Équations différentielles (suite)

1er février 2016

1 Gérer des EDO d'ordre 2, 3, ...

Considérons l'équation d'un amortisseur soumis à une excitation de pulsation ω :

$$my'' = -cy' - ky + \alpha \cos \omega t$$

Ce n'est pas une équation d'ordre 1. Comment la gérer ?

On introduit $Y(t) = (y(t), y'(t))$.

En posant $\pi_0 : (x, y) \mapsto x$ et $\pi_1 : (x, y) \mapsto y$:

$$\begin{aligned} Y'(t) &= (y'(t), y''(t)) \\ &= (y'(t), (-cy'(t) - ky(t) + \alpha \cos \omega t)m^{-1}) \\ &= (\pi_1(Y(t)), (-c\pi_1(Y(t)) - k\pi_0(Y(t)) + \alpha \cos \omega t)m^{-1}) \\ &= F(Y(t), t) \end{aligned}$$

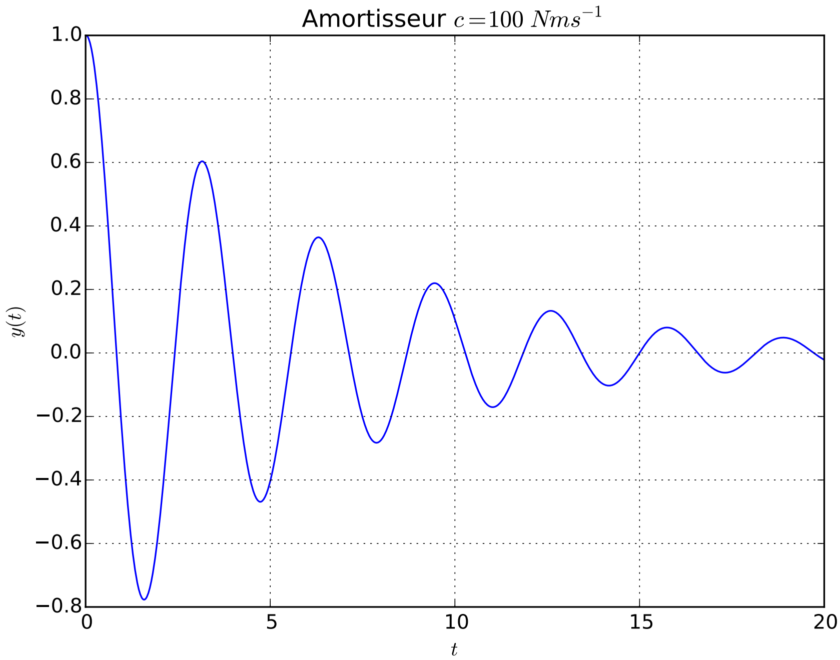
où $F : (Y, t) \mapsto (\pi_1(Y), (-c\pi_1(Y) - k\pi_0(Y) + \alpha \cos \omega t)m^{-1})$

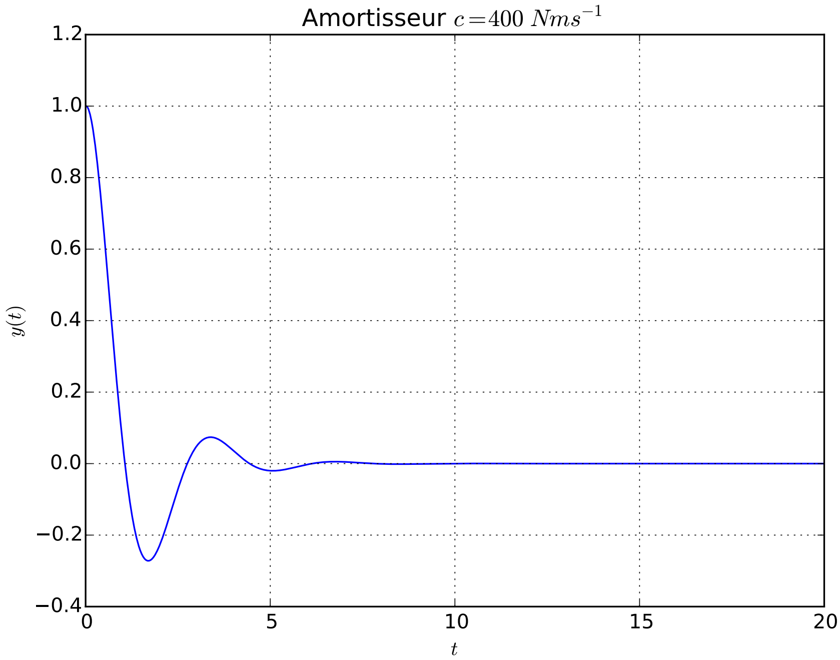
On s'est ramené à une équation d'ordre 1 !

