Travaux Dirigés 3 - Correction (2h)

Exercice 1 : Méthode de Euler

1. Rappeler la méthode de Euler.
   Il s'agit, pour résoudre l'équation différentielle :
   $$y'(x) = f(x, y(x))$$
   de poser l'approximation :
   $$y'(x_n) \simeq \frac{y(x_{n+1}) - y(x_n)}{h}$$
   avec $x_n = x_0 + nh$, soit encore :
   $$y'_n = \frac{y_{n+1} - y_n}{h}$$

2. Démontrer cette formule.
   On en déduit :
   $$y'_n = \frac{y_{n+1} - y_n}{h} = f(x_n, y_n) = f_n$$
   donc, très simplement :
   $$y_{n+1} = y_n + hf_n$$

3. Donner alors l'algorithme correspondant.

   Algorithm 1: méthode d'Euler

   Data: h, x_0, y_0, x_{max}
   Initialiser x à x_0 et y à y_0
   while $x < x_{max}$ **do**
   $f = f(x, y)$
   $y = y + hf$
   $x = x + h$
   afficher le point solution (x, y)
   end

4. Donner un code python correspondant.
   cf le code python : euler1dim.py

Exercice 2 : Méthode de Runge-Kutta 2

1. Rappeler la méthode de Runge-Kutta 2.
   Il existe une infinité de méthodes de RK2, selon la valeur que l'on retient pour le paramètre β. On rappelle simplement que l'algorithme se présente sous la forme : la fonction incrément est de la forme :
   $$\Phi(x,y,h) = (1-\beta)f(x,y) + \beta f(x+\frac{h}{2\beta}, y+\frac{h}{2\beta}f(x,y))$$
   avec par définition :
   $$y_{n+1} - y_n = h\Phi(x_n, y_n, h)$$
2. Démontrer cette formule en vous aidant du cours. On gardera le paramètre β.
   Cf le cours ! Pour démontrer la formule sur Δ, exprimer $e'(x+h)$ et le développer à l'ordre 1 en h, puis intégrer entre $h=0$ et h et on trouvera la formule. Si la démonstration semble trop compliquée, juste indiquer la méthode générale et l'esprit de la démonstration, ou passer même cette question complètement, et accepter le résultat.
3. Donner alors les formules de Runge-Kutta selon que $\beta = 1$ ou $\beta = \frac{1}{2}$.
   Parmi toutes les valeurs possibles de β, deux cas sont restés dans l'histoire :
   $$\beta = \frac{1}{2}$$
   $$y_{n+1} = y_n + \frac{h}{2}(f(x_n, y_n) + f(x_n+h, y_n + hf(x_n, y_n)))$$
   et aussi le choix :
   $$\beta = 1$$
   $$y_{n+1} = y_n + hf(x_n + \frac{h}{2}, y_n + \frac{h}{2}f(x_n, y_n))$$
4. Quelles autres valeurs judicieuses pourrait-on choisir pour β encore ? Justifier.
   Les élèves peuvent proposer d'autres valeurs, sauf zéro ! car sinon on retombe sur Euler, mais seule l'expérience numérique peut valider un choix précis de β.
5. Choisir une valeur de β et écrire l'algorithme de Runge-Kutta 2 correspondant.
   On choisit $\beta = 1$, d'où l'algorithme :

   Algorithm 2: méthode de Runge-Kutta2

   Data: h, x_0, y_0, x_{max}
   Initialiser x à x_0 et y à y_0
   while $x < x_{max}$ **do**
   - $k_1 = f(x,y)$
   - $k_2 = f(x+\frac{h}{2}, y+\frac{hk_1}{2})$
   - $y = y + hk_2$
   - $x = x + h$
   - afficher le point solution (x,y)

   end
6. Donner un code python correspondant.
   cf le code RK2unedim.py correspondant.

Exercice 3 : Méthode de Runge-Kutta 4

On fournit la méthode de RK4 :

$$k_1 = f(x_n, y_n)$$

$$k_2 = f(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_1)$$

$$k_3 = f(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_2)$$

$$k_4 = f(x_n + h, y_n + hk_3)$$

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

$$x_{n+1} = x_n + h$$

1. Donner l'algorithme de RK4.

Algorithm 3: méthode de Runge-Kutta4

Data: h, x_0, y_0, x_{max}
Initialiser x à x_0 et y à y_0
while $x < x_{max}$ **do**
  $k_1 = f(x, y)$
  $k_2 = f(x + \frac{h}{2}, y + \frac{hk_1}{2})$
  $k_3 = f(x + \frac{h}{2}, y + \frac{hk_2}{2})$
  $k_4 = f(x + h, y + hk_3)$
  $y = y + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$
  $x = x + h$
  afficher le point solution (x, y)
end

2. Donner un code python correspondant.
   cf code RK4.py correspondant.
3. Pourquoi cette méthode de Runge-Kutta est numérotée "4" ?
   Parce que l'erreur de troncation est de l'ordre de h^4.

Exercice 4 : Méthode de Romberg

1. A quoi est utile cette méthode ?
   Cette méthode permet de calculer la valeur numérique d'une intégrale.
2. Rappeler les deux formules à la base de l'algorithme ?
   On montre que :

$$T_k(h) = T_{k-1}(\frac{h}{2}) + \frac{T_{k-1}(\frac{h}{2}) - T_{k-1}(h)}{4^k - 1}$$

$$T_0^m = \frac{T_0^{m-1}}{2} + \frac{h}{2^m} \sum_{i=1}^{2^{m-1}} f(a + (2i-1)\frac{h}{2^m})$$

3. Retrouver l'algorithme de Romberg.

Algorithm 4: Algorithme de Romberg

Data: a, b, ϵ (précision), n_{max}
$h = b - a$
$T_1 = \frac{h}{2}(f(a) + f(b))$
$k = 0$
$k = 0$
while $|T(1) - T_d| > \epsilon$ **do**
  $k = k + 1$
  calcul de T_0^m :
  $if = 2^{k-1}$
  $s = 0$
  for i=1,...,if **do**
    $s = s + f(a + (i - 0,5) * \frac{h}{if})$
  end
  $T(k+1) = \frac{T(k)}{2} + \frac{h}{2if} * s$
  $T_d = T(1)$ (Td=T_{m-1}^0 qu'on sauve)
  calcul de T_m^0 :
  $q = 1$
  for i=k,...,1 *par -1* **do**
    $q = 4 * q$
    $T(i) = T(i+1) + \frac{T(i+1) - T(i)}{q-1}$
  end
end
Result: $T(1)$: approximation de l'intégrale cherchée

4. Ecrire un code Arduino DUE correspondant.
On introduit la variable $rh = \frac{h}{2^{k-1}} = \frac{h}{if}$, ce qui simplifie l'algorithme.
cf le code romberg.ino pour Arduino DUE.