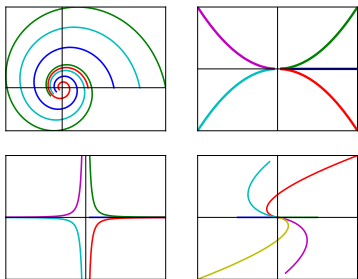


Équations différentielles (suite)



Skander Zannad et Judicaël Courant

Lycée La Martinière-Monplaisir

2014-01-31

1 Gérer des EDO d'ordre 2, 3, ...

Équation d'un amortisseur soumis à une excitation de pulsation ω :

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

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 $\pi_1 : (x, y) \mapsto y$:

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

où $F : (t, Y) \mapsto \dots$

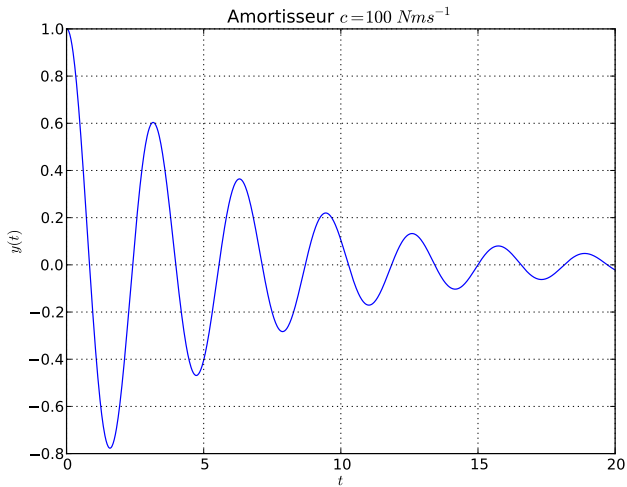
On s'est ramené à une équation d'ordre 1 !

Résolution :

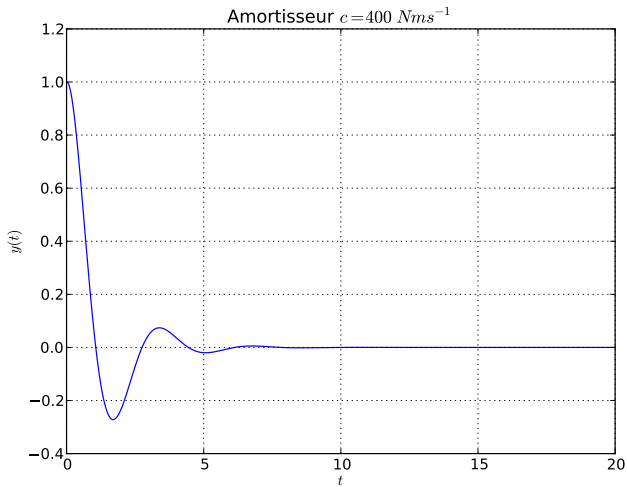
```
m = 250 # kg
omega = 1 # s ** (-1)
alpha = 0 # N
k = 1000 # N / m
c = 500 # N * m * s ** (-1)
y0 = 1 # m
yp0 = 0 # m . s ** (-1)
t0 = 0
t1 = 20 # s
n = 1000
h = float(t1 - t0) / n
```

```
def F(t, Y):  
    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)
```

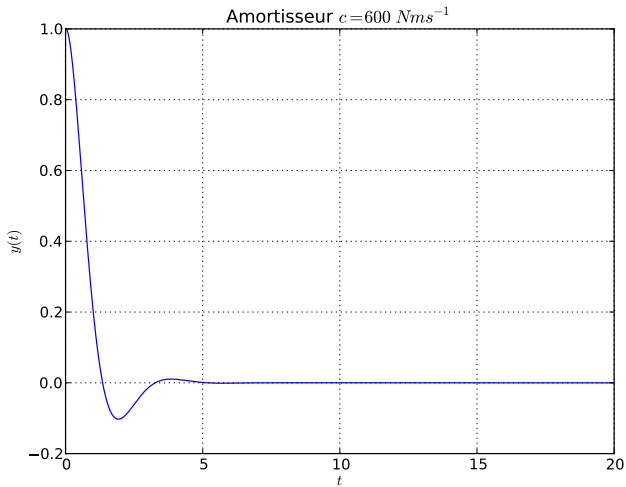
Équations différentielles (suite)



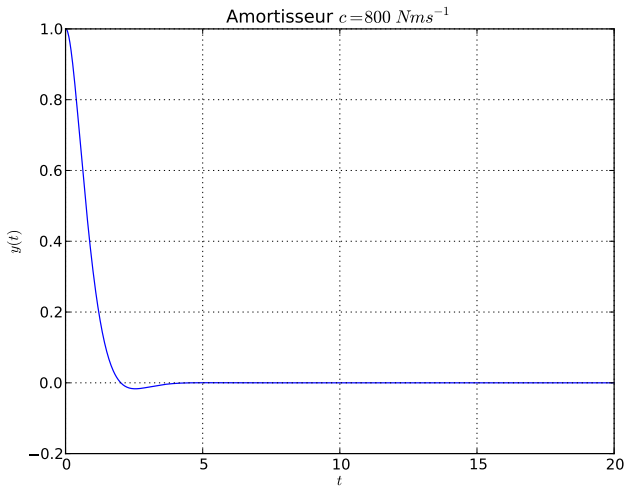
Équations différentielles (suite)