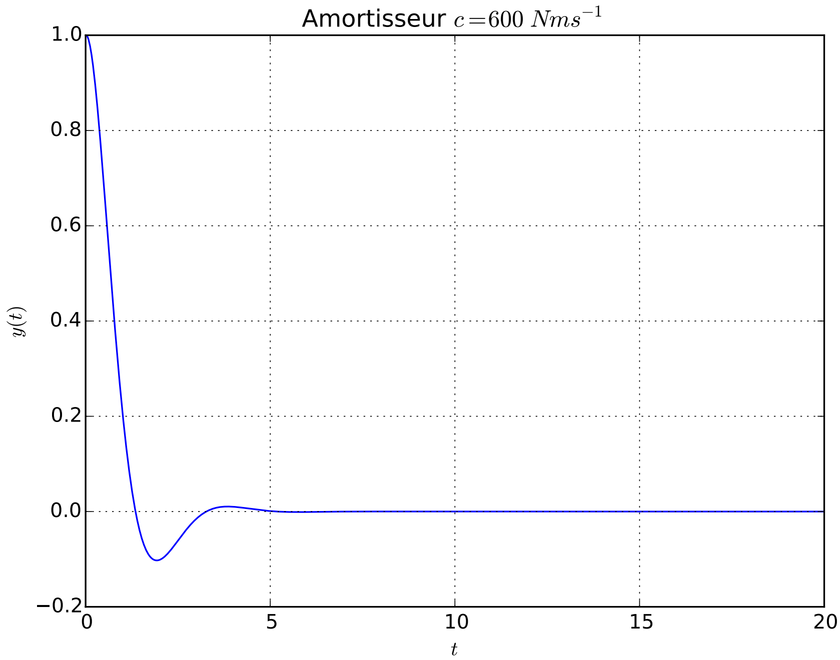
Résolution :

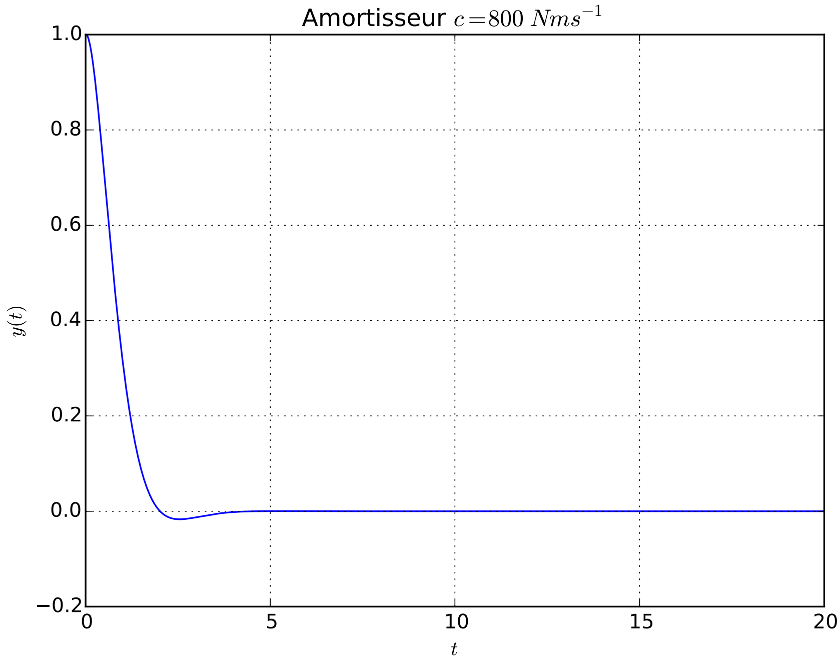
```
>>> m = 250 # kg
>>> omega = 1 # s ** (-1)
>>> alpha = 0 # N
>>> k = 1000 # N / m
>>> c = 500 # N * m * s**(-1)
>>> y0 = 1 # m
>>> yp0 = 0 # m . s**(-1)
>>> t0 = 0
>>> t1 = 20 # s
>>> n = 1000
>>> h = float(t1 - t0) / n
```

