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:

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

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

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:

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

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

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 $h = 0.01$.

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 $h = 0.01$.

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 les approximations obtenues?

Exemple d'équation différentielle non linéaire

Écrire en Python la méthode d'Euler sur $[-2, 2]$, avec un pas 0.005, appliquée au problème de Cauchy:

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

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

```
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

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

scipy.integrate

Le module `scipy.integrate` contient une fonction `odeint` ayant comme arguments:

1. une fonction `f` à 2 variables
2. une valeur initiale `y0`
3. un tableau `t`

telle que `odeint(f, y0, t)` renvoie un tableau contenant des approximations aux temps t de la solution de:

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

scipy.integrate

Le module `scipy.integrate` contient une fonction `odeint` ayant comme arguments:

1. une fonction `f` à 2 variables
2. une valeur initiale `y0`
3. un tableau `t`

telle que `odeint(f, y0, t)` renvoie un tableau contenant des approximations aux temps `t` de la solution de:

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

Attention: l'ED est écrite sous la forme $y'(t) = f(y(t), t)$ et non pas $y'(t) = f(t, y(t))$.

Étude qualitative d'ED (oral Centrale)

On veut connaître le comportement en $+\infty$ des solutions de l'ED:

$$y'(t) = t^3 - y(t)^3$$

Cette ED possède une infinité de solutions (une pour chaque valeur initiale).
En afficher quelques-unes, en utilisant `scipy.integrate.odeint`.

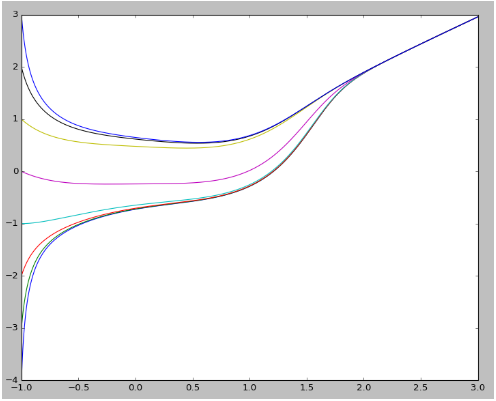
Étude qualitative d'ED (oral Centrale)

```
from scipy.integrate import *

def f(a, b):
    return b**3 - a**3

t = np.arange(-1, 3, 0.001)
for i in range(-4, 4):
    y = odeint(f, i, t)
    plt.plot(t, y)
plt.show()
```

Étude qualitative d'ED (oral Centrale)



Que peut-on conjecturer?

Étude qualitative d'ED (oral Centrale 2)

On veut connaître le domaine de définition de la solution du problème de Cauchy:

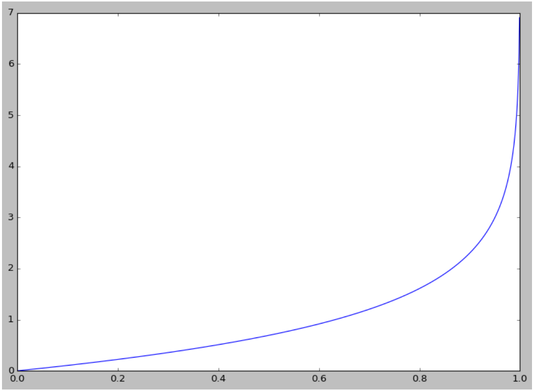
$$\left\{ \begin{array}{l} y'(t) = e^{y(t)} \\ y(0) = 0 \end{array} \right.$$

Étude qualitative d'ED (oral Centrale 2)

```
def f(a, b):
    return np.exp(a)

