Université de Nice
Département de Mathématiques
NOM :
PRENOM :

L3MASS et L3 Maths, année 2012-2013

Date :
Groupe :

**Feuille-question du TP 4**
**Les algorithmes d'Euler et de Runge-Kutta du 2ème ordre**

# 1 Programmation de l'algorithme d'Euler

On appelle algorithme de résolution d'une équation différentielle ordinaire $y' = f(t,y)$ une fonction $(t,y) \mapsto \Phi(t,y\,;h)$ qui doit être une bonne approximation $\tilde{y}(t+h)$ de la solution exacte $\overline{y}$ de l'équation qui vérifie $\overline{y}(t+h) = y$. Le nombre $h$ s'appelle le *pas d'intégration.* L'idée est de partir d'une *condition initiale* $(t_0, y_0)$ et de considérer la suite des $t_i = t_{i-1}+h = t_0+i*h$ et des $y_i = \Phi(t_{i-1}, y_{i-i}, h)$, en espérant que $y_i$ soit une bonne approximation de la valeur de la solution $\overline{y}(t_i)$ telle que $\overline{y}(t_0) = y_0$. On est satisfait si, tout chaque $T$ fixé, et pour $h := T/N$ la limite, lorsque $N$ devient grand (et donc le pas $h$ devient petit), de la différence $\Delta = \overline{y}(T) - y_N$ entre valeur exacte et approximation tends vers zéro, et d'autant plus si cette convergence est "rapide", c'est à dire qu'il n'est pas nécessaire de choisir $N$ trop grand pour atteindre une précision souhaitée. L'algorithme "de la droite tantente" dû à Euler (1707-1783) consiste à poser

$$\Phi(t,y,h) = y + hf(t,y). \tag{1}$$

Nous allons l'expérimenter tout d'abord sur l'équation différentielle $y' = -0.5y$. Rappeler quelle est la solution de cette équation de condition initiale $(0,2)$ et calculer sa valeur en $t = 3$.

La solution $t \mapsto y(t)$ de (1) telle que $y(0)=2$ est $y(t)=2e^{-0.5t}$
Sa valeur en $t=3$ est $y(3) = 2*exp(-0.5*3) = 0.4462603$

Saisissez les lignes ci-dessus pour avoir une représentation géométrique de cette solution entre $t = 0$ et $t = 3$.

```
function f=f(t,y) ; f=-0.5*y ; endfunction ;
t0=0;y0=2;t=0:0.1:3 ;
sol=ode(y0,t0,t,f) ;
xset("window",1);plot(t,sol) ;
```

Combien vaut cette solution en $t = 0.6$ et en $t = 1$ ? Expliquer comment vous calculez ces valeurs.

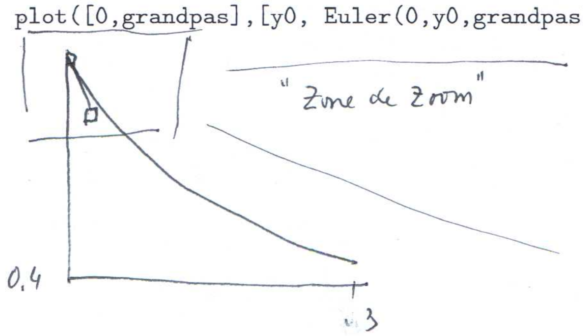
$y(0.6)$ vaut $2*exp(-0.5*0.6) = 1.4816364$
Sur la figure on lit 1,4..
Comme $0.6 = 0,1*6$, c'est aussi sol(1+6)=1.4816365
On observe une légère différence, de $10^{-7}$

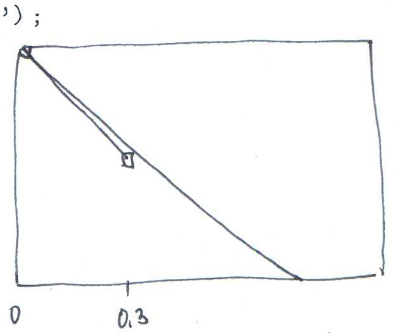
Voici comment coder l'algorithme d'Euler sous forme d'une fonction (sur un seul pas pour commencer) :

```
Tmax=3;
N=100 ;petitpas=Tmax/N ;
M=10 ;grandpas=Tmax/M ;
function y=Euler(t0,y0,pas) ;
      y=y0+pas*f(t0,y0) ;
endfunction ;
```