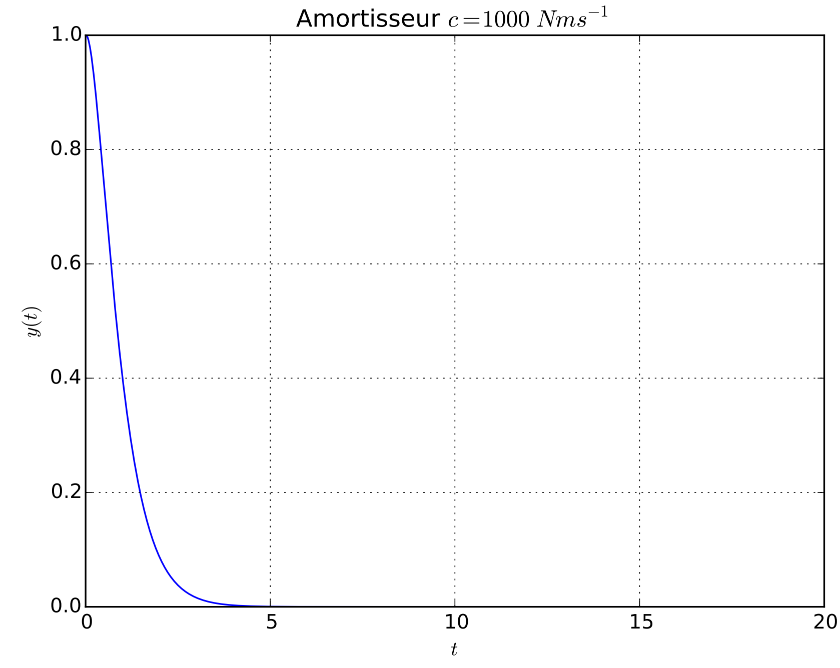
```
>>> def F(Y, t) :
...   y, yp = Y
...   ypp = (-c*yp - k*y + alpha*cos(omega*t)) / m
...   return array([yp, ypp])
... les_t, les_y = euler_vectoriel(F, t0, t1,
>>>   array([y0, yp0]), h)
```

