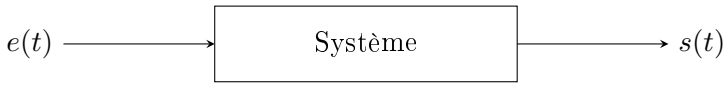
Dans ce TP, on présente la méthode de Euler qui permet de donner une solution approximative des équations différentielles.

1 Problèmes

On considère un système (par exemple un circuit électronique) recevant $e(t)$ en entrée et produisant $s(t)$ en sortie.



Le système obéit à une certaine équation différentielle.

$$(\mathcal{E}) : \quad s' + a \cdot s = e(t)$$

On sait également que $s(0) = 0$.

2 Principe

On sait $s'(t) = \frac{\mathrm{d}s}{\mathrm{d}t}$ avec $\mathrm{d}t$. Si $\mathrm{d}t$ est petit – au lieu d'être infinitésimal – cette égalité devient une approximation. On aura donc la variation $\mathrm{d}s \simeq s'(t) \cdot \mathrm{d}t$.

On se fixera donc un $\mathrm{d}t$ assez petit pour que l'approximation soit correcte.

— On connaît $s(t)$ et $e(t)$ à un instant t_0,
— en utilisant $\mathcal{E}$ on en déduit $s'(t_0)$,
— on calcule $s(t_0 + \mathrm{d}t) = s(t_0) + s'(t_0) \cdot \mathrm{d}t$

Ainsi, de proche en proche, on peut calculer tous les $s(t)$ avec $t \in [0; T]$.

En général, au lieu de fixer $\mathrm{d}t$ on préfère :

— fixer la durée T que l'on souhaite simuler,
— fixer un nombre N assez grand et poser $\mathrm{d}t = \frac{T}{N}$. Ainsi N représente le nombre de points que l'on obtiendra sur la courbe.

3 Code Python

(a) On commence par importer le module graphique

```python
import matplotlib.pyplot as plt # graphique
```

(b) On définit T et N ce qui permet de calculer $\mathrm{d}t$.

```python
T = 10
N = 1000
dt = T/N
```

(c) On fournit les paramètres de l'équation : il faut indiquer la valeur de a et l'expression de $e(t)$. Pour $e(t)$ on définit une fonction.

Pour l'exemple, on prendra une fonction $e(t) = \begin{cases} 0 \text{ si } t < 0 \\ 5 \text{ si } 0 \leqslant t \end{cases}$

```python
a = 0.3
def e(t):
    if t<0:
        return 0
    else:
        return 5
```

(d) On commence par initialiser des variables représentant le temps, e et s.

```
Py
t_actuel = 0
e_actuel = e(0)
s_actuel = 0
```

(e) On souhaite conserver les valeurs calculées dans des tableaux pour pouvoir les afficher ensuite. On amorce donc un tableau pour le temps, pour e et pour s.

```
Py
valeurs_t = [0]
valeurs_e = [e(0)]
valeurs_s = [0]
```

Les trois tableaux ne contiennent pour l'instant qu'un élément. On va les compléter progressivement.

(f) On a dit que l'on voulait faire N pas de temps dt. Nous allons faire une boucle répétitive :

```
Py
for i in range(N): # répète N fois
    s_deriv = e_actuel - a*s_actuel
    # calcul de nouvelles valeurs
    t_actuel += dt # += signifie augmente de ...
    e_actuel = e(t_actuel)
    s_actuel += s_deriv*dt
    # stockage de nouvelles valeurs dans les tableaux respectifs
    # append : ajoute au bout du tableau
    valeurs_t.append(t_actuel)
    valeurs_e.append(e_actuel)
    valeurs_s.append(s_actuel)
```

(g) Il ne reste qu'à représenter les courbes.

```
Py
plt.plot(valeurs_t, valeurs_e, 'r', label='entrée')
plt.plot(valeurs_t, valeurs_s, 'b', label='sortie')
plt.legend()
plt.show()
```