t = np.arange(0, 1, 0.001)
plt.plot(t, odeint(f, 0, t))
plt.show()
```

Étude qualitative d'ED (oral Centrale 2)



Que peut-on conjecturer?

Méthode d'Euler vectorielle

Informatique pour tous

Résumé de la méthode d'Euler

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

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

La méthode d'Euler, 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 $y_k \approx y(t_k)$.

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 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 $(t_0, \dots, t_n)$ 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 : $\forall k \in [\![0, n-1]\!], v_{k+1} = v_k + h\left(g - \frac{\lambda}{m}v_k^2\right)$.

Système d'équations différentielles

On veut maintenant résoudre un système d'équations différentielles sur $[a, b]$ (avec plusieurs fonctions inconnues y, z):

$$\left\{ \begin{array}{l} y'(t) = 4y(t) - 2z(t) \\ z'(t) = y(t) + 3z(t) \end{array} \right.$$

Comment approximer $y(t)$ et $z(t)$?

Système d'équations différentielles

On veut maintenant résoudre un système d'équations différentielles sur $[a, b]$ (avec plusieurs fonctions inconnues y, z):

$$\begin{cases} y'(t) = 4y(t) - 2z(t) \\ z'(t) = y(t) + 3z(t) \end{cases}$$

Comment approximer $y(t)$ et $z(t)$?

On peut appliquer la méthode sur chaque équation, en calculant des approximations y_k et z_k ($y_k \approx y(t_k)$ et $z_k \approx z(t_k)$).

Vectorialisation

$$\begin{cases} y'(t) = 4y(t) - 2z(t) \\ z'(t) = y(t) + 3z(t) \end{cases}$$

On peut aussi **vectorialiser** en posant $Y(t) = \begin{pmatrix} y(t) \\ z(t) \end{pmatrix}$.

Vectorialisation

$$\begin{cases} y'(t) = 4y(t) - 2z(t) \\ z'(t) = y(t) + 3z(t) \end{cases}$$

On peut aussi **vectorialiser** en posant $Y(t) = \begin{pmatrix} y(t) \\ z(t) \end{pmatrix}$.

$$Y'(t) = \begin{pmatrix} y'(t) \\ z'(t) \end{pmatrix} = \begin{pmatrix} 4 & -2 \\ 1 & 3 \end{pmatrix} Y(t)$$

On est revenu à une ED du 1er ordre donc on peut appliquer la méthode d'Euler!

Méthode d'Euler **vectorielle**

Soit $F : U \subseteq \mathbb{R} \longrightarrow \mathbb{R}^n$ C^1, on considère le problème suivant, où $Y(t) \in \mathbb{R}^n$, $\forall t$:

$$\begin{cases} Y'(t) = F(t, Y(t)) \\ Y(t_0) = Y_0 \end{cases}$$

La méthode d'Euler vectorielle, de pas h, consiste à approximer la solution Y par une suite $(Y_k)_{0 \leq k < n}$ de **vecteurs** de $\mathbb{R}^n$ telle que:

1. $Y_0 = Y(t_0)$
2. $Y_{k+1} = Y_k + h \times F(t_k, Y_k)$

Vectorialisation

$$Y'(t) = \begin{pmatrix} y'(t) \\ z'(t) \end{pmatrix} = \begin{pmatrix} 4 & -2 \\ 1 & 3 \end{pmatrix} Y(t)$$

Équation de récurrence de la méthode d'Euler vectorielle, avec $Y_k = \begin{pmatrix} y_k \\ z_k \end{pmatrix}$:

$$Y_{k+1} = Y_k + h \times \begin{pmatrix} 4 & -2 \\ 1 & 3 \end{pmatrix} Y_k$$

Vectorialisation

Écrit différemment, avec $Y_k = \begin{pmatrix} y_k \\ z_k \end{pmatrix}$:

$$\begin{cases} y_{k+1} = y_k + h \times (4y_k - 2z_k) \\ z_{k+1} = z_k + h \times (y_k + 3z_k) \end{cases}$$

Question

En déduire un algorithme en Python pour approximer les solutions sur $[0, 1]$ (avec un pas de 0.1) de:

$$\begin{cases} y'(t) = 4y(t) - 2z(t) \\ z'(t) = y(t) + 3z(t) \\ y(0) = 3 \\ z(0) = 7 \end{cases}$$

Cinétique chimique

Soient deux réactions chimiques d'ordre 1:

$$A \xrightarrow{\alpha} B$$

$$B \xrightarrow{\beta} C$$

On veut connaître les concentrations au cours du temps.

Cinétique chimique

Soient deux réactions chimiques d'ordre 1:

$$A \xrightarrow{\alpha} B$$

$$B \xrightarrow{\beta} C$$

On veut connaître les concentrations au cours du temps.

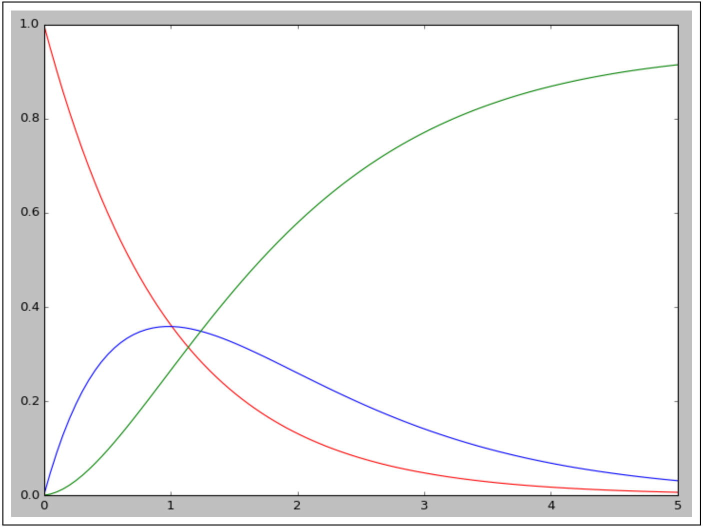
$$\begin{cases} \dfrac{d[A]}{dt} = -\alpha[A] \\ \dfrac{d[B]}{dt} = \alpha[A] - \beta[B] \\ \dfrac{d[C]}{dt} = \beta[B] \end{cases}$$

Cinétique chimique

$$\begin{cases} \dfrac{d[A]}{dt} = -\alpha[A] \\ \dfrac{d[B]}{dt} = \alpha[A] - \beta[B] \\ \dfrac{d[C]}{dt} = \beta[B] \end{cases}$$

Écrire en Python la méthode d'Euler pour $t \in [0, 5]$, avec $[A]_0 = 1$ et $[B]_0 = [C]_0 = 0$.

Cinétique chimique



Equation différentielle d'ordre 2

Comment appliquer la méthode d'Euler sur une équation différentielle d'ordre ≥ 2?
Par exemple (équation du pendule linéarisé):

$$\theta''(t) = -\theta(t)$$

Equation différentielle d'ordre 2

Comment appliquer la méthode d'Euler sur une équation différentielle d'ordre ≥ 2?
Par exemple (équation du pendule linéarisé):

$$\theta''(t) = -\theta(t)$$

On pose $z(t) = \theta'(t)$.

Equation différentielle d'ordre 2

Comment appliquer la méthode d'Euler sur une équation différentielle d'ordre ≥ 2?
Par exemple (équation du pendule linéarisé):

$$\theta''(t) = -\theta(t)$$

On pose $z(t) = \theta'(t)$.
On a alors $z'(t) = \theta''(t) = -\theta(t)$

Equation différentielle d'ordre 2

Comment appliquer la méthode d'Euler sur une équation différentielle d'ordre ≥ 2?
Par exemple (équation du pendule linéarisé):

$$\theta''(t) = -\theta(t)$$

On pose $z(t) = \theta'(t)$.
On a alors $z'(t) = \theta''(t) = -\theta(t)$
On obtient donc un système d'équa. diff. linéaires d'ordre 1:

$$\begin{cases} \theta'(t) = z(t) \\ z'(t) = -\theta(t) \end{cases}$$

Equation différentielle d'ordre 2

Comment appliquer la méthode d'Euler sur une équation différentielle d'ordre ≥ 2?
Par exemple (équation du pendule linéarisé):

$$\theta''(t) = -\theta(t)$$

On pose $z(t) = \theta'(t)$.
On a alors $z'(t) = \theta''(t) = -\theta(t)$
On obtient donc un système d'équa. diff. linéaires d'ordre 1:

$$\begin{cases} \theta'(t) = z(t) \\ z'(t) = -\theta(t) \end{cases}$$

On peut alors appliquer la méthode d'Euler pour obtenir des approximations $\theta_k \approx \theta(t_k)$ et $z_k \approx \theta'(t_k)$.

Équation différentielle d'ordre 2

$$\theta''(t) = -\theta(t)$$

On peut aussi vectorialiser en posant $\Theta(t) = \begin{pmatrix} \theta(t) \\ \theta'(t) \end{pmatrix}$.

Équation différentielle d'ordre 2

$$\theta''(t) = -\theta(t)$$

On peut aussi vectorialiser en posant $\Theta(t) = \begin{pmatrix} \theta(t) \\ \theta'(t) \end{pmatrix}$.

Alors $\Theta'(t) = \begin{pmatrix} \theta'(t) \\ \theta''(t) \end{pmatrix} = \begin{pmatrix} \theta'(t) \\ -\theta(t) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} \theta(t) \\ \theta'(t) \end{pmatrix}$.

On s'est ramené à une équa diff du 1er ordre: $\Theta'(t) = A\ \Theta(t)$.

Equation différentielle d'ordre 2

$$\Theta'(t) = \begin{pmatrix} \theta'(t) \\ \theta''(t) \end{pmatrix} = \begin{pmatrix} \theta'(t) \\ -\theta(t) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} \theta(t) \\ \theta'(t) \end{pmatrix}.$$

On s'est ramené à une ED $\Theta'(t) = A\,\Theta(t)$.

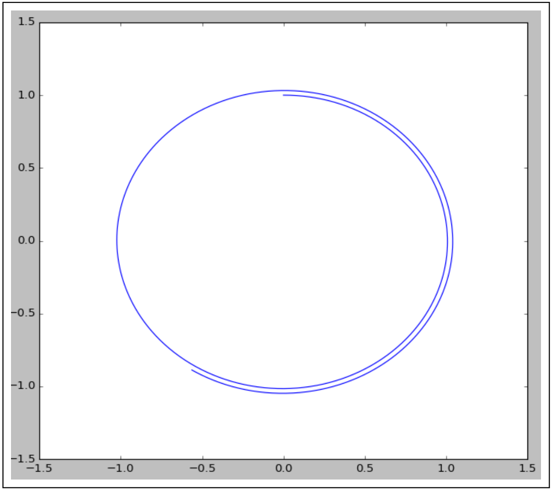
Les approximations de la méthode d'Euler sont donc des vecteurs Θ_k vérifiant:

$$\Theta_{k+1} = \Theta_k + h \times \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \Theta_k$$

Pendule linéarisé