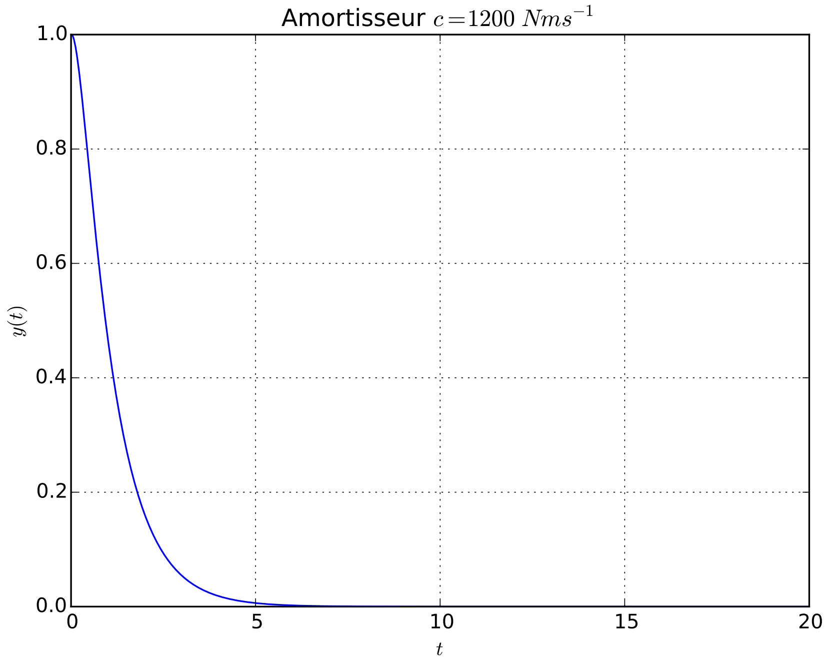


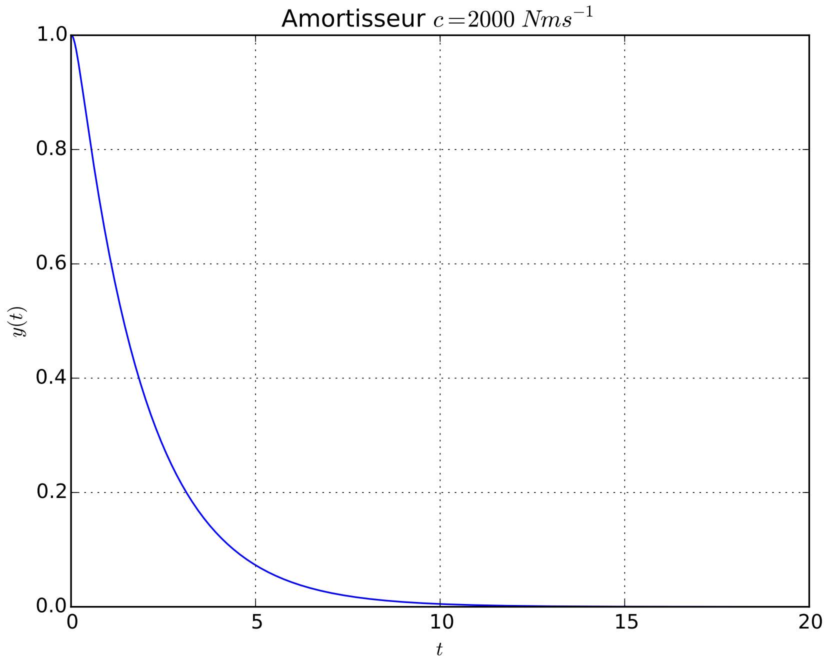

Amortisseur $c = 400\ Nms^{-1}$
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
$y(t)$
0
5
10
15
20
t


Amortisseur $c = 600\ Nms^{-1}$
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
$y(t)$
0
5
10
15
20
t


Amortisseur $c = 800\ Nms^{-1}$
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
$y(t)$
0
5
10
15
20
t


