L'essentiel : Résolution des Systèmes Différentiels

E est un $\mathbb{K}$-ev de dimension finie de dimension n.
L'objet de ce chapitre est la résolution des *Equations Différentielles Linéaires Vectorielles*.

- Forme 1 : Une telle équation est de la forme :

$$\boxed{x' = a(t)(x) + b(t)} \quad \text{avec} \quad \begin{cases} t, \text{ la variable, appartenant à un intervalle } I \subset \mathbb{R} \\ x, \text{ l'inconnue, une fonction de } I \to E \\ a, \text{ une fonction } \mathcal{C}^0 \text{ de } I \to \mathcal{L}(E) \end{cases} \quad \text{et } b \in \mathcal{C}^0(I, E)$$

- Forme 2 : En se plaçant dans une base de E, l'équation différentielle précédente prend la forme :

$$\boxed{X' = A(t)X + B(t)} \quad \text{avec} \quad \begin{cases} X(t) = \begin{pmatrix} x_1(t) \\ \vdots \\ x_n(t) \end{pmatrix} \text{ dérivable sur } I \\ \\ A(t) = \begin{pmatrix} a_{1,1}(t) & \dots & a_{1,n}(t) \\ \vdots & & \vdots \\ a_{n,1}(t) & \dots & a_{n,n}(t) \end{pmatrix} \mathcal{C}^0 \text{ sur } I \end{cases} \quad \text{et } B(t) = \begin{pmatrix} b_1(t) \\ \vdots \\ b_n(t) \end{pmatrix} \in \mathcal{C}^0(I, \mathfrak{M}_{n,1}(\mathbb{K}))$$

  C'est en général sous cette forme que l'on traite le problème.

- Forme 3 : Développé, ce système prend alors la forme d'un *Système Différentiel* :

$$\boxed{\begin{cases} x_1' = a_{1,1}(t)x_1 + \cdots + a_{1,n}(t)x_n + b_1(t) \\ \vdots \\ x_n' = a_{n,1}(t)x_1 + \cdots + a_{n,n}(t)x_n + b_n(t) \end{cases}}$$

Théorème : **Cauchy linéaire**

Le système $X' = A(t)X + B(t)$ admet une unique solution X vérifiant $X(t_0) = X_0$, où $\begin{cases} t_0 \in I \\ X_0 \in \mathfrak{M}_{n,1}(\mathbb{K}) \end{cases}$

I] Système Différentiel à coefficients quelconques

Méthode Générale de Résolution

1. On résout $H : X' = A(t)X$ le système homogène associé
2. On recherche une solution particulière $\tilde{X}$
3. Les solutions sont alors les fonctions :
$$X = X_H + \tilde{X}$$

Eléments de résolution de : $X' = A(t)X$

- L'ensemble des solutions est un sev de $\mathcal{C}^1(I,\ \mathfrak{M}_{n,1}(\mathbb{K}))$ de dimension n.
  Il « suffit » donc de rechercher n solutions indépendantes appelées *Système Fondamental de Solutions*

- On peut trouver des solutions en déterminant les éléments propres de $A(t)$.
  Si V est un vecteur propre constant alors on peut rechercher une solution sous la forme :

$$X(t) = \alpha(t)V$$

  Cette méthode n'est pas explicitement à connaître mais peut-être utilisée

- Lorsque $A(t)$ est diagonalisable ou trigonalisable avec une matrice de passage P indépendante de t, on peut se ramener à un nouveau système différentiel plus simple à résoudre en posant $Y = P^{-1}X$.

Recherche d'une solution particulière $\tilde{X}$ de : $X' = A(t)X + B(t)$

On applique la méthode de variation des constantes avec $(X_1,\ \dots,\ X_n)$ un SFS :

- On part de la solution générale de H : $\quad X_H(t) = \lambda_1 X_1(t) + \cdots + \lambda_n X_n(t)$.

- On recherche $\tilde{X}$ sous la forme $\tilde{X}(t) = \lambda_1(t)X_1(t) + \cdots + \lambda_n(t)X_n(t)$, sachant que :

$$\tilde{X} \text{ est solution} \quad \Longleftrightarrow \quad \lambda_1'(t)X_1(t) + \cdots + \lambda_n'(t)X_n(t) = B(t)$$

Dans le cas où $n = 2$, on pensera à appliquer les formules de Cramer pour déterminer les $\lambda_k'(t)$.

II] Système différentiel à coefficients constants

On s'intéresse ici aux systèmes de la forme : $\quad X' = AX + B(t) \quad$ où $\quad A \in \mathfrak{M}_n(\mathbb{K})$.

La méthode générale précédente s'applique.
La difficulté est donc de rechercher un système fondamental de solutions.

Théorème : Les solutions de $X' = AX$ sont les fonctions : $\quad X(t) = \exp(tA).X_0 \quad$ où $\quad X_0 \in \mathfrak{M}_{n,1}(\mathbb{K})$.

On ne peut appliquer ce résultat que si $\exp(tA)$ *se calcule facilement...*
Sinon :

- On peut diagonaliser :

Recherche d'un SFS par diagonalisation :

- Résultat à connaître : Si V est un vecteur propre de A associé à la valeur propre λ, alors :

$$X(t) = e^{\lambda t}V \quad \text{est une solution de} \quad X' = AX$$

- Lorsque A est diagonalisable, la solution générale de $X' = AX$ est alors :

$$X_H(t) = \alpha_1 e^{\lambda_1 t}V_1 + \cdots + \alpha_n e^{\lambda_n t}V_n \quad \text{où} \quad (V_1,\ \dots,\ V_n) \text{ est une base de vecteurs propres}$$

- On peut trigonaliser :

Par changement de base, on se ramène alors à un système triangulaire que l'on peut résoudre en partant de la dernière équation...

