire.

En gras les points devant faire l'objet d'une approche expérimentale.

Table des matières

1 Outils graphiques

Pour tracer des courbes, nous avons besoin des bibliothèques `numpy` et `matplotlib`.

Commande `plot()` : La fonction `plot()` permet de tracer des courbes qui relient des points dont les abscisses et ordonnées sont fournies dans des tableaux. Le code suivant permet de tracer les points du tableaux `x` en fonction de ceux du tableau `y`. De base, les points sont reliés entre eux.

```
import numpy as np
import matplotlib.pyplot as plt
```

```
x = np.array([1, 3, 4, 6])
y = np.array([2, 3, 5, 1])
plt.plot(x, y)

plt.show() # affiche la figure a l'écran, nécessaire pour avoir un affichage
```

Tant que la commande `plt.show()` n'est pas affichée, toutes les courbes générées par un `plot` seront stockées pour être affichées au final sur le même graphique. La commande initiale `plt.figure()` permet d'imposer le stockage des courbes dans une nouvelle figure (affichée éventuellement ultérieurement).

> ***Remarque :*** *On peut modifier la couleur de la courbe avec l'option* `plt.plot(x, y,'r')` *en modifiant la lettre (*`r` *pour rouge,* `'b'` *pour bleu,* `'y'` *pour yellow...). On peut modifier l'affichage de la courbe avec* `plt.plot(x, y,'+')` *(*`'+'` *pour des + non reliées,* `'o'` *pour des ronds non reliées,* `'o-'` *pour des ronds reliés par un trait plein...)*

Tracé de fonctions : Pour tracer une fonction, il faut tracer suffisamment de points proches les uns des autres pour donner l'impression d'une fonction continue. Par exemple pour la fonction cosinus ci-dessous.

```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0, 2*np.pi, 30) # créer une base des x composés de 30 points répartis uniformément entre 0 et 2 pi
                                                                    exclu.
y = np.cos(x)
plt.plot(x, y)

plt.show() # affiche la figure à l'écran
```

> ***Remarque :*** *On peut tracer en coordonnées polaires avec un paramètre* `theta` *et un paramètre* `r` *représentant $r(\theta)$ à l'aide de la commande* `plt.polar(theta, r)`.

Autres commandes :

- `plt.title("Titre")` permet l'affichage d'un titre au graphique ;
- `plt.axis([x1,x2,y1,y2])` permet d'imposer les valeurs minimales et maximales des axes ;
- `plt.legend()` permet l'affichage d'une légende, pour cela, il faut que la commande `plot` contienne un label `plt.plot(x, y, label ="Courbe 1")`, la commande `plt.legend(loc='best')` permet d'optimiser l'emplacement de la légende ;
- `plt.xlabel('Abscisse x')` et `plt.ylabel('Ordonnee y')` permet de nommer les axes ;
- `plt.errorbar(x,y,xerr=Delta_x,yerr=Delta_y,fmt='b+')` permet de tracer le tableau `x` avec une incertitude sur chaque point donnée dans la tableau `Delta_x` en fonction du tableau `y` (avec les incertitudes `Delta_x`). L'option `fmt='b+'` impose que chaque barre d'erreur sera matérialisé par une croix bleue.

2 Résolution numérique d'équation par méthode dichotomique

Soit une équation, éventuellement non linéaire, de la forme $f(x) = 0$ avec f une fonction définie et s'annulant une unique fois sur un intervalle $[a, b]$.

