

TPs EDP ^a

Travaux Pratiques N° 3



Algorithmique numérique : E.D.O.

a. Version du 5 novembre 2020

1 Résolution numérique d'équations différentielles ordinaires

1.1 Rappels de quelques schémas numériques usuels

Definition 1 (problème de Cauchy) Soit f l'application continue définie par

$$\begin{aligned} f &: [t^0, t^0 + T] \times \mathbb{R}^d &\longrightarrow & \mathbb{R}^d \\ &(t, \mathbf{y}) &\longmapsto & f(t, \mathbf{y}) \end{aligned}$$

avec $T \in]0, +\infty]$. Le **problème de Cauchy** revient à chercher une fonction \mathbf{y} définie par

$$\begin{aligned} \mathbf{y} &: [t^0, t^0 + T] &\longrightarrow & \mathbb{R}^d \\ &t &\longmapsto & \mathbf{y}(t) \end{aligned}$$

continue et dérivable, telle que

$$\mathbf{y}'(t) = f(t, \mathbf{y}(t)), \quad \forall t \in [t^0, t^0 + T] \tag{1}$$

$$\mathbf{y}(t^0) = \mathbf{y}^{[0]} \in \mathbb{R}^d. \tag{2}$$

■

On note $t^n, n \in \llbracket 0, N \rrbracket$, une discrétisation régulière de $[t^0, t^0+T]$, $\mathbf{y}^{[n]} \approx \mathbf{y}(t^n)$ et $\mathbf{f}^{[n]} = \mathbf{f}(t^n, \mathbf{y}^{[n]})$

1.1.1 Schéma d'Euler progressif

$$\begin{cases} \mathbf{y}^{[n+1]} &= \mathbf{y}^{[n]} + h\mathbf{f}^{[n]}, \quad \forall n \in \llbracket 0, N-1 \rrbracket \\ \mathbf{y}^{[0]} &= \mathbf{y}(t^0) \end{cases} \quad (3)$$

Ce schéma est d'ordre 1.

1.1.2 Schéma de la tangente améliorée

$$\begin{cases} \mathbf{y}^{[n+1]} &= \mathbf{y}^{[n]} + h\mathbf{f}(t^n + \frac{h}{2}, \mathbf{y}^{[n]} + \frac{h}{2}\mathbf{f}^{[n]}), \quad \forall n \in \llbracket 0, N-1 \rrbracket \\ \mathbf{y}^{[0]} &= \mathbf{y}(t^0) \end{cases} \quad (4)$$

Ce schéma est d'ordre 2.

1.1.3 Schéma de Runge-Kutta d'ordre 4

$$\begin{aligned} \mathbf{k}_1^{[n]} &= \mathbf{f}(t^n, \mathbf{y}^{[n]}) \\ \mathbf{k}_2^{[n]} &= \mathbf{f}(t^n + \frac{h}{2}, \mathbf{y}^{[n]} + \frac{h}{2}\mathbf{k}_1^{[n]}) \\ \mathbf{k}_3^{[n]} &= \mathbf{f}(t^n + \frac{h}{2}, \mathbf{y}^{[n]} + \frac{h}{2}\mathbf{k}_2^{[n]}) \\ \mathbf{k}_4^{[n]} &= \mathbf{f}(t^n + h, \mathbf{y}^{[n]} + h\mathbf{k}_3^{[n]}) \\ \mathbf{y}^{[n+1]} &= \mathbf{y}^{[n]} + \frac{h}{6}(\mathbf{k}_1^{[n]} + 2\mathbf{k}_2^{[n]} + 2\mathbf{k}_3^{[n]} + \mathbf{k}_4^{[n]}). \end{aligned} \quad (5)$$

1.1.4 Méthodes d'Adams-Bashforth

On note $\mathbf{f}^{[n]} = \mathbf{f}(t^n, \mathbf{y}^{[n]})$

$$\mathbf{y}^{[n+1]} = \mathbf{y}^{[n]} + \frac{h}{2} \left(3\mathbf{f}^{[n]} - \mathbf{f}^{[n-1]} \right). \quad (6)$$

$$\mathbf{y}^{[n+1]} = \mathbf{y}^{[n]} + \frac{h}{12} \left(23\mathbf{f}^{[n]} - 16\mathbf{f}^{[n-1]} + 5\mathbf{f}^{[n-2]} \right). \quad (7)$$

$$\mathbf{y}^{[n+1]} = \mathbf{y}^{[n]} + \frac{h}{24} \left(55\mathbf{f}^{[n]} - 59\mathbf{f}^{[n-1]} + 37\mathbf{f}^{[n-2]} - 9\mathbf{f}^{[n-3]} \right). \quad (8)$$

Ces schémas sont **explicites** et leur ordre correspond au nombre de pas.

