TD3 Scilab - Equations différentielles

Eric Ringeisen

Novembre 2016

La fonction scilab ode()

Lancez Scilab et SciNotes, et créez un nouveau fichier contenant les instructions suivantes

```
// nom1 prenom1
// nom2 prenom2
clear();         // Efface les variable entre deux exécutions
xdel(winsid()); // Détruit les fenêtres graphiques entre deux exécutions
format(20);     // Définit le format d'affichage des nombres réels
```

On s'intéresse à l'équation différentielle suivante décrivant l'évolution dans le temps d'une quantité $y(t)$:

$$y'(t) = 3\ t\ y(t) + 1 \qquad (E)$$

Q1 A l'aide d'un bloc `function ... endfunction` en scilab, définir une fonction de deux paramètres `(t, y)` qui calcule la quantité $G(t, y) = 3\ t\ y + 1$. On peut écrire l'équation différentielle sous la forme

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

En scilab, on prendra soin d'utiliser l'opérateur `.*` pour la multiplication au lieu de `*`
La fonction

```
y = ode("rk", u, a, t, G)
```

de Scilab permet de calculer une solution de l'équation différentielle. Concrètement, son argument `t` sera un vecteur d'abscisses $(t_1, t_2, \ldots, t_n)$, et la fonction retourne un vecteur $(y_1, y_2, \ldots, y_n)$ contenant les valeurs approchées, $y_i = y(t_i)$ de la solution aux instants t_i. L'argument `a` doit être l'instant initial t_1, et l'argument `u` est un nombre réel, la valeur $u = y(t_1)$ de la solution à l'instant t_1. Le premier argument `"rk"` indique à Scilab quelle méthode utiliser pour la résolution, parmi toutes celles qu'il connait. On choisit ici une méthode qui donne un résultat assez précis.

Q2 Définir en Scilab les valeurs `u, a, t` qu'il faut passer à l'appel de `ode()` pour obtenir la solution de l'équation (E) définie sur l'intervalle $t \in [0, 1]$ et vérifiant la condition initiale $y(0) = 0.5$. On pourra prendre un vecteur de 100 valeurs t_i.

Pour tracer la solution, on va ouvrir une fenêtre graphique par les instructions

```
fenetre = figure("Figure_name", "Equations", "position", [100 50 1000 600]);
fenetre.background = color("white");
set("current_figure", fenetre);
```

et on réalise le tracé comme suit à l'aide de la fonction `plot2d`

```
subplot(2, 2, 1);
plot2d(t', y', style=[color("black")]);
```

On a utilisé ici l'appel `subplot(2, 2, 1)` dont l'éffet est de découper la fenêtre graphique en quatre zones de dessin (2 lignes et 2 colonnes), et de sélectionner la première de ces quatre zones pour le tracé. On note que `plot2d` prend des vecteurs colonnes en paramètres, c'est pourquoi on lui passe `t'` et `y'` au lieu de `t, y`.

Remontée dans le temps

On aimerait tracer la solution sur l'intervalle $t \in [-1, 0]$ (avec toujours la condition $y(0) = 0.5$). Une astuce est possible : on considère la fonction $z(t) = y(-t)$, qui satisfait l'équation différentielle modifiée

$$z'(t) = 3\ t\ z(t) - 1 \qquad (E_2)$$

Q3 Définir en scilab une fonction `G2` valant $G_2(t, z) = 3\ t\ z - 1$
On peut ensuite calculer et tracer

```
z = ode("rk", u, a, t, G2);
plot2d(-t', z', style=[color("black")]);
```

Ajustez l'axe des ordonnées pour qu'il passe par l'origine

```
coor = get("current_axes");
coor.y_location = "origin";
```

Q4 Compléter la boucle suivante

```
for u = [0 1 2]
    y = ode ...;
    z = ode ...;
    plot2d ...;
    plot2d ...;
end;
```

pour ajouter à votre dessin les solutions satisfaisant les conditions initiales $y(0) = 0,\ y(0) = 1,\ y(0) = 2$.

