Méthode d'Euler
1er ordre

Informatique pour tous

Objectif

On veut calculer de façon approchée une solution $y : I \subseteq \mathbb{R} \longrightarrow \mathbb{R}$ d'une équation différentielle du 1er ordre, de la forme:

$$y'(t) = f(t, y(t))$$

Objectif

On veut calculer de façon approchée une solution $y : I \subseteq \mathbb{R} \longrightarrow \mathbb{R}$ d'une équation différentielle du 1er ordre, de la forme:

$$y'(t) = f(t, y(t))$$

1er ordre signifie qu'il y a seulement $y'(t)$ et $y(t)$ qui apparaissent, mais pas $y''(t)$, par exemple.

Équation différentielle du 1er ordre

$$y'(t) = f(t, y(t))$$

Quelques exemples:

1. Si $f(a, b) = a$:

Équation différentielle du 1er ordre

$$y'(t) = f(t, y(t))$$

Quelques exemples:

1. Si $f(a, b) = a$:

   solutions: $\boxed{y(t) = \frac{t^2}{2} + K}$

2. Si $f(a, b) = b$:

Équation différentielle du 1er ordre

$$y'(t) = f(t, y(t))$$

Quelques exemples:

1. Si $f(a, b) = a$:

   solutions: $\boxed{y(t) = \frac{t^2}{2} + K}$

2. Si $f(a, b) = b$:

   solutions: $\boxed{y(t) = Ke^t}$

3. Si $f(a, b) = ab$:

Équation différentielle du 1er ordre

$$y'(t) = f(t, y(t))$$

Quelques exemples:

1. Si $f(a, b) = a$:

   solutions: $\boxed{y(t) = \frac{t^2}{2} + K}$

2. Si $f(a, b) = b$:

   solutions: $\boxed{y(t) = Ke^t}$

3. Si $f(a, b) = ab$:

   solutions: $\boxed{y(t) = Ke^{\frac{t^2}{2}}}$

4. En général: on ne sait pas résoudre explicitement.

Théorème de Cauchy

Un **problème de Cauchy** consiste à trouver les solutions y de:

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

Théorème de Cauchy

Un **problème de Cauchy** consiste à trouver les solutions y de:

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

Théorème de Cauchy-Lipschitz (admis)

Si f est de classe C^1 alors le problème de Cauchy ci-dessus possède une **unique solution** maximale, définie sur un intervalle ouvert.

(Une version plus précise sera vue en maths, en 2ème année)

Méthode d'Euler

La **méthode d'Euler** consiste à approximer la solution y d'un problème de Cauchy sur un intervalle I.

On souhaite approximer une fonction (un objet continu) alors que l'informatique ne permet que de traiter d'objets discrets (finis).

Méthode d'Euler

La **méthode d'Euler** consiste à approximer la solution y d'un problème de Cauchy sur un intervalle I.

On souhaite approximer une fonction (un objet continu) alors que l'informatique ne permet que de traiter d'objets discrets (finis).

On va donc **discrétiser** le problème: on découpe I en n points t_0, t_1, ..., t_{n-1} régulièrement espacés de h (le **pas**) et on cherche des approximations y_k de $y(t_k)$.

Plus h est petit, plus l'approximation est bonne.

Méthode d'Euler

Soit y une solution de:

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

Méthode d'Euler

Soit y une solution de:

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

On peut approximer $y'(t_k)$ par un taux d'accroissement:

$$\frac{y_{k+1} - y_k}{t_{k+1} - t_k} \approx y'(t_k)$$

Méthode d'Euler

Soit y une solution de:

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

On peut approximer $y'(t_k)$ par un taux d'accroissement:

$$\frac{y_{k+1} - y_k}{t_{k+1} - t_k} \approx y'(t_k) = f(t_k, y(t_k))$$

Méthode d'Euler

Soit y une solution de:

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

On peut approximer $y'(t_k)$ par un taux d'accroissement:

$$\frac{y_{k+1} - y_k}{t_{k+1} - t_k} \approx y'(t_k) = f(t_k, y(t_k))$$

D'où:

$$\boxed{y_{k+1} = y_k + (\underbrace{t_{k+1} - t_k}_{h}) \times f(t_k, y_k)}$$

Résumé de la méthode d'Euler

Si f est C^1, l'équation différentielle suivante a une unique solution y avec une valeur fixée $y(t_0)$:

$$y'(t) = f(t, y(t))$$

Résumé de la méthode d'Euler

Si f est C^1, l'équation différentielle suivante a une unique solution y avec une valeur fixée $y(t_0)$:

$$y'(t) = f(t, y(t))$$

La méthode d'Euler (explicite), de pas h, consiste à approximer y par une suite récurrente $(y_k)_{0 \leq k < n}$ telle que:

1. $y_0 = y(t_0)$
2. $y_{k+1} = y_k + h \times f(t_k, y_k)$

On espère alors que $y_k \approx y(t_k)$.