1.1.5 Méthodes d'Adams-Moulton

$$\mathbf{y}^{[n+1]} = \mathbf{y}^{[n]} + \frac{h}{2} \left(\mathbf{f}^{[n+1]} + \mathbf{f}^{[n]} \right). \quad (9)$$

$$\mathbf{y}^{[n+1]} = \mathbf{y}^{[n]} + \frac{h}{12} \left(5\mathbf{f}^{[n+1]} + 8\mathbf{f}^{[n]} - \mathbf{f}^{[n-1]} \right). \quad (10)$$

$$\mathbf{y}^{[n+1]} = \mathbf{y}^{[n]} + \frac{h}{24} \left(9\mathbf{f}^{[n+1]} + 19\mathbf{f}^{[n]} - 5\mathbf{f}^{[n-1]} + \mathbf{f}^{[n-2]} \right). \quad (11)$$

Ces schémas sont **implicites** et leur ordre correspond au nombre de pas plus un.

1.2 Schéma prédicteur-correcteur

Une méthode de prédiction-correction utilise un schéma explicite comme prédicteur et un schéma implicite comme correcteur. Pour déterminer $\mathbf{y}^{[n+1]}$ de la méthode de prédiction-correction, on va tout d'abord *prédire* une valeur $\bar{\mathbf{y}}^{[n+1]}$ à l'aide du schéma explicite en utilisant les termes $\mathbf{y}^{[n]}, \mathbf{y}^{[n-1]}, \dots$ calculés par la méthode de prédiction-correction. Ensuite $\mathbf{y}^{[n+1]}$ est calculé à l'aide du schéma implicite en utilisant les termes $\mathbf{y}^{[n]}, \mathbf{y}^{[n-1]}, \dots$ et en remplaçant les occurrences de $\mathbf{y}^{[n+1]}$ par $\bar{\mathbf{y}}^{[n+1]}$.

1.3 Travail à effectuer

Le but est de représenter graphiquement les erreurs données par plusieurs schémas et de retrouver numériquement leur ordre. Pour cela il faudra pouvoir connaître explicitement la solution du problème de Cauchy étudié. Voir l'annexe 3.1 pour plusieurs exemples de problèmes de Cauchy avec solutions.

Q. 1 1. *Ecrire les cinq fonctions Matlab suivantes correspondant à la résolution d'un problème de Cauchy :*

- $[t, Y]=redEUP(f,a,b,yo,N)$: schéma d'Euler progressif.
- $[t, Y]=redTGA(f,a,b,yo,N)$: schéma de la tangente améliorée.
- $[t, Y]=redRK4(f,a,b,yo,N)$: schéma de Runge et Kutta d'ordre 4.
- $[t, Y]=redAB3(f,a,b,yo,N)$: schéma d'Adams-Bashforth d'ordre 3.
- $[t, Y]=redPC4(f,a,b,yo,N)$: schéma de type prédiction-correction utilisant les schémas d'Adams-Bashforth et d'Adams-Moulton d'ordre 4.

Ici les paramètres f , a , b , yo correspondent respectivement aux $\mathbf{f}, t^0, t^0 + T, \mathbf{y}^{[0]}$ du problème de Cauchy (1-2). Enfin, Y est le tableau contenant les $\mathbf{y}^{[n]}$, $n \in \{0, \dots, N\}$ et t est le tableau contenant les $N+1$ nombres t^n , $n \in \{0, \dots, N\}$

2. *Ecrire le programme principal (fichier `erreur.m`) permettant le calcul et le tracé des erreurs. Pour une méthode donnée le tracé de l'erreur*

correspond au tracé de l'ensemble des points $(t^n, \text{abs}(\mathbf{y}^{[n]} - \mathbf{y}(t^n)))$, $n \in \{0, \dots, N\}$.

Voir la figure 1 pour un exemple de tracé.

