Calcul scientifique en Fortran 90
TP EDO

Préambule

Dans ce TP, on va tester des schémas numériques pour résoudre des systèmes d'équations différentielles autonomes de la forme :

$$\frac{d}{dt}U = F(U), \tag{1}$$

où $U = U(t)$ est le vecteur des inconnues et F une fontion régulière de U. On associe cette équation à une condition initiale $U(0) := U_0$.
On s'intéressera à trois exemples :

Système de Lotka-Volterra
Il s'agit d'un modèle représentant l'évolution d'un écosystème composé de deux espèces : des prédateurs et des proies. Il s'écrit :

$$U = \begin{pmatrix} x \\ y \end{pmatrix}, \; F(U) = \begin{pmatrix} \alpha x - \beta xy \\ \gamma xy - \delta y \end{pmatrix},$$

où x représente la densité des proies, y celle des prédateurs et α, β, γ et δ sont des paramètres positifs qu'on pourra fixer à 1.
On pourra dans ce cas choisir $U_0 = (1, 0.1)^\top$.

Système de Lorenz
Ce système a été proposé par E. Lorenz en 1963 pour résoudre un problème simplifié de climatologie. Les résultats qu'il obtint en firent l'un des systèmes d'EDO les plus célèbres. Il s'agit du système de 3 équations défini par :

$$U = \begin{pmatrix} x \\ y \\ z \end{pmatrix}, \; F(U) = \begin{pmatrix} \sigma(y - x) \\ x(\rho - z) - y \\ xy - \beta z \end{pmatrix},$$

Type	k [kg/m]	m [kg]	Angle de tir	Vitesse init. [m/s]
Canon de 75mm	$2.6E-4$	5.314	$-11 \to 18°$	550
Canon de 155mm	$3.55E-3$	40	$-12 \to 28°$	470
Krupp 420mm "Dicke Bertha"	$2.1E-3$	400	$40 \to 75°$	333
Pariser Kanonen	$1.3E-3$	300	$55°$	$1500 \to 1600$

TABLE 1 – Caractéristiques de canons de la 1ère guerre mondiale

où σ, ρ et β sont des paramètres positifs. On pourra choisir comme Lorenz $\sigma = 10$, $\beta = 8/3$ et $\rho = 28$.
Pour commencer, on pourra utiliser la donnée initiale suivante : $U_0 = (1, 1, 1)^\top$.

Tir balistique
Il s'agit de la trajectoire d'un obus en tenant compte des frottements de l'air. Les équations sont :

$$U = \begin{pmatrix} u \\ w \\ x \\ z \end{pmatrix}, \; F(U) = \begin{pmatrix} -\frac{k}{m}\sqrt{u^2+w^2}u \\ -\frac{k}{m}\sqrt{u^2+w^2}w - g \\ u \\ w \end{pmatrix},$$

où u et w sont les composantes horizontales et verticales de la vitesse et x et z sont la distance horizontale et l'altitude. On prendra $g = 9.81 m.s^{-2}$.
Par ailleurs m est la masse du projectile et k le coefficient de frottement dans l'air. Normalement, k dépend de l'altitude, mais on le supposera constant ici.
Pour les applications numériques, centennaire oblige, on utilisera les caractéristiques de quelques canons emblématiques de la première guerre mondiale résumées dans le tableau 1.

Première partie - schéma d'Euler explicite

Pour rappel, on peut calculer des solutions approchées du système d'EDO (1) de la manière suivante. Pour $\Delta t > 0$ donné, on va calculer les premières valeurs de $(U^n)_n$ une suite d'approximations de la solution telle que $U^n \simeq U(t^n)$ où $t^n = n\Delta t$. Les valeurs de U^n sont définies par le choix d'un schéma numérique.
Dans cette première partie, on va écrire la structure du programme en se focalisant sur le schéma d'Euler explicite :

$$U^{n+1} = U^n + \Delta t F(U^n). \tag{2}$$

1. Dans un module `Edo`, définir les variables globales suivantes : `U` et `params` qui sont des vecteurs de réels au profil dynamique, les entiers `dim` et `nbparams` ainsi que les réels `deltat, tfin`.
2. Écrire dans ce même module une procédure `Init` qui va lire dans un fichier `Parametres.dat` la valeur de `dim, nbparams, deltat, tfin`, allouer `U` de taille `dim` et `params` de taille `nbparams`, puis remplir ces deux vecteurs en utilisant la valeur de U_0 et des paramètres qui doivent également être lus dans le fichier `Parametres.dat`.
3. Écrire dans ce module une fonction `F` qui, à partir d'un vecteur `V` calcule F(`V`) donné par les applications du préambule. On utilisera `Select Case` et on pourra discriminer les différentes applications selon leur dimension (représentée par `dim`). On rajoutera un exemple linéaire de dimension 1 pour vérifier le code.
4. Programmer un module `Schema`, qui doit utiliser `Edo` et qui contient une procedure `Un_pas_temps`. Cette procédure doit calculer la mise à jour de `U` par le schéma d'Euler explicite.
   Explication : en début de procédure `U` contient U^n et en fin de procédure, il contient U^{n+1} donné par la formule (2).
5. Écrire un programme principal `ResEdo`, qui utilise les modules précédents. Il doit :
   - Faire appel à la fonction `Init`,
   - Faire une boucle en temps pour $t = 0$ jusqu'à `tfin` qui appelle `Un_pas_temps`,
   - Ecrire à chaque itération les valeurs de t et de chaque composante de `U` dans un fichier formatté `Resultats.dat`.