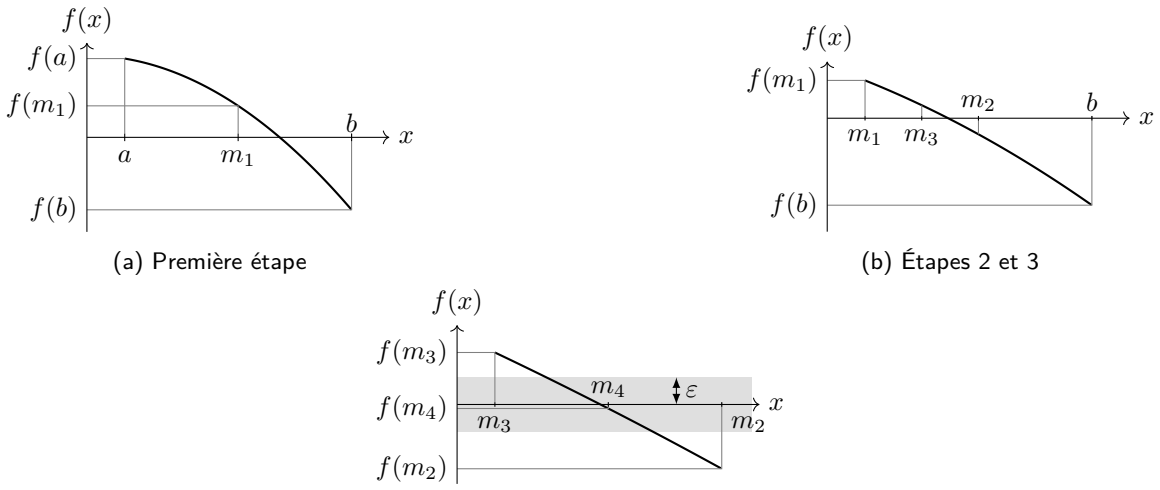
Présentation de la méthode

La méthode dichotomique permet de trouver une solution approchée de cette équation. En effet, après avoir fixé une précision arbitraire ε, la méthode permet de renvoyer c telle que $|f(c)| < \varepsilon$. La méthode dichotomique peut se décrire avec les schéma de la figure 1.

- **Étape 1** (figure 1a) : On cherche une solution entre a et b. On pose $m_1 = (a + b)/2$ le milieu du segment $[a, b]$. On constate que la valeur de $f(m_1)$ est de même signe que $f(a)$, la solution est à rechercher entre m_1 et b.
- **Étape 2 et 3** (figure 1b) : On cherche une solution entre m_1 et b. On pose $m_2 = (m_1 + b)/2$ le milieu du segment $[m_1, b]$ On constate que la valeur de $f(m_2)$ est de même signe que $f(b)$, la solution est à rechercher entre m_1 et m_2. On cherche ensuite une solution entre m_1 et m_2 puis entre m_3 et m_2 avec m_3 le milieu du segment $[m_1, m_2]$.
- **Étape finale** (figure 1c) : On cherche une solution entre m_2 et m_3. On pose m_4 le milieu du segment $[m_2, m_3]$. On constate que $|f(m_4)|$ est inférieure à ε fixée, m_4 est donc la valeur approchée renvoyée par l'algorithme.

> ***Remarque :*** *Il est possible de trouver une solution approchée de l'unique solution de l'équation $h(x) = g(x)$ avec h et g deux fonctions sur l'intervalle en posant la fonction $f = h - g$.*

Pour trouver la valeur approchée, on réalise l'algorithme suivant, qui prend en entrée la fonction f, les réels a et b ainsi que la précision ε :



(a) Première étape

(b) Étapes 2 et 3

(c) Étape finale

Fig. 1 – Représentation graphiques des étapes de la recherche de l'annulation d'une fonction par dichotomie.

- ▷ initialiser $x_{\min} = a$ et $x_{\max} = b$;
- ▷ initialiser le milieu de l'intervalle $x_{\mathrm{m}} = (x_{\min} + x_{\max})/2$;
- ▷ tant que $|f(x_{\mathrm{m}})| > \varepsilon$, faire :
  - ▷ si $f(x_{\min})$ et $f(x_{\mathrm{m}})$ sont de même signe, alors la solution recherchée est comprise entre x_{m} et $x_{\max}$, on affecte donc à la variable $x_{\min}$ la valeur x_{m} ;
  - ▷ sinon, la solution recherchée est comprise entre $x_{\min}$ et x_{m}, on affecte donc à la variable $x_{\max}$ la valeur x_{m} ;
  - ▷ ensuite, on affecte à la variable x_{m} la nouvelle valeur du milieu de l'intervalle $(x_{\min} + x_{\max})/2$;
  - ▷ renvoyer x_{m}.

La valeur finale de x_{m} correspond à la valeur c recherchée.

Cette méthode permet de diviser par deux l'intervalle de recherche à chaque itération. Celui-ci diminue donc de taille rapidement. Si la fonction f s'annule une unique fois sur l'intervalle, cet algorithme se terminera nécessairement et fournira une valeur de c. La vitesse d'exécution de l'algorithme dépendra de ε. Plus on cherche une précision élevée, plus le programme sera long.

> ***Remarque :*** *Il est possible de prendre comme critère d'arrêt de la méthode $|x_{\max} - x_{\min}| < \varepsilon$. Ce choix pourra être plus rapide dans le cas de fonction avec une très forte pente.*

Plutôt que de coder systématiquement cet algorithme, on pourra utiliser la fonction `bisect(f,a,b)` de la bibliothèque `scipy.optimize` qui donne une valeur approchée de la solution de l'équation $f(x) = 0$ avec x compris entre a et b par un algorithme de dichotomie optimisé.