Amortisseur $c = 1000\ Nms^{-1}$
1.0
0.8
0.6
0.4
0.2
0.0
$y(t)$
0
5
10
15
20
t

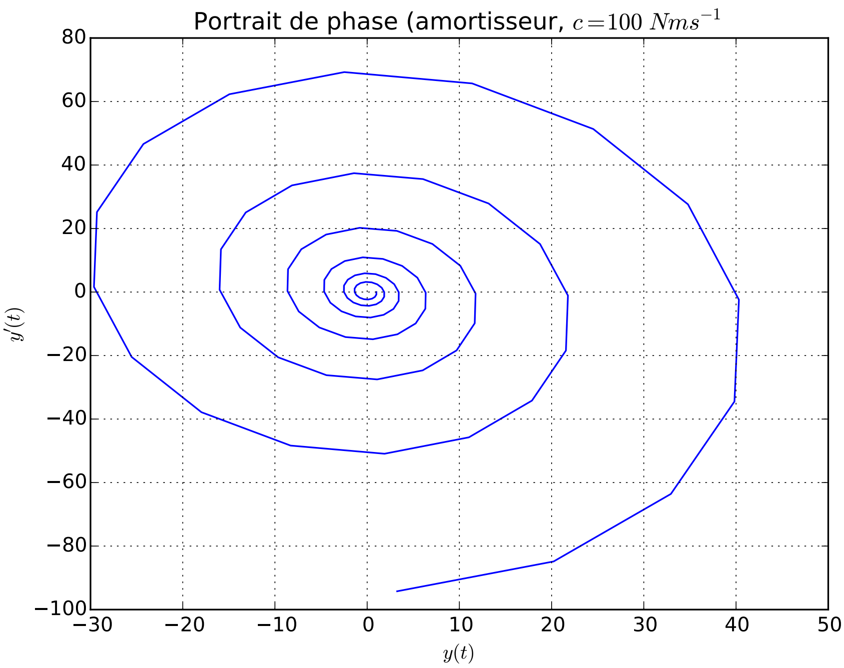




2 Portraits de phase

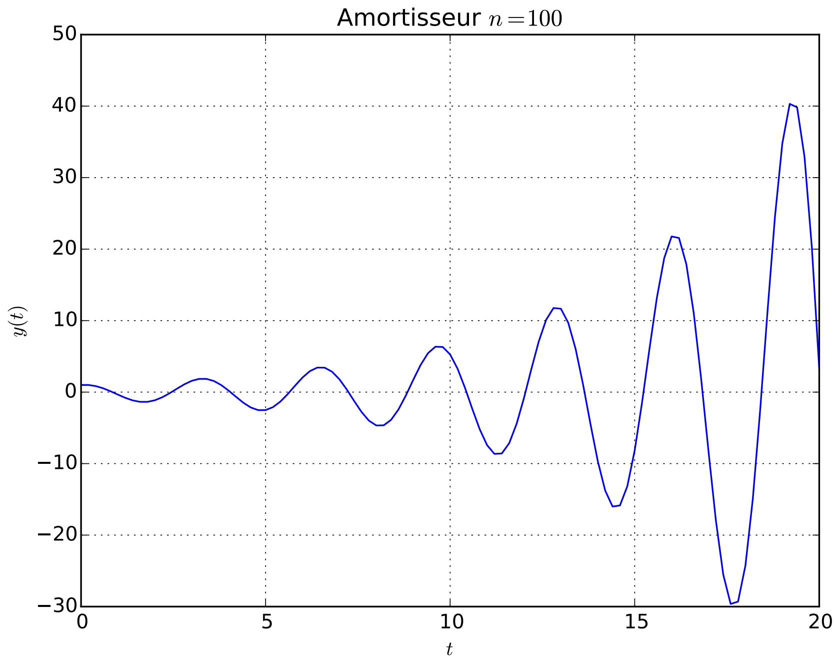
On porte sur le graphe les couples $(y(t), y'(t))$:

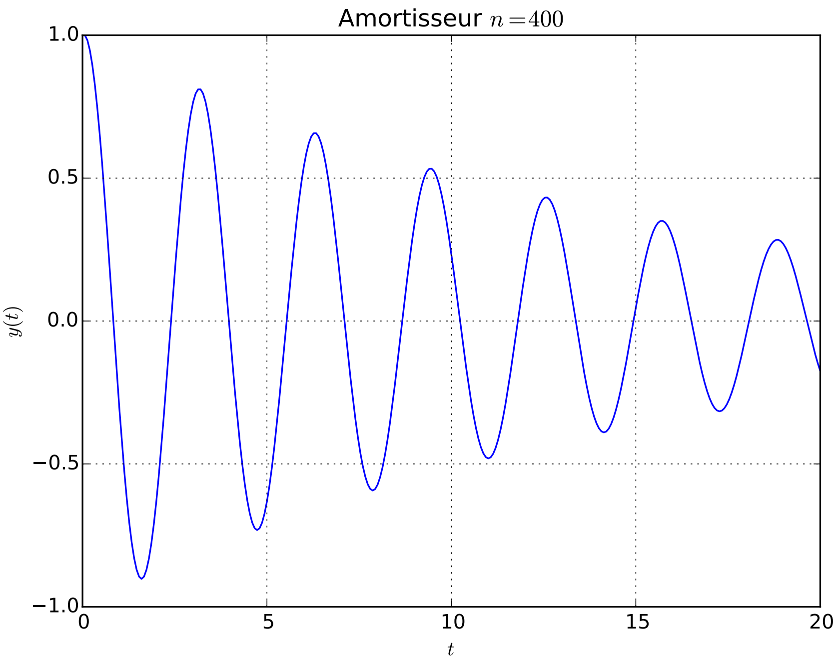
```
>>> def phase(legende, nomfichier) :
...     _, les_y = euler_vectoriel(F, t0, t1,
...         array([y0, yp0]), h)
...     pl.clf()
...     pl.xlabel('$y(t)$')
...     pl.ylabel("$y'(t)$")
...     pl.grid()
...     pl.plot(array(les_y)[:, 0], array(les_y)[:, 1])
...     pl.title(legende)
...     pl.savefig(nomfichier)
```