Exemple

On souhaite approximer la solution, sur $[0, 1]$, de:

$$\left\{ \begin{array}{l} y'(t) = y(t) \\ y(0) = 1 \end{array} \right.$$

Exemple

On souhaite approximer la solution, sur $[0,1]$, de:

$$\begin{cases} y'(t) = y(t) \\ y(0) = 1 \end{cases}$$

On subdivise, par exemple, $[0,1]$ en 101 points: $t_0 = 0$, $t_1 = 0.01$, ..., $t_{99} = 0.99$, $t_{100} = 1$. Le pas est donc

Exemple

On souhaite approximer la solution, sur $[0, 1]$, de:

$$\left\{ \begin{array}{l} y'(t) = y(t) \\ y(0) = 1 \end{array} \right.$$

On subdivise, par exemple, $[0, 1]$ en 101 points: $t_0 = 0$, $t_1 = 0.01$, ..., $t_{99} = 0.99$, $t_{100} = 1$. Le pas est donc $h = 0.01$.

Les approximations de la méthode d'Euler vérifient:

Exemple

On souhaite approximer la solution, sur $[0, 1]$, de:

$$\begin{cases} y'(t) = y(t) \\ y(0) = 1 \end{cases}$$

On subdivise, par exemple, $[0, 1]$ en 101 points: $t_0 = 0$, $t_1 = 0.01$, ..., $t_{99} = 0.99$, $t_{100} = 1$. Le pas est donc $h = 0.01$.

Les approximations de la méthode d'Euler vérifient:

$$y_{k+1} = y_k + h \times y_k$$

Exemple

Implémentation en Python:

Exemple

Implémentation en Python:

```
y = [1]
for k in range(0, 100):
    y.append(y[k] + 0.01 * y[k])
```

On peut alors obtenir une approximation de *e*:

Exemple

Implémentation en Python:

```
y = [1]
for k in range(0, 100):
    y.append(y[k] + 0.01 * y[k])
```

On peut alors obtenir une approximation de e:

```
In [15]: y[100]
Out[15]: 2.704813829421526
```

Exemple

Implémentation en Python:

```
y = [1]
for k in range(0, 100):
    y.append(y[k] + 0.01 * y[k])
```

On peut alors obtenir une approximation de e:

```
In [15]: y[100]
Out[15]: 2.704813829421526
```

Question

Comment afficher graphiquement les approximations obtenues?

Exemple

Implémentation en Python:

```
y = [1]
for k in range(0, 100):
    y.append(y[k] + 0.01 * y[k])
```

On peut alors obtenir une approximation de e:

```
In [15]: y[100]
Out[15]: 2.704813829421526
```

Question

Comment afficher graphiquement les approximations obtenues?

`plt.plot(t, y)` où `t` est la liste des temps d'approximations t_k.

Exemple d'équation différentielle non linéaire

Écrire en Python la méthode d'Euler sur $[-2, 2]$, avec un pas 0.001, appliquée au problème de Cauchy:

$$\begin{cases} y'(t) = t\sin(y(t)) + 1 \\ y(-2) = -1 \end{cases}$$

Exemple d'équation différentielle non linéaire

Écrire en Python la méthode d'Euler sur $[-2, 2]$, avec un pas 0.001, appliquée au problème de Cauchy:

$$\left\{\begin{array}{l} y'(t) = t\sin(y(t)) + 1 \\ y(-2) = -1 \end{array}\right.$$

Voir le code euler_ex sur hugoprépa.

Implémentation de la méthode d'Euler générale

Écrire une fonction ayant en argument une fonction f, des valeurs initiales $t0$, $y0$, un pas h, un nombre d'itérations n et renvoyant la liste des t_k et des y_k ($0 \leq k \leq n$) de la méthode d'Euler appliquée à:

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

Implémentation de la méthode d'Euler générale

```python
def euler(f, t0, y0, h, n):
    t = [t0]
    y = [y0]
    for k in range(n):
        y.append(y[-1] + h * f(t[-1], y[-1]))
        t.append(t[-1] + h)
    return [t, y]
```

Implémentation de la méthode d'Euler générale

```python
def euler(f, t0, y0, h, n):
    t = [t0]
    y = [y0]
    for k in range(n):
        y.append(y[-1] + h * f(t[-1], y[-1]))
        t.append(t[-1] + h)
    return [t, y]
```

Question

Utiliser cette fonction pour approcher sur $[2, 7]$, avec un pas de 0.1, une solution de:

$$\begin{cases} y'(t) = \sqrt{t} + y(t)^2 \\ y(2) = 1 \end{cases}$$

Implémentation de la méthode d'Euler générale

```python
def euler(f, t0, y0, h, n):
    t = [t0]
    y = [y0]
    for k in range(n):
        y.append(y[-1] + h * f(t[-1], y[-1]))
        t.append(t[-1] + h)
    return [t, y]
```

Quelle est la complexité de `euler`?

Implémentation de la méthode d'Euler générale

```
def euler(f, t0, y0, h, n):
    t = [t0]
    y = [y0]
    for k in range(n):
        y.append(y[-1] + h * f(t[-1], y[-1]))
        t.append(t[-1] + h)
    return [t, y]
```

Quelle est la complexité de `euler`?