Code python

Voici un exemple de code python correspondant à cet algorithme.

```
'''
On suppose que la fonction f est préalablement définie et qu'elle vérifie les hypothèses d'application de la méthode.

eps est la précision recherchée et a et b sont les deux bornes de l'intervalle de recherche de la solution
'''

# Initialisation des bornes de recherche de la solution
xmin = a
xmax = b

# Initialisation du milieu
xmilieu = (xmin + xmax) / 2

while abs(f(xmilieu)) > eps :
  if f(xmin)*f(xmilieu) > 0 :
    xmin = xmilieu
  else :
    xmax = xmilieu
  # On calcule xmilieu pour le tour suivant
  xmilieu = (xmin + xmax) / 2

# Les variables xmilieu, xmin ou xmax contiennent à la fin de cette algorithme une solution approchée de f(x) = 0
```

3 Intégration et dérivation

3.1 Intégration numérique par méthode des rectangles

Soit une fonction f définie et intégrable sur tout le segment $[a, b]$. La formule de Riemann donne une définition mathématique pour l'intégrale sur le segment $[a, b]$ grâce à la relation

$$\int_a^b f(x)\,\mathrm{d}x = \lim_{N\to+\infty}\left(\frac{b-a}{N}\sum_{i=0}^{N-1} f(t_i)\right)$$

avec t_i compris entre $x_i = a + i\dfrac{b-a}{N}$ et $x_{i+1} = a + (i+1)\dfrac{b-a}{N}$ (on vérifie bien $x_0 = a$ et $x_N = b$).

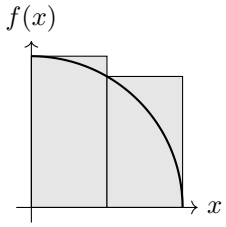
Cette formule permet de construire l'intégrale comme une somme de rectangles de largeur $h = \dfrac{b-a}{N}$ et de hauteur $f(t_i)$. Informatiquement, cette définition permet de calculer de façon approximative une intégrale. Le nombre h est appelé « pas » de l'intégrale et doit être choisi le plus petit possible. Autrement dit, le nombre N de subdivisions du segment $[a, b]$ doit être pris le plus grand possible.

La façon de choisir le point t_i permet de définir plusieurs méthodes d'approximation de l'intégrale.

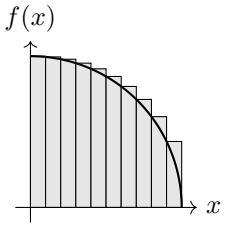
On représente ci-dessous plusieurs méthodes d'intégrations numériques sur un quart de cercle de rayon 1 représenté par la fonction $f : x \mapsto \sqrt{1-x^2}$ entre 0 et 1. Analytiquement, on a $\int_0^1 f(x)\mathrm{d}x = \pi/4 = 0.785\,392...$

Méthode des rectangles à droite :

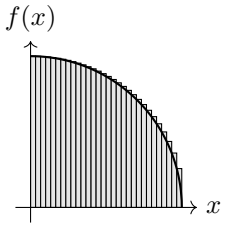
On prend $t_i = x_i$.



(a) 2 subdivisions, l'aire grisée vaut environ 0.933 012.



(b) 10 subdivisions, l'aire grisée vaut environ 0.826 129.

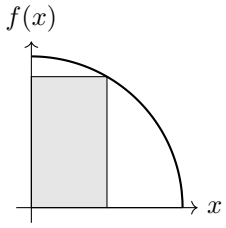


(c) 30 subdivisions, l'aire grisée vaut environ 0.800 277.

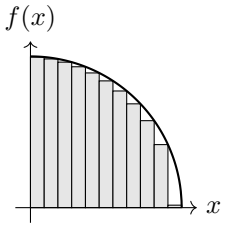
Fig. 2 – Visualisation de la méthode d'intégration numérique des rectangles à droite.

Méthode des rectangles à gauche :

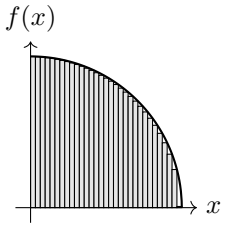
On prend $t_i = x_{i+1}$.



(a) 2 subdivisions, l'aire grisée vaut environ 0.433 012.



