BE 2 – Introduction à l'analyse numérique des Équations Différentielles Ordinaires (EDO)

1 EDO non-linéaire à coefficients variables

On cherche à résoudre :

$$\begin{cases} \frac{dy}{dt} = -2t\,y^2 \\ y(0) = 1 \end{cases} \tag{1}$$

On vérifiera que (1) a pour solution exacte $y(t) = \frac{1}{t^2+1}$.

1. Implémenter la méthode d'Euler explicite avec $h = 0.1$ et $0s < t < 2s$. Afficher la solution exacte et la solution approchée. Calculer l'erreur maximale $0s < t < 2s$.
2. Répéter 1. avec $h = 0.01$. Quel est le changement sur l'erreur maximale ? Justifier
3. Résoudre l'EDO (1) en utilisant la méthode de Runge-Kutta au second ordre (ou méthode d'Euler modifiée, donnée en annexe) avec $h = 0.2$ et $0s < t < 2s$. Afficher la solution exacte et la solution approchée pour $h = 0.01$ et $h = 0.2$. Calculer l'erreur maximale sur $0s < t < 2s$. En vous basant sur cette erreur maximale, quelle est la méthode la plus précise et de quel facteur ?
4. Résoudre l'EDO (1) sur $0s < t < 2s$ en utilisant la fonction Runge-Kutta 4 (`RK4.m` disponible sur *le LMS rubrique SUPAERO 1A/MA101*). Afficher la solution exacte et la solution approchée pour $h = 0.01$ et $h = 0.2$. Calculer l'erreur maximale sur $0s < t < 2s$ et la comparer aux résultats des questions 1. et 3.
5. Résoudre l'EDO (1) sur $0s < t < 2s$ en utilisant la fonction Matlab `ode45` (taper `help ode45` pour plus d'informations). Afficher la solution exacte et la solution approchée pour $h = 0.01$ et $h = 0.2$. Calculer l'erreur maximale sur $0s < t < 2s$ et la comparer aux résultats des questions 1. et 3.

   `Note sur ode45` : Cette fonction réalise un processus itératif en adaptant le pas. Pour pouvoir comparer à la fonction RK4, il faut utiliser la bonne option :`options=odeset('MaxStep',h)` et passer `options` en argument de la fonction `ode45`.
6. Représenter sur un même diagramme (log-log, `en matlab la fonction est loglog`) les erreurs maximales ET moyennes sur $0s < t < 2s$ obtenues pour différents pas h, et ce pour les quatre méthodes étudiées : Euler explicite, Euler modifiée, Runge-Kutta 4 et ODE45. Vous devez pour cela programmer une fonction `[ME,RMSE]=ErreursMethodes(h)` qui renvoit l'erreur maximale ME, et l'erreur quadratique moyenne RMSE. Conclure sur l'ordre de ces méthodes.

2 Système mécanique oscillant

On considére un système mécanique oscillant. L'équation du mouvement donne l'EDO suivante :

$$\begin{cases} m\frac{d^2x}{dt^2} + c\frac{dx}{dt} + kx = F(t) \\ x(0) = \frac{dx}{dt}(0) = 0 \end{cases} \tag{2}$$

où l'excitation F est de type échelon,i.e.

$$F(t) = \begin{cases} 1 & \text{pour} \quad t > 0 \\ 0 & \text{pour} \quad t \leq 0 \end{cases}$$

Comme (2) fait intervenir $\frac{d^2x}{dt^2}$, on doit écrire un système équivalent de deux équations couplées du premier ordre en temps.

1. Ecrire ce système en posant $y_1(t) = x(t)$ et $y_2(t) = \frac{d}{dt}x(t)$.
2. Comparer la réponse exacte à un échelon unité à l'approximation calculée par la fonction `ode45` :
   Paramètres : $m = 1$ kg, $K = 1$ N/m, $c = 0.2$ Ns/m sur un horizon $t = 0$ à $50s$ et un pas de $0.25s$.
3. Programmer la méthode d'Euler explicite pour un pas petit $h = 0.0625$ (1/4 du pas de `ode45`) et grand $h = 0.25$. Que constatez-vous ? Conclure.
4. Programmer la méthode d'Euler implicite pour un pas petit $h = 0.0625$ et grand $h = 0.25$. Que constatez-vous ? Conclure.