```
Theta = [0]
Theta_p = [1]
for k in range(999):
    Theta.append(Theta[k] + 0.01*Theta_p[k])
    Theta_p.append(Theta_p[k] - 0.01*Theta[k])
plt.plot(Theta, Theta_p)
plt.show()
```

Portrait de phase du pendule linéarisé



Pendule non linéarisé

Équation du pendule non linéarisé:

$$\theta''(t) = -\sin(\theta(t))$$

Pendule non linéarisé

Équation du pendule non linéarisé:

$$\theta''(t) = -\sin(\theta(t))$$

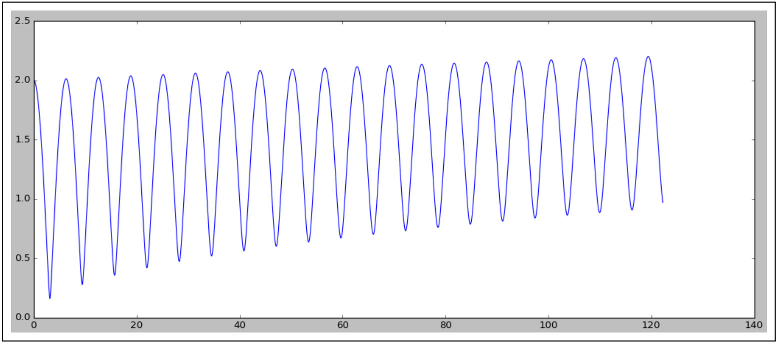
On pose $\Theta(t) = \begin{pmatrix} \theta(t) \\ \theta'(t) \end{pmatrix}$.

Alors $\Theta'(t) = \begin{pmatrix} \theta'(t) \\ \theta''(t) \end{pmatrix} = \begin{pmatrix} \theta'(t) \\ -\sin(\theta(t)) \end{pmatrix}$.

Pendule non linéarisé

