Résolution numérique d'équations différentielles d'ordre 1

- Méthode d'EULER - HEUN - RK4 -

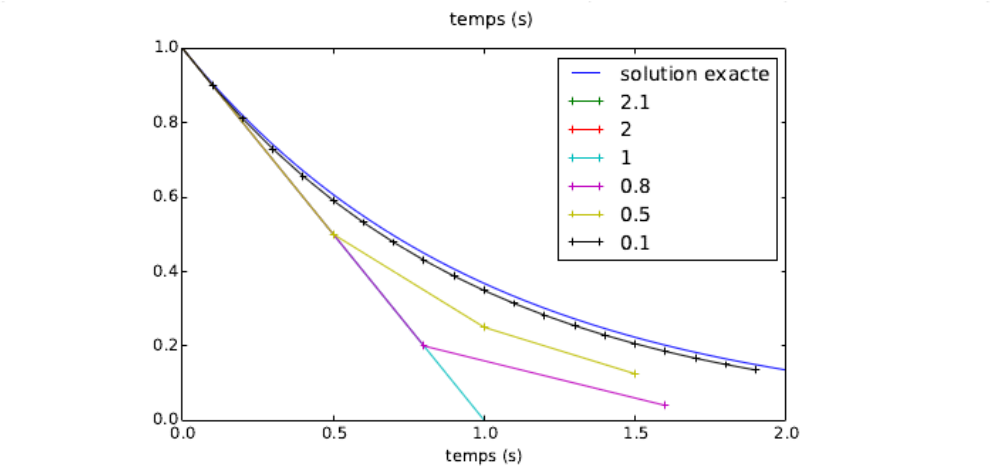


FIGURE 1 – Solution de l'application pour le schéma Euler explicite.

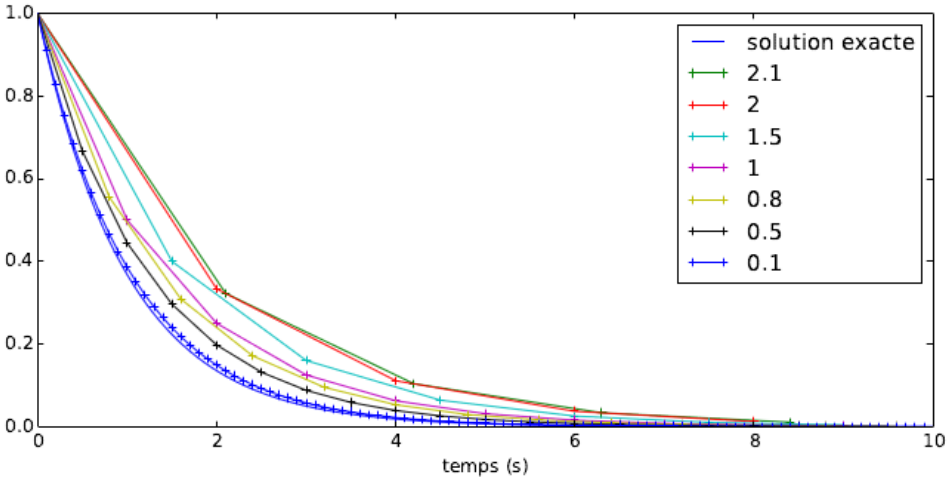


FIGURE 2 – Solution de l'application pour le schéma Euler implicite.

Résolution par EULER d'équations différentielles d'ordre 1

1. Rappel sur l'équation différentielle scalaire d'ordre 1 – problème de Cauchy

Définition de l'équation différentielle, relation entre y' et y :

$$\{P\} \begin{cases} y' = f(y(t),t) \text{ pour } \forall t \in [a,b] \\ y(a) = y_0 \end{cases}$$

 # on a une equ diff

 # et une C.I

Exemple :

① $y' + y = e^t$ *avec y dépend de t*

$\Leftrightarrow y' = -y + e^t$

$\Leftrightarrow y' = f(y,t)$ où $\boxed{f(y,t) = -y + e^t}$

② $y' + ty^2 = \sin(t)$

$\Leftrightarrow y' = -ty^2 + \sin(t)$

$\Leftrightarrow y' = f(y,t)$ où $\boxed{f(y,t) = -ty^2 + \sin(t)}$

Remarques :

Une équation différentielle est une relation entre une fonction, inconnue *y* de *x* qu'il s'agit de déterminer, et ses dérivées y', y'' ... $y^{(n)}$. Une équation différentielle du premier ordre est une équation différentielle ne faisant intervenir que la dérivée première.