(b) 10 subdivisions, l'aire grisée vaut environ 0.726 129.

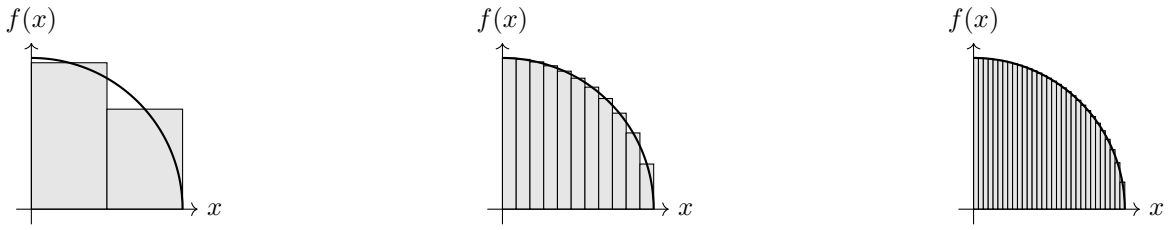


(c) 30 subdivisions, l'aire grisée vaut environ 0.766 944.

Fig. 3 – Visualisation de la méthode d'intégration numérique des rectangles à gauche.

Méthode du point médian :

On prend le milieu du segment $[x_i, x_{i+1}]$ soit $t_i = (x_i + x_{i+1})/2$.



(a) 2 subdivisions, l'aire grisée vaut environ 0.814 841.

(b) 10 subdivisions, l'aire grisée vaut environ 0.788 102.

(c) 30 subdivisions, l'aire grisée vaut environ 0.785 921.

Fig. 4 – Visualisation de la méthode d'intégration numérique du point médian.

On constate que la méthode d'intégration numérique converge plus rapidement que les deux autres vers la valeur exacte. Ce résultat peut se démontrer théoriquement. L'erreur commise par les deux premières méthodes numériques est proportionnelle à $1/N$ (avec N le nombre de subdivisions) tandis que l'erreur commise pour la troisième méthode est proportionnelle à $1/N^2$. À N fixé, la méthode du point médian est donc plus proche de la valeur exacte que les autres. Cette convergence plus rapide est due au fait que, avec la méthode de point médian, les sur-estimations de l'intégrale par le rectangle sont en partie compensée par les sous-estimations sur le même intervalle.

> ***Remarque :*** *Plutôt que d'approximer la fonction par un rectangle, on peut choisir de l'approximer par exemple par un trapèze ou un polynôme. Ces deux approximations ont, pour N fixé, une erreur plus faible entre la valeur numérique et la valeur exacte de l'intégrale.*

Sous python, pour obtenir un résultat numérique plus satisfaisant, on peut utiliser la fonction déjà implémentée quad de la bibliothèque `scipy.integrate` qui est optimisé pour le calcul numérique d'intégrales.

3.2 Calcul approché d'un nombre dérivé

Considérons une fonction f continue et dérivable sur un intervalle I.

Informatiquement, il est impossible de sauvegarder en mémoire l'ensemble des valeurs prises par la fonction sur I. En effet, ce nombre de valeurs est infini alors que la mémoire informatique est finie. Ainsi, tout comme pour le tracé graphique, la fonction sera représentée par une liste finie de valeurs de f, image d'une liste de valeur finie de I.

Par définition, un nombre dérivé est la limite du taux d'accroissement. Cette limite peut s'écrire de plusieurs façons, par exemple, pour une fonction continue et dérivable autour du point x, on a

$$f'(x) = \lim_{\varepsilon\to 0} \frac{f(x+\varepsilon)-f(x)}{\varepsilon} = \lim_{\varepsilon\to 0} \frac{f(x)-f(x-\varepsilon)}{\varepsilon} = \lim_{\varepsilon\to 0} \frac{f(x+\varepsilon)-f(x-\varepsilon)}{2\varepsilon}\ .$$

Ces trois expressions ont la même valeur grâce à la limite mathématique.

Informatiquement, il est impossible de faire tendre l'écart entre deux points vers 0. Cet écart est au minimum égal à une valeur appelée le **pas**, qui est traditionnellement noté h.

Les trois expressions de la limite mathématique peuvent donc chacune être approché par un quotient.

