Université de Montpellier **Année Universitaire 2023-2024**
Faculté des Sciences
Mathématiques

L3 : module HAX604X Analyse numérique des équations différentielles.

TP1 INTRODUCTION A LA RÉSOLUTION NUMERIQUE D'ÉQUATIONS DIFFÉRENTIELLES.

Vous utiliserez, *au choix*, Python ou Matlab. En Python, je vous conseille d'utiliser l'interface jupyter notebook pour coder. Pour plus d'information et l'installation sur votre machine personnelle voir `https://jupyter.org/`.

Pour Matlab, Vous pouvez installer sur votre machine personnelle un clone gratuit Octave `https://www.gnu.org/software/octave/` ou une version gratuite pendant une durée limitée, voir `https://fr.mathworks.com/` ou travailler à distance sur une des machines de la FdS avec le client `x2go` `https://moodle.umontpellier.fr/mod/page/view.php?id=126345`.

1 Interprétation géométrique d'une équation différentielle.

Pour apprendre à visualiser le champ de vecteur correspondant à une équation différentielle.

Télécharger depuis moodle le fichier SlopeFields.cdf et l'ouvrir avec l'application Wolfram Player (menu education). (NB Le Wolfram CDF Player est gratuit et permet de lire des fichiers .cdf. Pour *créer* des fichiers cdf il faut le logiciel Mathematica que la FDS n'a pas acheté.) Ce script SlopeFields.cdf permet de visualiser le champ de vecteurs (ou champ de pentes) associé à une équation différentielle

$$y' = f(x, y).$$

En chaque point (x, y) d'une grille, on trace un petit segment de droite de pente égale à $f(x, y)$. Résoudre l'équation différentielle pour une condition initiale donnée, c'est trouver une courbe $x \mapsto y(x)$ vérifiant $y'(x) = f(x, y(x))$, ce qui signifie précisément que la pente de la tangente à la courbe est égale à la pente du petit segment de droite défini au point (x, y), autrement dit que la courbe est *tangente* à tous les petits segments de droites qu'elle rencontre.
La donnée d'une condition initiale $y(x_0) = y_0$ signifie simplement que la courbe passe par le point (x_0, y_0).

Le script .cdf propose un menu de 14 fonctions; *Prenez le temps d'en essayer quelques unes en variant les paramètres, la donnée initiale et de vous interroger sur le résultat visualisé pour développer votre intuition des équations différentielles.* Pensez à **cocher la case show exact solution**. Dans plusieurs exemples, $f(x, y)$ ne dépend pas de x, on dit que l'équation différentielle est *autonome*. Le champ de vecteurs est alors invariant par translation $x \mapsto x + C$. L'ensemble des solutions aussi. Dans un exemple $f(x, y) = g(y/x)$ ne dépend que de la pente y/x. *Par quel type de transformation le champ de vecteur est-il invariant ?* Réponse : homothétie de centre 0. Dans un exemple, $f(x, y)$ ne dépend pas de y, il ne s'agit pas d'une « vraie » équation différentielle mais seulement de la recherche d'un primitive de f. Le champ de vecteur est alors invariant par translation $y \mapsto y + C$. Moralité : Quand un champ de vecteur a une certaine symétrie, l'ensemble des solutions a la même symétrie. [1]

1. Pour plus d'information vous pourrez *visionner chez vous une courte video* sur YouTube sur ce thème : `https://www.youtube.com/watch?v=24pxJ1DuWR8`.

2 Ordre de convergence de la méthode d'Euler.

Ecrire un script Matlab `euleredo.m` ou un jupyter notebook Python `euleredo.ipynb` qui résout l'équation différentielle ordinaire

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

par la méthode d'Euler. Vous considérerez l'équation différentielle $y' = y$. On prendra les paramètres : intervalle de temps `t0=0,tfinal=4` , donnée initiale `y0=1`.
pas de temps initial `h=(tfinal - t0)/4`. On écrira la boucle qui code le schéma d'Euler pour un pas de temps h donné ainsi (syntaxe Matlab)

```
for j=2:N
        y = y + h*f(t,y);
        yapp(j) = y;
        t = t + h;
end
```

Pour un pas de temps h donné, on tracera sur le même graphe la solution calculée par le schéma et la solution exacte $y(t) = \exp t$.

Ensuite vous exécuterez le code avec le pas plus fin $h := h/2$. Vous raffinerez le pas disons 6 fois. Codez cela dans une boucle.

Pour chaque valeur du pas de temps h, on calculera l'erreur $\mathtt{max_j}(\mathtt{abs}(\mathtt{yapp}(\mathtt{j}) - \mathtt{exp}(\mathtt{t_j})))$ entre la solution approchée `yapp` et la solution exacte $\exp(t)$.

On tracera ensuite l'erreur en fonction du pas en échelle log-log. Vérifiez qu'on obtient bien une droite de pente 1. Pourquoi ?

Refaire tourner le code en changeant la fonction $f(t, y) = y^2$, mais il faut rentrer alors la solution exacte $y(t) = 1/(1 - t)$ et ajuster les paramètres (prendre `tfinal` $<$1).