Saisissez-la dans scilab. L'instruction suivante permet alors de représenter le premier pas de l'algorithme, pour $h = 0.3$ choisi volontairement assez grand pour permettre de visualiser le résultat ; executez-la et donner un schéma du tracé que vous obtenez.

```
plot([0,grandpas],[y0, Euler(0,y0,grandpas)],'r-o') ;
```



Les instructions suivantes répètent cet algorithme jusqu'à atteindre **Tmax**. Executez-les et observez l'écart entre la solution et son approximation.

```
pas=grandpas ;
tt=t0:pas:Tmax ;
yy=zeros(1+Tmax/pas) ;yy(1)=y0 ;
for i=1:1:Tmax/pas ;
      yy(i+1)=Euler(tt(i),yy(i),pas) ;
end ;
plot(tt,yy,'g->') ;
disp(sol(M+1)-yy(1+Tmax/pas),'difference=',pas,'pas=') ; //notez l'ordre inverse
```

Ici il convient d'abord de poser
$t = 0:0.3:3;$
$sol = ode(y0, t0, t, f);$
ou directement
$sol = ode(y0, t0, tt, f)$

Quel écart $\Delta$ trouvez-vous entre la valeur de la solution à l'instant 3 $y(3)$ et son approximation $y_M$ =yy(1+M) ?

$\Delta_{0.3} =$ 0,052116

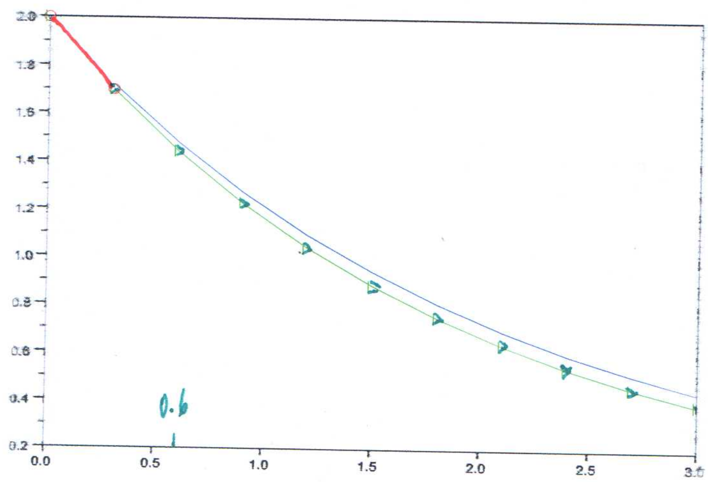
Expliquez pourquoi on a bien $\tilde{y}(T_{Max})$ =yy(M+1)=yy(1+Tmax/pas).

$\tilde{y}(T_{max})$ est la valeur de la solution approchée au point $T_{max}$ avec un pas = grandpas (= 0.3); il faut M pas pour atteindre $T_{max}$ ; cette valeur est mémorisée dans la M+1ème place dans yy, d'où $\tilde{y}(T_{max})$ = yy(M+1). Comme pas = grandpas = Tmax / M, on a bien M = Tmax/pas, et donc yy(M+1) = yy(1+Tmax/pas).

Représentez avec soin ci-dessous le dessin que vous obtenez ; indiquez sur votre dessin quelle valeur approximative de $\tilde{y}(0.6)$ vous lisez sur le dessin

Quelle est la valeur exacte de $\tilde{y}(0.6)$ trouvée par l'algorithme pour cet $t = 0.6$

$\tilde{y}(0.6) =$ yy(1+2) = yy(3) = 1.445 car 0.6 = 2 * pas = 2 * grandpas
C'est le deuxième point (triangle vert) calculé.



Reprenez cette étude avec $h = 0.03$ :

```
// et maintenant avec un petit pas
pas=petitpas ;
```

Utilisez la commande ci-dessous pour obtenir que le nouveau tracé soit en rouge

```
plot(tt,yy,'r-.') ;
```

Qu'observez-vous ? L'approximation en rouge est plus proche de la solution exacte, en bleu.

Notons $\Delta_{0.3}$ et $\Delta_{0.03}$ les écarts, en $t = 3$, entre solution exacte et solution approchée pour $h = 0.3$ et $h = 0.03$. Donnez vos résultats :

$\Delta_{0.3} =$ 0,052116

$\Delta_{0.03} =$ 0,0050425

$\Delta_{0.03}/\Delta_{0.3} =$ 0,0960276 de l'ordre de petitpas/grandpas.

Comment évolue cette erreur avec la taille du pas choisi ?