Limite à droite :

L'estimation de la limite par la limite à droite est représentée figure 5a. Elle consiste à faire l'approximation suivante

$$\lim_{\varepsilon\to 0} \frac{f(x_i+\varepsilon)-f(x_i)}{\varepsilon} \approx \frac{f(x_{i+1})-f(x_i)}{h}\ .$$

On pose alors $f'_d(x_i) = \dfrac{f(x_{i+1})-f(x_i)}{h}$. L'hypothèse $f'(x_i) = f'_d(x_i)$ consiste à approximer la pente autour de x_i par la pente moyenne entre x_i et x_{i+1}.

Limite à gauche :

L'estimation de la limite par la limite à gauche est représentée figure 5a. Elle consiste à faire l'approximation suivante

$$\lim_{\varepsilon\to 0} \frac{f(x_i)-f(x_i-\varepsilon)}{\varepsilon} \approx \frac{f(x_i)-f(x_{i-1})}{h}\ .$$

On pose alors $f'_g(x_i) = \dfrac{f(x_i)-f(x_{i-1})}{h}$. L'hypothèse $f'(x_i) = f'_g(x_i)$ consiste à approximer la pente autour de x_i par la pente moyenne entre x_{i-1} et x_i.

Limite moyenne :

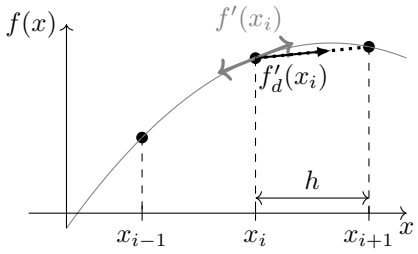
L'estimation de la limite par la limite moyenne est représentée figure 5c. Elle consiste à faire l'approximation suivante

$$\lim_{\varepsilon\to 0}\frac{f(x_i+\varepsilon)-f(x_i-\varepsilon)}{2\varepsilon}\approx\frac{f(x_{i+1})-f(x_{i-1})}{2h}\ .$$

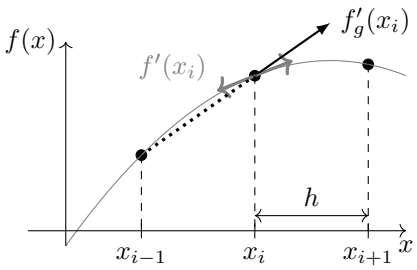
On pose alors $f'_m(x_i)=\dfrac{f(x_{i+1})-f(x_{i-1})}{h}$. L'hypothèse $f'(x_i)=f'_m(x_i)$ consiste à approximer la pente autour de x_i par la pente moyenne entre x_{i-1} et x_{i+1}.

Propriété. La formule de la limite moyenne permet d'estimer au mieux le nombre dérivé autour d'un point. Il est nécessaire que le pas h soit très petit devant la longueur typique des variations de la fonction f.

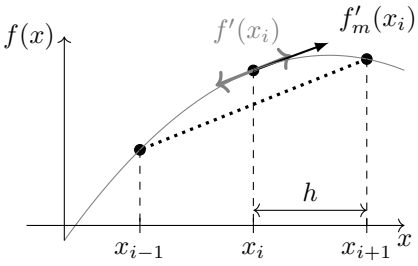
> ***Exemple 1 :*** *Pour estimer numériquement les nombres dérivés de la fonction $x\mapsto\sin(x)$, il faut que le pas soit très petit devant π qui est la taille typique des variations de la fonction.*



(a) La limite à droite consiste à estimer la tangente par la flèche noire à l'aide de la droite moyenne en pointillé.



(b) La limite à gauche consiste à estimer la tangente par la flèche noire à l'aide de la droite moyenne en pointillé.



(c) La limite moyenne consiste à estimer la tangente par la flèche noire à l'aide de la droite moyenne en pointillé.

Fig. 5 – Représentations graphiques des calculs numériques de nombre dérivés. La double flèche grise représente la tangente en x_i, soit le nombre dérivé que l'on cherche à évaluer.

4 Résolution numérique d'équations différentielles

4.1 Méthode d'Euler explicite sur une équation différentielle d'ordre un

On cherche à résoudre numériquement un problème général de Cauchy d'ordre un, soit un système composé d'une équation différentielle d'ordre un et d'une condition initiale. Un tel problème se met sous la forme

$$\begin{cases} y'(t) &= f(t,y(t)) \\ y(t_0) &= y_0 \end{cases} \tag{4.1}$$

avec f une fonction de deux variables et t_0 et y_0 deux constantes définissant les conditions initiales.