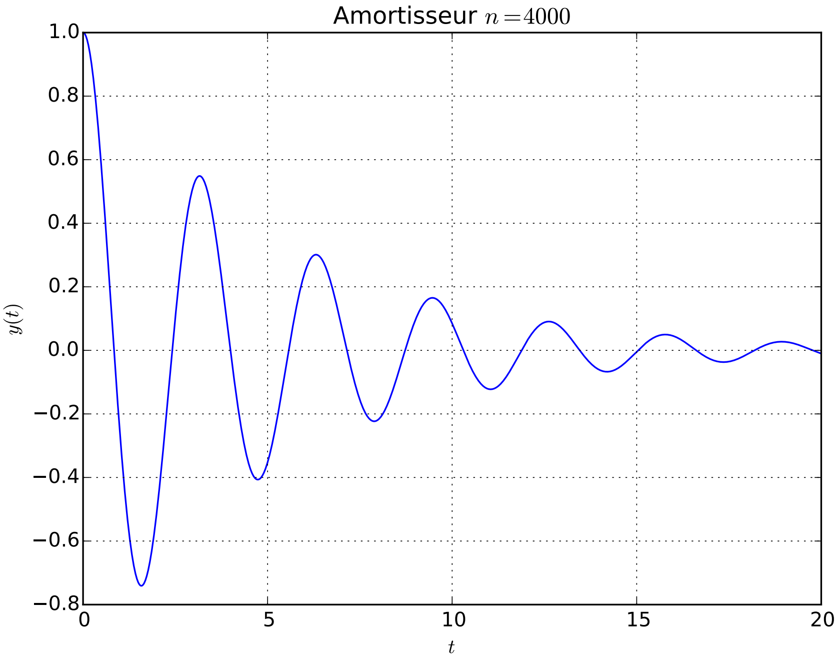


3 Influence du pas de discrétisation

On reprend l'exemple de l'amortisseur, avec $c = 100$ et avec différentes valeurs du pas de discrétisation $h = (t_1 - t_0)/n$:


Amortisseur $n = 100$
50
40
30
20
10
0
−10
−20
−30
0
5
10
15
20
t
$y(t)$


Amortisseur $n = 400$
1.0
0.5
0.0
−0.5
−1.0
0
5
10
15
20
t
$y(t)$



Le premier graphe est clairement délirant sur le plan physique ...
Plusieurs facteurs contribuent à l'erreur :

— des *erreurs de méthode* (la méthode d'Euler n'est pas parfaite)

— des *erreurs de calcul* (les flottants ne sont pas les réels)

Pour les erreurs de méthode, on dit qu'il y a *convergence* si :

1. la somme $e(h) = \sum_{k=0}^{n} |y(t_k) - y_k|$ des petites erreurs commises à chaque étape tend vers 0 quand h tend vers 0 ($e(h)$ est appelée *erreur de consistance relative à la solution* y).
2. et certaines conditions de *stabilité* de la méthode employée sont respectées (essentiellement, si l'on commet une petite erreur à chaque étape du calcul des y_k, menant à de nouvelles approximations $\tilde{y}_k$, alors si l'erreur de consistance relative aux y_k est petite, l'erreur entre les y_k et les $\tilde{y}_k$ doit être petite aussi).