Résoudre une équation différentielle revient à **trouver une fonction *y(t)* dont les dérivées sont solutions de l'équation**.

{P} admet une unique solution sous certaines hypothèses (système déterministe). Il n'est pas toujours possible de trouver une expression analytique formule pour la solution de {P}.
Donc, il faut parfois mettre en place des **méthodes numériques** pour les résoudre.

2. Résolution numérique – algorithme d'EULER

- h >0 est le pas de discrétisation avec $\boxed{\mathbf{h=(b-a)/n}}$ où n est le nombre de points de calcul

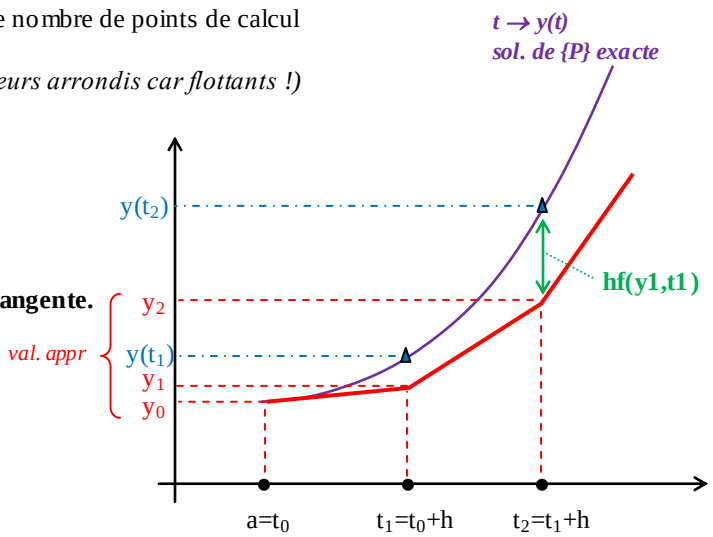
  (plus n est grand plus on est précis…mais attention aux erreurs arrondis car flottants !)

- y_0 est une C.I
- $t_0 = a$, on connait $y(t_0)=y(a) = y_0$

Quel est le principe de la méthode d'EULER ?

Idée **EULER** : sur $[t_0,t_0+h]$, on **remplace la solution par sa tangente.**

rq : - *si on veut valeur entre 2 points → on interpole*
- *pour améliorer précision, on réduit le pas h*
- *les valeurs $y(t_i)$ on ne les a pas, mais on peut avoir les valeurs approchées par les tangentes y_i*
- *la ligne brisée reliant les points (t_i,y_i) est appelée **polygône d'EULER.***



Résolution par EULER d'équations différentielles d'ordre 1

Mais comment obtenir y_1 tel que $y(t_1) \approx y_1$?

je l'ai *y' donnée par y'=f(y(t),t)*

$\forall t \in [t_1,t_2]$, l'expression de la tangente est : $l(t) = y(t_0) + y'(t_0)(t-t_0)$ *# l comme ligne brisée*

$= y_0 + f(y(t_0),t_0)(t-t_0)$ *car y sol. de {P}*

et $y_1 = l(t_1) = y_0 + f(y_0,t_0)(t_1-t_0)$

$\Leftrightarrow$ **$y_1 = y_0 + hf(y_0,t_0)$** *# pente de la droite passant de y_0 à y_1*

Si je connais la valeur à t_0, je peux calculer la valeur approchée à t_1.

Comment faire pour calculer une approximation de y(t2) tel que $y(t2) \approx y_2$?

On peut recommencer …

$\forall t \in [t_1,t_2]$, $l_1(t) = y_1 + y'(t_1)(t-t_1)$

$= y_1 + f(y(t_1),t_1)(t-t_1)$

Pour $t=t_2$: $l_1(t_2) = y_1 + f(y(t_1),t_1)(t_2-t_1)$ *# ce n'est pas très rigoureux mathématiquement mais un moyen d'avoir le résultat*

 # je veux la relation $y_{n+1} = f(y_n)$

$\Leftrightarrow$ **$y_2 = y_1 + hf(y_1,t_1)$**

Ainsi de suite jusqu'à t=b.

La méthode d'EULER est donc la suivante:

$t_0 = a$

$t_{n+1} = t_n + h$ $\quad \forall n \in N$, **$y_{n+1} = y_n + hf(y_n,t_n)$**

$y_0 = y_0$ *(C.I)*

 # qd h→0: le polygône d'Euler approche de mieux en mieux la sol

Algorithme d'EULER

On a besoin de **2 suites** construites par récurrence (le **pas de discrétisation** du temps et **l'approximation des valeurs**).