> ***Exemple 2 :*** ▷ *L'équation différentielle du premier ordre $my'(t)+\kappa y^2(t)=mg$ avec κ, m et g des constantes réelles peut s'écrit sous la forme de l'équation (4.1) en posant $f(t,y)=-\dfrac{\kappa}{m}y^2+g$. Dans ce cas, la fonction f ne dépend pas explicitement du temps.*
>
> ▷ *L'équation différentielle du premier ordre $y'(t)+\dfrac{1}{\tau}y(t)=\dfrac{C}{\tau}$ avec τ et C des constantes réelles peut s'écrit sous la forme de l'équation (4.1) en posant $f(t,y)=-\dfrac{1}{\tau}y+\dfrac{C}{\tau}$. Dans ce cas, la fonction f ne dépend pas explicitement du temps.*

Numériquement, on résout le système (4.1) sur l'intervalle $I = [t_0, t_m]$ de façon approximative en raisonnant de proche en proche. Pour cela, on définit un nombre de points N correspondant au nombre de subdivisions souhaitées de l'intervalle I. Cela permet de définir le **pas** de résolution $\boxed{h = \dfrac{t_m - t_0}{N}}$.

L'instant t_i est défini par la relation $t_i = t_0 + i \times h$. De façon exacte, par définition de l'intégrale, on a la relation

$$y(t_{i+1}) = y(t_i) + \int_{t_i}^{t_{i+1}} y'(t)\mathrm{d}t \ . \tag{4.2}$$

Or, grâce au système (4.1), cette relation devient

$$y(t_{i+1}) = y(t_i) + \int_{t_i}^{t_{i+1}} f(t, y(t))\mathrm{d}t \ .$$

Toute résolution numérique d'équation différentielle consiste à donner une méthode numérique pour estimer l'intégrale de cette relation.

Définition. La **méthode d'Euler explicite** consiste à réaliser l'approximation

$$\int_{t_i}^{t_{i+1}} f(t, y(t))\mathrm{d}t \approx h f(t_i, y(t_i)) \ .$$

Cette estimation correspond à un calcul d'intégrale à l'aide de la méthode des rectangles à gauche.
L'utilisation de la méthode d'Euler explicite permet d'approximer la relation (4.2) par la relation

$$\boxed{y(t_{i+1}) = y(t_i) + h f(t_i, y(t_i))}$$

avec h le **pas** de la méthode. Cette relation s'appelle **l'équation aux différences finies**.

On peut remarquer que cette relation consiste simplement à estimer $y(t_{i+1})$ en approximant linéairement la courbe représentant y avec sa tangente représentant y' calculée à l'aide de la fonction f. En d'autres termes, il s'agit d'un développement limité de la fonction y tronqué à l'ordre 1.

> ***Remarque :*** *L'utilisation de la méthode des rectangles à droite s'appelle la méthode d'Euler implicite tandis que la méthode du point médian est la méthode Runge Kutta d'ordre 2. Il existe de nombreuses autre méthodes pour estimer cette intégrale. La méthode d'Euler explicite est la plus simple et la moins précise.*

Exemple de l'application de la méthode d'Euler explicite :

Considérons le problème de Cauchy suivant :

$$\begin{cases} y'(t) &= -y(t) \\ y(0) &= 1 \end{cases} \quad \Rightarrow \quad f(t, y) = -y \tag{4.3}$$

Ce système se résout explicitement et la solution est $y(t) = \exp(-t)$ pour tout $t \in \mathbb{R}$.

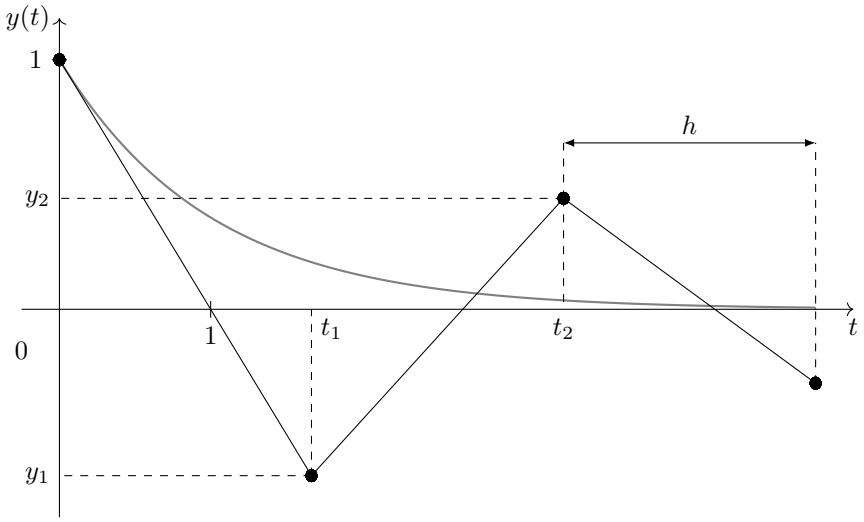
On réalise une résolution numérique pour $t \in [0, 4]$. On constate sur la figure 6a que, pour un faible nombre de points, le résultat de la méthode d'Euler explicite oscille alors que la solution est strictement monotone. Cette propriété traduit la **non stabilité** de cette méthode numérique. De plus, on constate sur l'ensemble des figures 6 que le nombre de points doit être important pour s'approcher de la solution exacte.