   Valider votre programme, puis le faire tourner sur le problème de Lotka-Volterra avec `tfin=30`.
   La théorie prévoit que l'on doit trouver une solution périodique. Que penser des résultats numériques que l'on vient d'obtenir ?
6. Avec cette méthode, le temps d'exécution est très ralenti par le fait de laisser ouvert le fichier `Resultats.dat` et d'écrire dedans à chaque itération. Pour remédier à celà, on utilisera un tableau dynamique `W` de taille `dim x nmax` où `nmax` est le nombre d'éléments de la suite $(U^n)_n$ calculés (y compris U_0 !). Faire les modifications dans votre code pour utiliser ce tableau `W` et n'écrire le fichier `Resultats.dat` qu'une fois les calculs terminés.

Deuxième partie - méthodes de Runge-Kutta explicites

La méthode d'Euler explicite n'est que d'ordre 1, on va améliorer la précision

en utilisant des méthodes d'ordre plus élevé. Pour celà, on se place dans le formalisme des méthodes de Runge-Kutta explicites.
Pour rappel, un pas d'une méthode de Runge-Kutta explicite à s étages peut s'écrire sous la forme :

$$\begin{aligned} k_1 &= f(y_n), \\ k_i &= f\Big(y_n + \Delta t \sum_{j=1}^{i-1} a_{i,j} k_j\Big), \text{ pour } i = 2 \text{ à } s, \\ y_{n+1} &= y_n + \Delta t \sum_{i=1}^{s} b_i k_i, \end{aligned}$$

1. En supposant connus la matrice $A = (a_{i,j})_{1 \le i,j \le s}$ et le vecteur $b = (b_i)_{i=1,s}$, proposer une manière générique de modifier la procédure `Un_pas_temps` pour qu'elle calcule la mise à jour U^{n+1} en utilisant la méthode de Runge-Kutta correspondante.
2. Rajouter dans le module `Edo` une variable globale `sch` qui est un entier lu dans le fichier `Parametres.dat` permettant de choisir un schéma de Runge-Kutta.
   Rajouter dans le module `Schema` trois variables semi-publiques (visibles uniquement à l'intérieur du module) `A` et `b` qui sont des tableaux dynamiques à 2 et 1 dimensions, ainsi que `s` qui représente le nombre d'étages.
   Programmer également une procédure `Init_RK` qui alloue et remplit ces deux tableaux et `s` en fonction de la valeur de `sch`. On considèrera les schémas suivants :

$$\text{Euler explicite} : A = \begin{pmatrix} 0 \end{pmatrix}, \ b = \begin{pmatrix} 1 \end{pmatrix},$$

$$\text{Heun} : A = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}, \ b = \begin{pmatrix} 1/2 & 1/2 \end{pmatrix},$$

$$\text{Kutta} : A = \begin{pmatrix} 0 & 0 & 0 \\ 1/2 & 0 & 0 \\ -1 & 2 & 0 \end{pmatrix}, \ b = \begin{pmatrix} 1/6 & 2/3 & 1/6 \end{pmatrix},$$

$$\text{RK4} : A = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \ b = \begin{pmatrix} 1/6 & 1/3 & 1/3 & 1/6 \end{pmatrix}.$$

3. Modifier la procédure `Un_pas_temps` pour qu'elle mette à jour la solution en utilisant le schéma de Runge-Kutta idoine.
   On utilisera un tableau `ki` qui contient les $(k_i)_{i=2,s}$ (attention : les k_i sont des vecteurs).

4. Tester ces schémas sur le modèle de Lotka-Volterra avec `tfin`=30 et Δt le plus grand pas de temps possible pour que tous ces schémas soient stables.
5. Tester ces schémas sur le problème de Lorenz.
   Notes : on peut tracer un résultat en 3D sous `Gnuplot` en utilisant la commande `splot` au lieu de `plot`. Par ailleurs, quand un fichier contient plus de 2 colonnes, on peut tracer sous `Gnuplot` la colonne j en fonction de la colonne i avec gnuplot en tapant
   `plot 'Resultats.dat' u i:j w lp`.
   (a) Faire un premier calcul avec les paramètres suggérés en introduction et des valeurs de `tfin` allant de 10 à 100. Observer les résultats en 3D, dans les plans (x, y), (x, z) et (y, z) mais aussi les valeurs de x, y et z en fonction de t.
   (b) Faire le calcul en utilisant des conditions initiales légèrement différentes $U_0 = (1+\varepsilon, 1+\varepsilon, 1+\varepsilon)^\top$ pour $\varepsilon = 10^{-2},\ 10^{-4}$ et 10^{-6}. Tracer les courbes sur le même graphe que celle obtenue avec $U_0 = (1, 1, 1)^\top$ (utiliser plusieurs fichiers). Que constate-t-on ?
   Ce système est à l'origine de la métaphore du battement d'ailes dun papillon. Comment peut-on faire cette analogie ?
6. Tester ces schémas sur le problème de balistique.
   (a) Modifier le programme principal pour que la boucle en temps n'ait lieu que tant que $z \geq 0$.
   (b) Etudier l'emplacement des différents points d'impact prévus par les 4 schémas en temps. Que peut-on dire ?