Entrées : a, b, y_0, f # f : fonction qui dépend de l'équation diff.
a, b : bornes de l'intervalle de recherche

h ← (b-a)/2 # pas constant de discrétisation

$t_0 \leftarrow a$

$y_0 \leftarrow y_0$

pour i allant de 1 à n-1 faire # attention range Python !

$t_{i+1} \leftarrow t_i + h$

$y_{i+1} \leftarrow y_i + hf(y_i,t_i)$

Sortie : t, y

Résolution par EULER d'équations différentielles d'ordre 1

Dans le TP, pour programmer cet algorithme, vous devrez **créer 2 listes**:

- t: liste des t_i, par défaut initialisée à 0, discrétisation de l'intervalle [a,b]
- y: liste des y_i, par défaut initialisée à 0, approximation de la solution de {P}

3. Précisions par rapport au sujet de TP

Dans le TP, on va voir **d'autres méthodes de résolution numérique**. On va analyser et comparer leur comportements en fonction de h. On va pouvoir observer l'évolution de **l'erreur en fonction de h** *(qui dépend aussi performances de la machine !)*

On définit l'erreur par: $E(h) = \max | y(t_i) - y_i |$ *avec y sol de {P} et y0,...,y_n sol approchées*

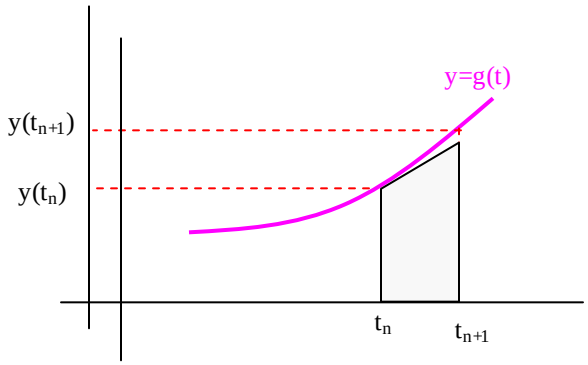
On dit que la méthode est d'**ordre** au moins p si $E(h) \leq A.h^p$
la méthode **d'EULER est d'ordre 1 car E(h) = O(h) donc linéaire**

On va essayer d'observer l'ordre des méthodes et de les comparer (**EULER, HEUN, RK4**)
On comprend que si l'ordre est grand, c'est bien, car augmente la **précision.**

3.1. Méthode de HEUN (ordre 2)

Avec Heun, j'intègre de t_n à t_{n+1}.

$$\int y'(t)\,dt = \int f(y(t),t)\,dt$$
$$\Leftrightarrow y(t_{n+1}) - y(t_n) = \int f(y(t),t)\,dt$$
$$\Leftrightarrow \boxed{y(t_{n+1}) = y(t_n) + \int f(y(t),t)\,dt} \quad (1)$$ *# ressemble euler*



On va approcher $\int$ par la **méthode des trapèzes**

$$\int g(t)\,dt = h*\frac{g(t_n)+g(t_{n+1})}{2}$$

donc (1) $\Leftrightarrow y(t_{n+1}) = y(t_n) + (h/2)(f(y(t_n),t_n) + f(y(t_{n+1}),t_{n+1})$

on fait un peu d'Euler

$$= y(t_n) + (h/2)(f(y(t_n),t_n) + f(y(t_n) + hf(y(t_n),t_n),t_{n+1})$$

Et on retrouve la formule du TP:

$$\boxed{y_{n+1} = y_n + (h/2)(f(y_n,t_n) + f(y_n + hf(y_n,t_n),t_{n+1})}$$

 # on fait simplifié écriture dans python : $y(t_n) \rightarrow y_n$

Résolution par EULER d'équations différentielles d'ordre 1

3.2. Méthode de RK4 : Runge Kutta (ordre 4)

Avec Heun, on a remplacé les rectangles *(fonction cte)* par des trapèzes *(polynôme 1er ordre)* , que puis je faire d'autre ? Et bien passer par un polynôme du 2nd degré...

Attention, avec un degré 2 il faut 3 coefficients (pt milieu). Je cherche la parabolle qui passe par ces 3 points $\Rightarrow$ approche aire au mieux.

On arrive à la formule du TP:

$$y_{n+1} = y_n + (h/6)(f(y_n,t_n) + 2f(\alpha_n,t_n + h/2) + 2f(\beta_n,t_n + h/2) + f(\gamma_n, t_{n+1}))$$