III] Equations linéaires scalaires d'ordre n

Il s'agit maintenant de résoudre :

$$(E) : x^{(n)} = a_{n-1}(t)x^{(n-1)} + \cdots + a_o(t)x + b(t) \quad \text{avec} \left\{ \begin{array}{l} a_0, \ \ldots, \ a_{n-1} : I \to \mathbb{K} \\ b : I \to \mathbb{K} \end{array} \right. \quad \text{continues}$$

Le dernier chapitre de l'année sera consacré à la résolution de ces équations.
On peut cependant d'ores et déjà retenir que :

- L'ensemble des solutions de $(H) : x^{(n)} = a_{n-1}(t)x^{(n-1)} + \cdots + a_o(t)x$ est un $\mathbb{K}$-ev de dimension n
- Il existe une unique solution de (E) vérifiant des conditions initiales de la forme : $\left\{ \begin{array}{l} x(t_0) = x_0 \\ \vdots \\ x^{(n-1)}(t_0) = x_{n-1} \end{array} \right.$
- Les solutions générales sont de la forme $x = \tilde{x} + x_H$ où $\tilde{x}$ est une solution particulière.
- On peut résoudre (E) en se ramenant au système différentiel :

$$\left\{ \begin{array}{lcl} x_1' & = & x_2 \\ x_2' & = & x_3 \\ \vdots & & \\ x_{n-1}' & = & x_n \\ x_n' & = & a_{n-1}(t)x_n + \cdots + a_o(t)x_1 + b(t) \end{array} \right.$$

Les solutions x_1 sont alors les solutions de (E).

IV] Résolution numérique

Exemple : Pour résoudre numériquement $\left\{ \begin{array}{l} x' = 2tx + t^2y + \cos(t) \\ y' = t^2x + 2ty + \sin(t) \end{array} \right.$ pour $t \in [t_0, \ t_N]$ de CI $\left\{ \begin{array}{l} x(t_0) = -1 \\ y(t_0) = -1 \end{array} \right.$.

Méthode 1 : Avec l'algorithme d'Euler

On définit le système différentiel par deux fonctions : $\left\{ \begin{array}{l} f_1(x, y, t) = 2tx + t^2y + \cos(t) \\ f_2(x, y, t) = t^2x + 2ty + \sin(t) \end{array} \right.$.

- On discrétise l'intervalle de temps $[t_0, \ t_N]$: $\quad t_n = t_0 + np \quad$ avec $\quad p = \frac{t_1 - t_0}{N}$.
- On construit les suites $\left\{ \begin{array}{l} (x_n) \\ (y_n) \end{array} \right.$ définies par :

$$\left\{ \begin{array}{l} x_0 \\ y_0 \end{array} \right. \quad \text{et pour tout } n \in \mathbb{N}^* \quad \left\{ \begin{array}{l} x_{n+1} = x_n + pf_1(x_n, y_n, t_n) \\ y_{n+1} = y_n + pf_2(x_n, y_n, t_n) \end{array} \right.$$

Python

```python
from matplotlib.pylab import plot, cos, sin

f1 = lambda x,y,t : 2*t*x+t**2*y+cos(t)
f2 = lambda x,y,t : 2*t*y+t**2*x+sin(t)

def Euler(a,b,t0,t1,N):
    p = (t1-t0)/N
    x = [a]
    y = [b]
    t = t0
    for k in range(1,N+1):
        t = t + p
        x.append (x[k-1]+p*f1(x[k-1],y[k-1],t))
        y.append (y[k-1]+p*f2(x[k-1],y[k-1],t))
    plot(x,y)

Euler(-1,-1,-1,1,100)
```

Méthode 2 : Avec la fonction Odeint(F,[x0,y0],[t0,t1,t2,...,tN])

On utilise ici la fonction `odeint` de la bibliothèque `scipy.integrate`.

- `F` définit le système différentiel.
  Dans notre exemple, il s'agit de la fonction F :

$$F([x,y],t) = (2tx + t^2y + \cos(t), t^2x + 2ty + \sin(t))$$

- `[x0,y0]` définit les CI, c'est à dire les valeurs de x et y en t_0.
- `[t0,t1,t2,...,tN]` définit l'intervalle de temps discrétisé.
  Il s'agit d'une liste donnant les valeurs discrétisées de l'intervalle de variation de la variable t.

La fonction `odeint` renvoie le tableau de points :

`sol = array([[x0,y0], [x1,y1], ..., [xN,yN]])`

On peut donc tracer la courbe intégrale en extrayant les listes $\left\{ \begin{array}{l} \texttt{x = sol[:,0]} \\ \texttt{y = sol[:,1]} \end{array} \right.$ et en appliquant `plot(x,y)`.

Python

```python
from scipy.integrate import odeint
from matplotlib.pylab import plot, cos, sin

x0 = -1                     # Conditions initiales
y0 = -1

t0 = -1                     # Intervalle de temps
tN = 1

N = 100                     # Division de l'intervalle de temps

F = lambda X,t:(2*t*X[0]+t**2*X[1]+cos(t),
                t**2*X[0]+2*t*X[1]+sin(t))      # Ne pas remplacer X par [x,y]

temps = [t0 + k*p for k in range(N+1)]

sol = odeint(F,[x0,y0],temps)
plot(sol[:,0],sol[:,1])                         # Bien noter la syntaxe !
```