$O(n)$, si f se calcule en temps constant.

Implémentation de la méthode d'Euler générale

```python
def euler(f, t0, y0, h, n):
    t = [t0]
    y = [y0]
    for k in range(n):
        y.append(y[-1] + h * f(t[-1], y[-1]))
        t.append(t[-1] + h)
    return [t, y]
```

Quelle est la complexité de `euler`?

$O(n)$, si f se calcule en temps constant.
La méthode est plus précise si le nombre de points d'approximations est élevé, mais elle est aussi plus lente.

Implémentation de la méthode d'Euler générale

```
def euler(f, t0, y0, h, n):
    t = [t0]
    y = [y0]
    for k in range(n):
        y.append(y[-1] + h * f(t[-1], y[-1]))
        t.append(t[-1] + h)
    return [t, y]
```

Si on veut approximer une solution sur un intervalle de longueur ℓ, alors $h = \frac{\ell}{n}$ et `euler` est aussi en $O(\frac{1}{h})$: plus le pas est petit, plus la méthode est précise mais lente.

Question 19 :

On considère le problème de cauchy : $\begin{cases} \frac{dy}{dt}(t) = y^3(t) \\ y(0) = 1 \end{cases}$ pour $t \in [0, 1]$. On décide de calculer de manière approchée par une méthode d'Euler la solution. Soit n un entier non nul on pose $y_k = y(\frac{k}{n})$. Parmi les assertions suivantes, lesquelles correspondent bien à un schéma d'Euler explicite pour le problème de Cauchy posé :

A) $y_0 = 1$ et $y_{k+1} = y_k + n.y_k^3, \forall k \in [|0, n-1|]$.

B) $y_0 = 1$ et $y_{k+1} = ny_k + y_k^3, \forall k \in [|0, n-1|]$.

C) $y_0 = 1$ et $y_{k+1} = y_k + \frac{1}{n}.y_k^3, \forall k \in [|0, n-1|]$.

D) $y_0 = 1$ et $y_{k+1} = y_k + n.y_{k+1}^3, \forall k \in [|0, n-1|]$.

Question 19 :

On considère le problème de cauchy : $\begin{cases} \frac{dy}{dt}(t) = y^3(t) \\ y(0) = 1 \end{cases}$ pour $t \in [0,1]$. On décide de calculer de manière approchée par une méthode d'Euler la solution. Soit n un entier non nul on pose $y_k = y(\frac{k}{n})$. Parmi les assertions suivantes, lesquelles correspondent bien à un schéma d'Euler explicite pour le problème de Cauchy posé :

A) $y_0 = 1$ et $y_{k+1} = y_k + n.y_k^3, \forall k \in [|0, n-1|]$.

B) $y_0 = 1$ et $y_{k+1} = ny_k + y_k^3, \forall k \in [|0, n-1|]$.

C) $y_0 = 1$ et $y_{k+1} = y_k + \frac{1}{n}.y_k^3, \forall k \in [|0, n-1|]$.

D) $y_0 = 1$ et $y_{k+1} = y_k + n.y_{k+1}^3, \forall k \in [|0, n-1|]$.

Réponse: le pas est de $\frac{1}{n}$ $(= \frac{k+1}{n} - \frac{k}{n})$. Donc la méthode d'Euler conduit à la récurrence **C)**.

$\boxed{\textbf{Question 18}}$ On modélise la vitesse de la chute d'un grêlon entre les temps $t = 0s$ et $t = 120s$ par l'équation différentielle suivante :

$$(E) : v' = g - \frac{\lambda}{m} v^2,$$

où m désigne la masse du grêlon et λ le coefficient de frottement fluide de l'air.

La méthode d'Euler (explicite) consiste à calculer des approximations v_k de $v(t_k)$ (pour $k \in [\![0, n]\!]$), où $(t_0, \ldots, t_n)$ est une discrétisation régulière de l'intervalle de temps $[0, 120]$ de pas $h = \frac{120}{n}$ (avec $n \in \mathbb{N}^\star$).

Parmi les affirmations suivantes, indiquez celle ou celles qui sont vraies.

A) Il s'agit de résoudre une équation différentielle linéaire d'ordre 1.

B) La méthode de Newton est plus efficace pour résoudre ce type de problèmes que la méthode d'Euler explicite.

C) Les v_k satisfont la récurrence : $\forall k \in [\![0, n-1]\!], v_{k+1} = v_k + g - h\frac{\lambda}{m} v_k^2$.

D) Les v_k satisfont la récurrence : $\forall k \in [\![0, n-1]\!], v_{k+1} = v_k + h\left(g - \frac{\lambda}{m} v_k^2\right)$.

Réponse:

L'équation n'est pas linéaire, à cause du v^2.

La méthode de Newton ne permet pas de résoudre une équation différentielle, contrairement à la méthode d'Euler.

La bonne équation de récurrence est la **D)**.