```
Theta = [0]
Theta_p = [2]
for k in range(9999):
    Theta.append(Theta[k] + 0.01*Theta_p[k])
    Theta_p.append(Theta_p[k] - 0.01*np.sin(Theta[k]))
plt.plot(Theta, Theta_p)
plt.show()
```

Pendule non linéarisé



Implémentation de la méthode d'Euler vectorielle générale

Écrire une fonction ayant en argument une fonction vectorielle F, des valeurs initiales $t0$, $Y0$, un pas h, un nombre d'itérations n et renvoyant la suite des t_k et des Y_k de la méthode d'Euler vectorielle appliquée à:

$$\begin{cases} Y'(t) = F(t, Y(t)) \\ Y(t0) = Y0 \end{cases}$$

Implémentation de la méthode d'Euler vectorielle générale

```python
def euler(F, t0, Y0, h, n):
    t = [t0]
    Y = [Y0]
    for k in range(n):
        Y.append(Y[k] + h * F(t[k], Y[k]))
        t.append(t[k] + h)
    return [t, Y]
```

Implémentation de la méthode d'Euler vectorielle générale

```
def euler(F, t0, Y0, h, n):
    t = [t0]
    Y = [Y0]
    for k in range(n):
        Y.append(Y[k] + h * F(t[k], Y[k]))
        t.append(t[k] + h)
    return [t, Y]
```

```
def pendule(a, b):
    return np.array([b[1], -np.sin(b[0])])

r = euler(pendule, 0, np.array([0, 1]), 0.1, 1000)
```