L'erreur $y(t_k) - y_k$ commise lors du calcul de l'étape k est proportionnelle à $y''(t_k)h^2/2$ (ce résultat provient de l'écriture du développement limité d'ordre 2 de y). De plus, cette erreur est ensuite cumulative car les valeurs sont calculées de proches en proches. C'est-à-dire que l'erreur commise lors de l'ordre k va se répercuter à tous les ordres suivants.

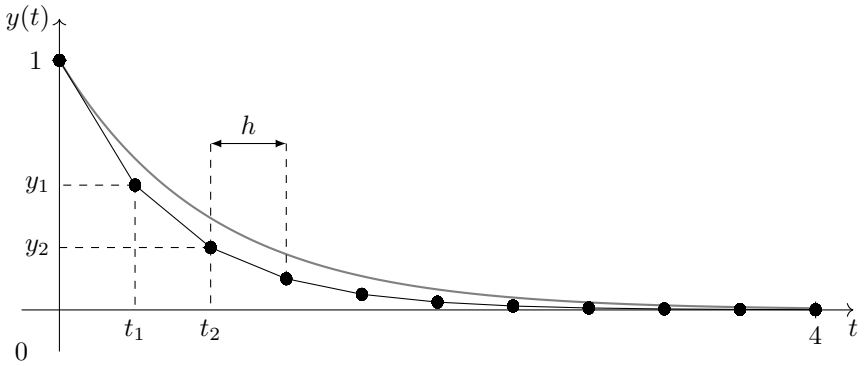
Pour minimiser cette erreur de l'algorithme, il est nécessaire de diminuer h le plus possible. Toutefois, plus h est petit, plus le temps de calcul sera long. Il faut donc trouver un compromis entre précision et temps de calcul.

Propriété. Pour que la méthode d'Euler explicite donne un résultat « assez proche » de la solution réelle, il est nécessaire de prendre un pas de calcul h très faible devant la taille des variations typiques de la fonctions.

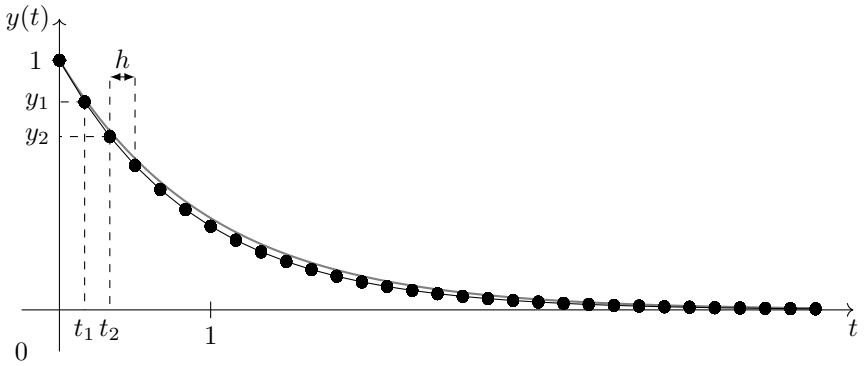
Pratiquement, il faut généralement prendre au moins un facteur 100 entre h et ces variations typiques. Dans le cas de l'exemple, l'intervalle de variation typique de l'exponentielle est de 1, il faudrait donc prendre $h = 0.01$ soit 400 subdivisions de l'intervalle $[0, 4]$.



(a) Méthode d'Euler explicite avec 3 subdivisions de l'intervalle.



(b) Méthode d'Euler explicite avec 10 subdivisions de l'intervalle.



(c) Méthode d'Euler explicite avec 30 subdivisions de l'intervalle.

Fig. 6 – Schéma des résultats de la méthode d'Euler explicite pour trois pas différents. En gris la fonction exacte et les points noirs représentent les points y_i issus de la méthode d'Euler.