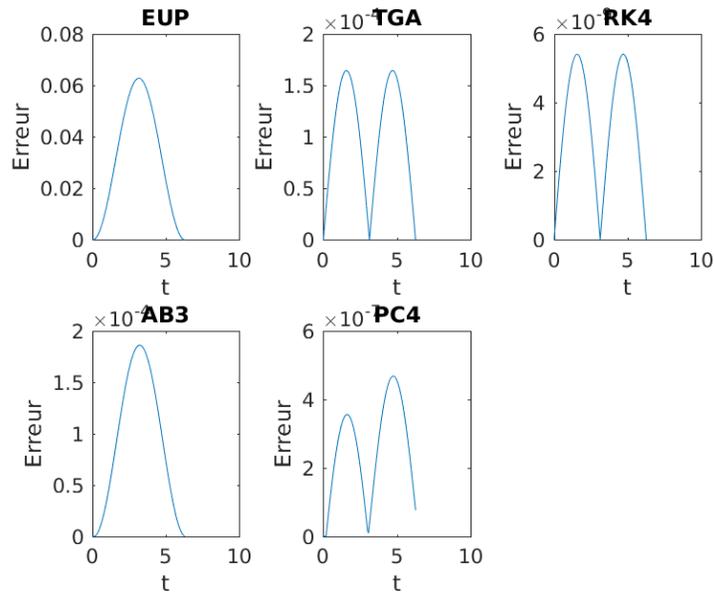


FIGURE 1 – Valeurs absolues des erreurs des 5 schémas

3. Prendre $N=100$ dans le programme `erreur` et noter les valeurs maximales des 5 courbes d'erreurs. Prendre $N=1000$ dans le programme `erreur`. A quoi doit-on s'attendre ? (lien avec les ordres des schémas)
4. Ecrire le programme principal (fichier `ordre.m`) permettant de calculer numériquement l'ordre des 5 schémas et de le représenter. Voir la figure 2 pour un exemple de tracé.

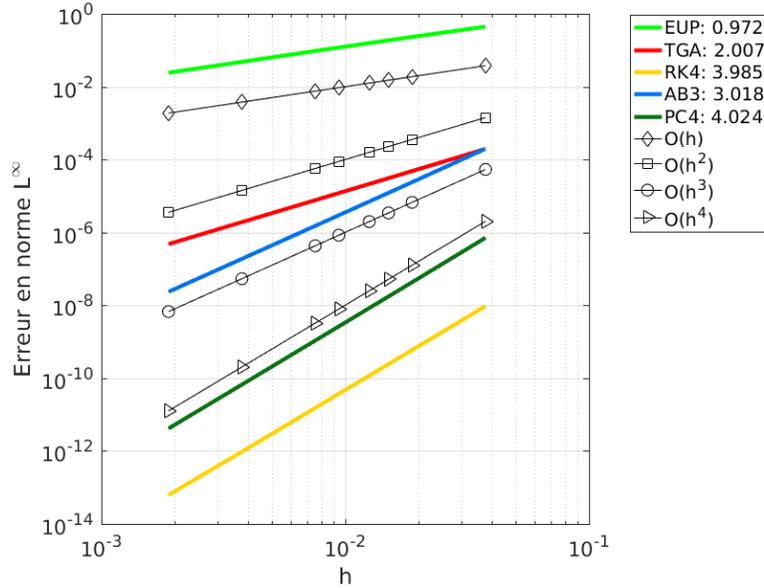


FIGURE 2 – Ordre des 5 schémas

2 Le système solaire

Une archive compressée au format **zip**

www.math.univ-paris13.fr/~cuvelier/docs/Enseignements/mac2/TPs-EDP/20-21/TP3_CodesFournis_SystemeSolaire.zip

est disponible en ligne sur ma page web et sur le serveur Discord. Il faut télécharger l'archive et la décompresser dans un répertoire.

2.1 Position du problème et équations différentielles

Le but de cette partie est de prédire la position de planètes dans le système solaire à partir de positions et de vitesses initiales des planètes. Les unités choisies sont : les masses sont relatives au soleil, les distances sont en unités astronomiques (1 U.A. = 149 597 870 km) et le temps en jour terrestre.