Méthode d'Euler

La méthode d'Euler pour l'équation (E) consiste à remplacer la pente $y'(t)$ par la différence finie

$$\frac{y_{i+1} - y_i}{t_{i+1} - t_i} = G(t_i, y_i) \qquad \Longrightarrow \qquad y_{i+1} = y_i + (t_{i+1} - t_i)\, G(t_i, y_i)$$

Cette formule donne un algorithme permettant de calculer par récurrence les valeurs $y(i)$. Voici une implémentation possible en Scilab : la fonction suivante résout l'équation différentielle sur l'intervalle $[a, b]$ avec la condition $y(a) = u$ et un nombre n de points de discrétisation t_i, c'est à dire que l'intervalle $[a, b]$ est découpé en $n - 1$ intervalles de longeur $h = \dfrac{b - a}{n - 1}$

```
function [t, y] = euler(g, u, a, b, n)
    t = linspace(a, b, n)   // calcul du vecteur d'abscisses t(i)
    h = (b-a)/(n-1);        // c'est la longueur t(i+1)-t(i)
    y = zeros(n);           // on définit un vecteur contenant n valeurs nulles
    y(1) = u;               // la première valeur vaut u (condition initiale)
    for i = 1:n-1           // cette boucle calcule chaque valeur à partir de la précédente
        y(i+1) = y(i) + h * g(t(i), y(i));
    end;
endfunction;
```

On remarque que cette fonction retourne deux résultats : le vecteur d'abscisses `t` et le vecteur d'ordonnées `y`. On appellera donc la fonction par exemple par

```
[te, ye] = euler(G, 2, 0, 1, 5);
```

Q5 Expliquez ce qui est calculé par l'appel de fonction précédent.

Q5 Utilisez `subplot(2, 2, 2);` pour tracer dans le deuxième quart de la fenêtre et tracer en bleu à l'aide de `plot2d` la solution qui vient d'être calculée par la méthode d'Euler.

Q6 Calculez et tracez en vert la solution pour la méthode d'euler avec la même donnée initiale et 20 points de discrétisation.

Q7 A titre de comparaison, tracez en noir sur le même graphique la solution `yref` obtenue par la fonction `ode()`, avec la même donnée initiale.

On peut calculer l'écart entre la solution trouvée et la solution de référence au point $t = 1$ par l'expression

```
erreur = abs(ye(length(ye)) - yref(length(yref)));
```

Q8 Vérifiez en faisant varier le nombre `n` de points de discrétisation (c'est à dire la longueur du vecteur `ye`), que le produit `n * erreur` est à peu près constant.

Méthode de Heun

Une variation de la méthode d'Euler, la méthode de Heun, consiste à utiliser la formule de récurrence

$$v = y_i + (t_{i+1} - t_i)\, G\,(t_i, y_i) \qquad y_{i+1} = y_i + (t_{i+1} - t_i)\, \frac{G\,(t_i, y_i) + G\,(t_{i+1}, v)}{2}$$

La valeur v calculée est celle donnée par la méthode d'Euler, et la pente utilisée est une moyenne entre les pentes aux points (t_i, y_i) et (t_{i+1}, v).

Q9 Ecrire une fonction `function [t, y] = heun(g, u, a, b, n) ... endfunction` ressemblant beaucoup à la fonction `euler()`, qui calcule la solution par la méthode de `Heun`.

Q10 Dans le `subplot(2, 2, 3)`, tracer les solutions obtenues avec 5 points de discrétisation pour les méthodes de Heun et d'Euler, ainsi que la solution de référence. On utilisera des couleurs différentes pour les courbes.

Complément : méthode du point milieu

C'est une autre variation de la méthode d'Euler qui utilise la formule de récurrence

$$v = y_i + 0.5\,(t_{i+1} - t_i)\, G\,(t_i, y_i) \qquad y_{i+1} = y_i + (t_{i+1} - t_i)\, G\left(\frac{t_i + t_{i+1}}{2}, v\right)$$

Q11 Ecrire une fonction `function [t, y] = mipoint(g, u, a, b, n) ... endfunction` ressemblant beaucoup à la fonction `euler()`, qui calcule la solution par la méthode du point milieu.

Q12 Ajouter au dessin précédent la courbe de la solution de cette méthode pour 5 points de discrétisation.