Code python

Voici un exemple de code python correspondant à cet algorithme.

```
'''
Définition de la fonction d'étude yprime.
Nous prenons l'habitude de toujours mettre le temps t en seconde variable de cette fonction.
Ce sera indispensable pour l'utilisation de odeint.
'''
def fprime(f,t):
  return - f
```

```
'''
Résolution
'''
# Le temps caractéristique est enregistré comme variable globale, le pas de résolution h ainsi que la condition
                                                                initiale y0
tau = 1
h = 0.01 # On prend soin d'avoir h << tau
y0 = 1

# a et b sont les bornes de l'intervalle de résolution
a,b = 0,4

# La liste y contiendra les valeurs de y entre a et b, tandis que la liste t contient les instants correspondants
y = [y0]
t = [a]

# Boucle de calcul

while t[-1] < b : # on rappelle que t[-1] contient le dernier élément de la liste t
  val = y[-1] + h * fprime(t[-1],y[-1])
  y.append(val)
  t.append(t[-1] + h)
```

4.2 Réduction de l'ordre d'une équation différentielle

La méthode d'Euler explicite, comme toutes les méthodes de résolution numérique d'équations différentielles, s'applique aux équations différentielles d'ordre 1.

Pour discuter les équations différentielles d'ordre plus élevé, il faut utiliser la méthode de réduction de l'ordre d'une équation différentielle, qui consiste à transformer l'équation différentielle scalaire en équation différentielle vectorielle.

Prenons pas exemple l'équation différentielle du second ordre

$$\ddot{x}(t) + \frac{\kappa}{m}(\dot{x}(t))^2 + \omega_0^2 x(t) = K\sin(\omega t)\ . \tag{4.4}$$

Posons le vecteur $X(t) = (x(t), v(t))$ avec $v(t) = \dot{x}(t)$. On constate que la dérivée première de ce vecteur s'écrit $\dot{X}(t) = (\dot{x}(t), \dot{v}(t)) = (v(t), \ddot{x}(t))$.

Posons alors la fonction vectorielle F, qui prend le temps t ainsi qu'un vecteur de deux dimensions x et v en paramètres et qui renvoie un vecteur à deux dimensions, telle que

$$F(t, X) = F\left(t, \begin{pmatrix} x \\ v \end{pmatrix}\right) = \begin{pmatrix} v \\ -\frac{\kappa}{m}v^2 - \omega_0^2 x + K\sin(\omega t) \end{pmatrix}\ .$$

On constate alors que $\dot{X}(t) = F(t, X)$ est équivalent à l'équation différentielle (4.4). Mais cette nouvelle équation est une équation différentielle du premier ordre vectorielle. Le schéma numérique de résolution s'applique à cette équation. La différence est qu'au lieu de manipuler uniquement y_i à chaque étape, il est maintenant nécessaire de manipuler un couple de valeur (x_i, v_i) qui varient selon le schéma numérique à chaque étape.

Code python

Sur l'exemple du pendule simple, l'équation différentielle est $\ddot{\theta}(t) + \omega_0^2\theta(t) = 0$.

```
'''
Définition de la fonction vectorielle, où om est définie comme variable globale par ailleurs
Attention, le temps doit être la seconde variable
'''
def pendule(y,t):
    # Attention, les angles doivent être en radian
    theta,thetapoint = y
    return [thetapoint,-om**2*np.sin(theta)]
```

4.3 Utilisation du module `odeint`

Nous avons constaté dans les figures 6 que la méthode d'Euler explicite n'était pas optimale à moins d'avoir un grand nombre de points. En réalité, la méthode d'Euler explicite est le schéma numérique le plus simple possible pour résoudre une équation différentielle. À ce titre, c'est le moins efficace. La seule vertu de cette méthode numérique est sa simplicité de mise en œuvre. Toutefois, pour tout travail sérieux sur une résolution numérique d'équation, on utiliser la fonction `odeint` de la bibliothèque `scipy.integrate`. Celle-ci permet de résoudre numériquement des équations différentielles de façon optimisée.

Code python

```
from scipy.integrate import odeint

# Initialisation de la liste des temps
t = np.linspace(0, 10, 100)

# Résolution de l'équation différentielle
sol = odeint(pendule, [0,0.1], t)
```

`sol` est une matrice. La première colonne contient les valeurs des angles (première variable de `pendule`) et la seconde contient les vitesses de rotation (seconde variable de `pendule`).