En astrophysique, la masse solaire est l'unité de masse conventionnellement utilisée pour exprimer les masses des corps célestes massifs et des structures formées d'étoiles (amas, superamas, galaxies, etc.). Son symbole et sa valeur sont :

$$M_{\odot} = 1,9891 \times 10^{30} \text{ kg.}$$

La masse solaire peut aussi être déterminé par l'Unité astronomique (U.A.),

l'année et le Constante gravitationnelle (G) :

$$M_{\odot} = \frac{4\pi^2 \times (1\text{UA})^3}{G \times (1\text{ans})^2}.$$

La constante gravitationnelle est alors

$$G = \frac{4\pi^2}{(365,25636567)^2} \approx 2.959130713485796 \times 10^{-4},$$

une année sidérale correspondant à 365,25636567 jours.

On note $\mathbf{q}_i(t)$ la position du i ème corps au temps t , $\mathbf{v}_i(t)$ sa vitesse au temps t et m_i sa masse. Les données initiales pour le soleil (corps $i = 1$) sont $\mathbf{q}_1 = (0, 0, 0)^t$ et $\mathbf{v}_1 = (0, 0, 0)^t$. Toutes les données sont fournies dans le fichier `SysSolData.m` et ont récupérées sur JPL Solar System Dynamics. Elles correspondent aux positions, vitesses, etc, de différentes planètes le 6 mai 2019 à 0h00m00s.

De manière classique, un problème à N corps se modélise par

$$\frac{d^2 \mathbf{q}_i}{dt^2}(t) = -G \sum_{j=1, j \neq i}^N m_j \frac{\mathbf{q}_i(t) - \mathbf{q}_j(t)}{\|\mathbf{q}_i(t) - \mathbf{q}_j(t)\|^3}, \quad \forall i \in \llbracket 1, N \rrbracket.$$

et l'énergie du système ,correspondant à la somme des énergies cinétiques et des énergies potentielles gravitationnelle est donnée par

$$\mathcal{E}(t) = \frac{1}{2} \sum_{i=1}^N m_i \|\mathbf{v}_i(t)\|^2 - G \sum_{i=1}^N \sum_{j=1}^{i-1} \frac{m_i m_j}{\|\mathbf{q}_i(t) - \mathbf{q}_j(t)\|}.$$

2.2 Résolution numérique classique

Q. 2 (Sur feuille, à rendre, non noté) 1. *Ecrire de manière détaillée le problème de Cauchy associé à un problème à N corps sachant que la fonction vectorielle inconnues \mathbf{y} est définie par*

$$\mathbf{y}(t) = \begin{pmatrix} \mathbf{q}_1(t) \\ \vdots \\ \mathbf{q}_N(t) \\ \mathbf{q}'_1(t) \\ \vdots \\ \mathbf{q}'_N(t) \end{pmatrix} \in \mathbb{R}^{6N}$$

2. *Quelles sont les données du problème de Cauchy obtenu ? (avec leur type détaillé : entier, réel, complexe, vecteur, matrice, fonction, ...)*
3. *Quelles sont les inconnues du problème de Cauchy obtenu ? (avec leur type détaillé : entier, réel, complexe, vecteur, matrice, fonction, ...)* ■

2.3 Système à 3 corps : Soleil, Saturne et Jupiter

Pour récupérer les données initiales de ce système, on peut utiliser les commandes Matlab :

```
SysSol3=SysSolData();  
SysSol3.planetes=SysSol3.planetes(1:3);
```

Les deux fonctions fournies `plotSysSol` et `PlotPlanets` (necessite `PlotPlanet`) permettent de représenter, après calculs, respectivement l'orbite des planetes de la structure `SysSol` (voir fonction `SysSolData`) et les planètes à un instant donné. Leur syntaxe est la suivante `plotSysSol(T,Y,SysSol)` et `PlotPlanets(SysSol,q)` avec

- `(T,Y)` tableaux retournés lors de la résolution du problème de Cauchy associé,
- `SysSol` structure contenant divers renseignements sur le système solaire,
- `q` positions des planetes.