Elle évolue comme le rapport des tailles 0.03/0.3 = 0.10

Reprenez cette étude pour l'équation différentielle $y' = -y^2$ sur une nouvelle figure :

```
////////////////////////// on recommence avec -y^2
xset("window",2) ;
```

Notons à nouveau $\Delta_{0.3}$ et $\Delta_{0.03}$ les écarts, en $t = 3$, entre solution exacte et solution approchée pour $h = 0.3$ et $h = 0.03$. Donnez vos résultats :

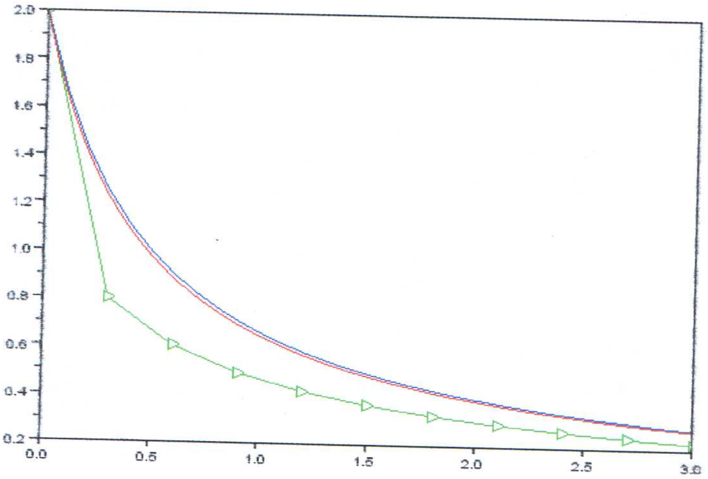
$\Delta_{0.3} =$ 0,0562388

$\Delta_{0.03} =$ 0,0047914

$\Delta_{0.03}/\Delta_{0.3} =$ 0.0851968

Comment évolue cette erreur avec la taille du pas choisi ?

Elle évolue comme le rapport des tailles 0.03/0.3 = 0,10 ≈ 0.085



## 2 Méthode de Runge-Kutta du 2ème ordre

Cette méthode est meilleure que la méthode d'Euler : c'est ce que nous allons expérimenter ici. Elle est inspirée de la méthode du point milieu pour l'intégrale et consiste à poser

$$\Phi(t,y,h) = y + hf(t+h/2, y+hf(t,y)/2). \tag{2}$$

Ci-dessous son expression sous forme d'une fonction scilab :

```
/////////// On reprend y'=-0.5y avec la methode de RK2
//RK(t0,y0,pas) est une approximation de y(t0+pas) pour y(t0)=y0.
function y=RK(t0,y0,pas) ;
// variables locales: k1,k2 ;
      k1=pas*f(t0,y0) ;
      k2=pas*f(t0+pas/2,y0+k1/2) ;
      y=y0+k2 ;
endfunction ;
```

Mise en oeuvre :

```
xset("window",3) ;
t=0:petitpas:Tmax ;
sol=ode(y0,t0,t,f) ;
plot(t,sol) ;//la solution exacte
plot([t0,grandpas],[y0,RK(t0,y0,grandpas)],'r-o') ;//methode sur un seul pas
pas=grandpas ;//a remplacer par petitpas pour h=0.03
tt=t0:pas:Tmax ;
yy=zeros(1+Tmax/pas) ;yy(1)=y0 ;
for i=1:1:Tmax/pas ;
      yy(i+1)=RK(tt(i),yy(i),pas) ;
end ;
plot(tt,yy,'g->') ;// remplacer 'g->' par 'r-.' pour h=0.03
disp(sol(N+1)-yy(1+Tmax/pas),'difference=',pas,'pas=') ; //notez l'ordre inverse
```

Notons à nouveau $\Delta_{0.3}$ et $\Delta_{0.03}$ les écarts, en $t = 3$, entre solution exacte et solution approchée pour $h = 0.3$ et $h = 0.03$. Donnez vos résultats :

$\Delta_{0.3} =$ $-0.0028182$

$\Delta_{0.03} =$ $-0,0000253$

$\Delta_{0.03}/\Delta_{0.3} =$ $0.0089933$

Comment évolue cette erreur avec la taille du pas choisi ?

Le rapport (0.00899..) des erreurs est de l'ordre de 0,01, soit $(0,1)^2 = \left(\frac{\text{petitpas}}{\text{grandpas}}\right)^2$

c'est l'effet pratique du fait que RK2 est une méthode d'ordre 2.

Notons pour finir que la commande `ode` utilise bien entendu elle aussi un algorithme de calcul approché de la solution. Calculez l'erreur entre la solution exacte (calculée à la première question) et la solution approchée.

Utilisez le "help" de `scilab` pour determiner quel est l'algorithme utilisé par `ode`.