A ODE45.m et RK4.m

Les fonctions ode45 et rk4 ont la même syntaxe. `[t,y] = ode45(fname, tspan, y0, opts)`

- fname est le nom de la fonction `function dydt = fname(t,y)` à intégrer
- tspan est un vecteur à 2 éléments définissant l'intervalle d'intégration (peut être aussi le vecteur définissant tous les instants d'intégration de la solution)
- y0 vecteur des conditions initiales
- opts argument optionnel, structure Matlab pour contrôler les détails du calcul
- t valeurs du vecteur temps (peut ne pas être uniforme si on utilise TSPAN, algorithme à pas adaptatif)
- y valeurs de la solution

B Système du second ordre : solution analytique

On donne la solution exacte de la réponse à un échelon d'un système du second ordre (2) :

$$x(t) = 1 - e^{-\zeta\omega_n t}\Big(\frac{\zeta}{\sqrt{1-\zeta^2}}\sin\omega_d t + \cos\omega_d t\Big)\,; \tag{3}$$

$$\text{avec} \quad \omega_n := \sqrt{\frac{k}{m}}, \tag{4}$$

$$\zeta := \frac{c}{2\sqrt{\frac{k}{m}}}, \tag{5}$$

$$\omega_d := \omega_n\sqrt{1-\zeta^2}\,. \tag{6}$$

C PROGRAMMING NUMERICAL METHODS IN MATLAB

In this section, we first show how to program Euler method, modified-Euler and Runge-Kutta methods in MATLAB. Although MATLAB has built-in solvers, learning to program such algorithms will improve your understanding of these methods.

C.1 PROGRAMMING THE EULER METHOD

The Euler algorithm for the equation $\dot{y} = f(t, y)$ is

$$y_{k+1} = y_k + h\, f(t_k, y_k) \tag{7}$$

where $t_{k+1} = t_k + h$.

This equation can be applied successively at the times t_k, for example, by putting it in a `for` loop in a MATLAB program.

C.2 PROGRAMMING THE MODIFIED-EULER METHOD

The modified Euler algorithm for the equation $\dot{y} = f(t, y)$ is

– Euler predictor

$$x_{k+1} = y_k + h\, f(t_k, y_k) \tag{8}$$

– Trapezoidal predictor

$$y_{k+1} = y_k + \frac{h}{2}\Big(f(t_k, y_k) + f(t_{k+1}, x_{k+1})\Big) \tag{9}$$

C.3 PROGRAMMING THE RUNGE-KUTTA METHOD

The fourth-order Runge-Kutta algorithm is

$$y_{k+1} = y_k + w_1 g_1 + w_2 g_2 + w_3 g_3 + w_4 g_4 \tag{10}$$

where

$$\begin{aligned}
g_1 &= h\, f(t_k, y_k) && (11)\\
g_2 &= h\, f(t_k + \alpha_1 h, y_k + \alpha_1 g_1) && (12)\\
g_3 &= h\, f(t_k + \alpha_2 h, y_k + \beta_2 g_2 + (\alpha_2 - \beta_2)) && (13)\\
g_4 &= h\, f(t_k + \alpha_3, y_k + \beta_3 g_2 + \gamma_3 g_3 + (\alpha_3 - \beta_3 - \gamma_3) g_1) && (14)
\end{aligned}$$

We will use the parameters for the classical Runge-Kutta method, which are

$$\begin{array}{ll}
w_1 = w_4 = \frac{1}{6} & w_2 = w_3 = \frac{1}{3} \\
\alpha_1 = \alpha_2 = \frac{1}{2} & \beta_2 = \frac{1}{2} \\
\gamma_3 = \alpha_3 = 1 & \beta_3 = 0
\end{array} \tag{15}$$