- Q. 3 (Matlab)**
1. Ecrire la fonction Matlab `fSysSol3` (fichier `fSysSol3.m`) correspondant à la fonction f du problème de Cauchy associé au problème à 3 corps (Soleil, Saturne et Jupiter). On pourra utiliser une variable globale (mais il y a mieux) pour les paramètres physiques stockés dans une structure (voir fichier `SysSolData.m`).
 2. Ecrire le programme `prgSysSol3` (fichier `prgSysSol3.m`) qui résoud le problème de Cauchy par `resEUP` et `resRK4` puis représente, pour chaque méthode, les orbites des 3 planètes ainsi que leurs positions à l'instant final avec comme données $T = 24000$ et $N = 8000$ (voir figure 3) On représentera aussi l'énergie du système pour chaque méthode.



FIGURE 3 – Orbites Soleil-Saturne-Jupiter

2.4 Système à 6 corps

On étudie ici le système composé des 6 planetes Soleil, Saturne, Jupiter, Uranus, Neptune et Pluton.

- Q. 4 (Matlab)**
1. Ecrire la fonction Matlab `fSysSol6` (fichier `fSysSol6.m`) correspondant à la fonction \mathbf{f} du problème de Cauchy associé au problème à 6 corps (Soleil, Saturne, Jupiter, Uranus, Neptune et Pluton). On pourra utiliser une variable globale pour les paramètres physiques (voir fichiers `SysSolDataHairer.m` ou `SysSolData.m`).
 2. Ecrire le programme `prgSysSol6` (fichier `prgSysSol6.m`) qui résoud le problème de Cauchy par `resEUP` et `resRK4` puis représente, pour chaque méthode, les orbites des 6 planètes ainsi que leurs positions à l'instant final avec comme données $T = 100000$ et $N = 10000$ (voir figure 4) On représentera aussi l'énergie du système pour chaque méthode.

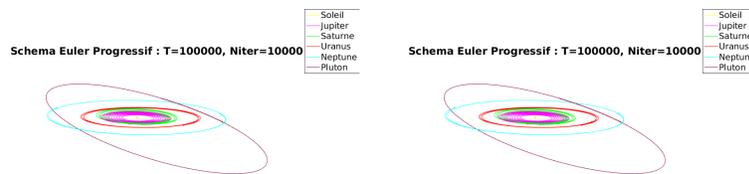


FIGURE 4 – Orbites Soleil, Saturne, Jupiter, Uranus, Neptune et Pluton

2.5 Système à 7 corps

On étudie ici le système composé des 7 planètes Soleil, Saturne, Jupiter, Uranus, Neptune, Pluton et Terre.

- Q. 5 (Matlab)**
1. Ecrire la fonction Matlab `fSysSol7` (fichier `fSysSol7.m`) correspondant à la fonction \mathbf{f} du problème de Cauchy associé au problème à 7 corps (Soleil, Saturne, Jupiter, Uranus, Neptune, Pluton et Terre). On pourra utiliser une variable globale pour les paramètres physiques (voir fichiers `SysSolDataHairer.m` ou `SysSolData.m`).
 2. Ecrire le programme `prgSysSol7` (fichier `prgSysSol7.m`) qui résout le problème de Cauchy par `resEUP` et `resRK4` puis représente, pour chaque méthode, les orbites des 7 planètes ainsi que leurs positions à l'instant final avec comme données $T = 100000$ et $N = 10000$. On représentera aussi l'énergie du système pour chaque méthode. ■

3 Annexes

3.1 Quelques E.D.O. du premier ordre

3.1.1 Exemple 1

Soit $\alpha \in \mathbb{R}$. L'E.D.O.

$$\begin{cases} y'(t) = \cos(t), \forall t \geq 0, \\ y(0) = \alpha, \end{cases}$$

a pour solution $y(t) = \sin(t) + \alpha$.

3.1.2 Exemple 2

Soit $\beta \in \mathbb{R}$. L'E.D.O.

$$\begin{cases} y'(t) = \sin(t), \forall t \geq 0, \\ y(0) = \beta, \end{cases}$$

a pour solution $y(t) = -\cos(t) + 1 + \beta$.

3.1.3 Exemple 3

L'E.D.O.

$$\begin{cases} y'(t) = -y(t) \sin(t), \forall t \geq 0, \\ y(0) = e = \exp(1), \end{cases}$$

a pour solution $y(t) = \exp(\cos(t))$.

Références

- [1] C. Lubich E. Hairer and G. Wanner. *Geometrical Numerical Integration*. Springer, 2006.
- [2] S. Trilles J. Anton and P. Mercier. Study of integration schemes suited for the long-term extrapolation.