On dit qu'une méthode est d'*ordre* p s'il existe une constante $K > 0$ telle que $e(h) \leq Kh^p$, autrement dit l'erreur de consistance est un $O(h^p)$.
La méthode d'Euler est une méthode d'ordre 1.

4 Autres méthodes

4.1 Méthode d'Euler implicite

La méthode d'Euler présentée jusqu'ici est dite *explicite* : y_{k+1} ne dépend que y_k et t_k.

$$y_{k+1} = y_k + hF(y_k, t_k)$$

Mais en reprenant les approximations qui ont conduit à ce schéma, nous pouvons aussi bien écrire :

$$y_{k+1} = y_k + hF(y_{k+1}, t_k)$$

ce qui mène à la méthode d'Euler *implicite*.

Elle a un gros inconvénient : pour trouver y_{k+1}, il faut résoudre une équation (numériquement).

Mais elle a un avantage : elle est souvent plus stable.

4.2 Méthode de Heun

Si $y_n = y(t_n)$, il existe $c \in]t_n, t_{n+1}[$ tel que

$$y(t_{n+1}) = y_n + hy'(c)$$

Tout le problème est d'estimer $y'(c)$.

On connaît l'idée de la méthode d'Euler explicite :

$$y'(c) \approx y'(t_n)$$

Donc on pose $\boxed{k_1 = F(y_n, t_n)}$.

Puis

$$\boxed{y_{n+1} = y_n + hk_1}$$

(erreur locale en $O(h^2)$)

L'idée de la méthode de Heun est la suivante :

$$y'(c) \approx \frac{y'(t_n) + y'(t_{n+1})}{2}$$

On pose $\boxed{k_1 = F(y_n, t_n)}$. Alors $k_1 = y'(t_n)$.

On estime $y'(t_{n+1})$ en utilisant $y'(t_{n+1}) = F(y(t_{n+1}), t_{n+1})$.

Pour cela, on estime $y(t_{n+1})$ par la méthode d'Euler explicite.

On pose donc : $\boxed{k_2 = F(y_n + hk_1, t_n + h)}$ (erreur en $O(h^2)$). Puis

$$\boxed{y_{n+1} = y_n + h\left(\frac{k_1 + k_2}{2}\right)}$$

On peut alors montrer que l'erreur locale est en $O(h^3)$.

4.3 Méthode de Runge-Kutta

Pour évaluer $y'(c)$, on calcule

$$\frac{k_1 + 2k_2 + 2k_3 + k_4}{6}$$

où k_1, k_2, k_3 et k_4 sont des évaluations des pentes :

k_1 pente en y_n

k_2 pente évaluée en $t_n + \frac{h}{2}$ en utilisant k_1 pour estimer $y(t_n + \frac{h}{2})$

k_3 pente évaluée en $y_n + \frac{h}{2}$ en utilisant k_2 pour estimer $y(t_n + \frac{h}{2})$

k_4 pente évaluée en $t_n + h$ en utilisant k_3 pour estimer $y(t_n + h)$.

$$\begin{aligned}
k_1 &= F(y_n, t_n) \\
k_2 &= F(y_n + \frac{h}{2}k_1, t_n + \frac{h}{2}) \\
k_3 &= F(y_n + \frac{h}{2}k_2, t_n + \frac{h}{2}) \\
k_4 &= F(y_n + hk_3, t_n + h) \\
y_{n+1} &= y_n + h\left(\frac{k_1 + 2k_2 + 2k_3 + k_4}{6}\right)
\end{aligned}$$

On peut alors montrer que l'erreur locale est en $O(h^5)$.
Il s'agit d'une méthode d'ordre de convergence 4.
Elle est facile à programmer et donc très populaire.