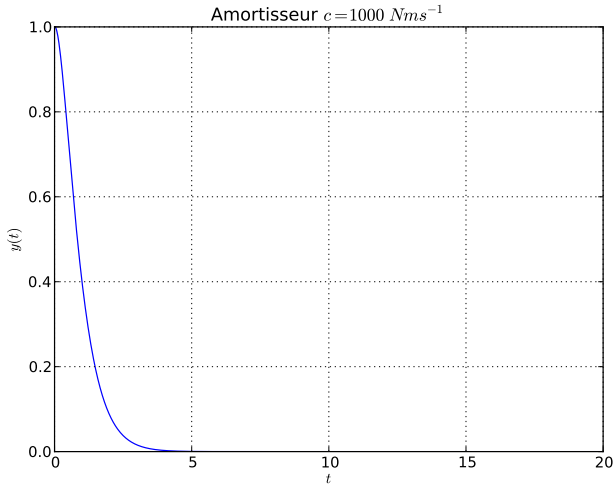
Équations différentielles (suite)



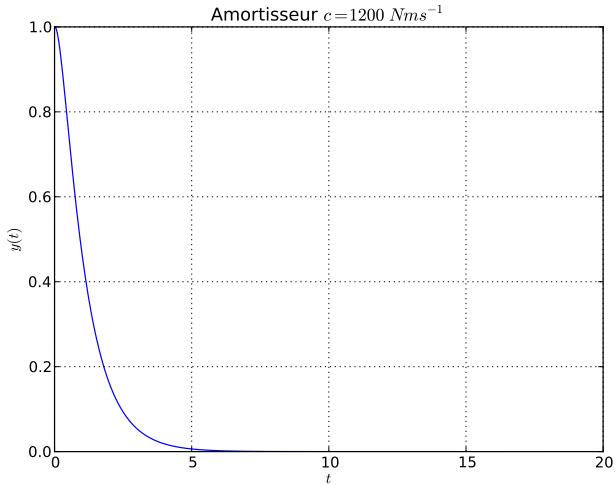
Équations différentielles (suite)



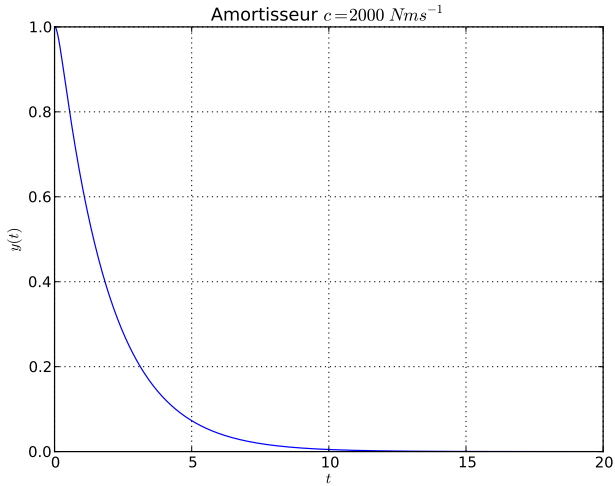
Équations différentielles (suite)



Équations différentielles (suite)



Équations différentielles (suite)

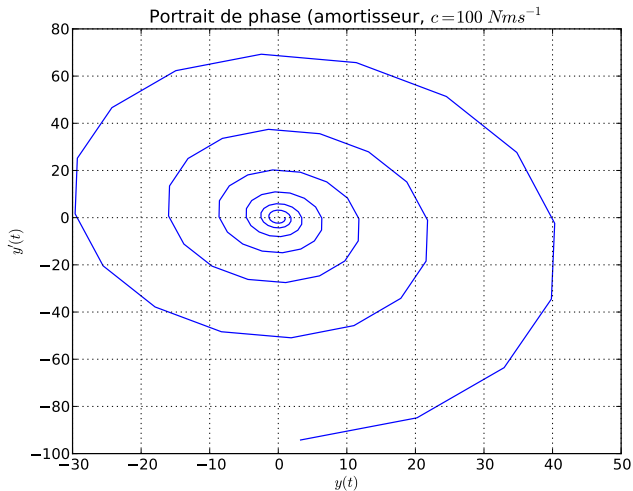


2 Portraits de phase

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)
```

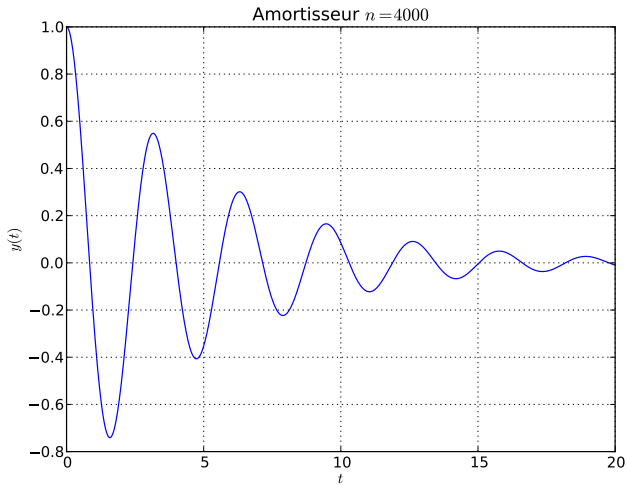
Équations différentielles (suite)



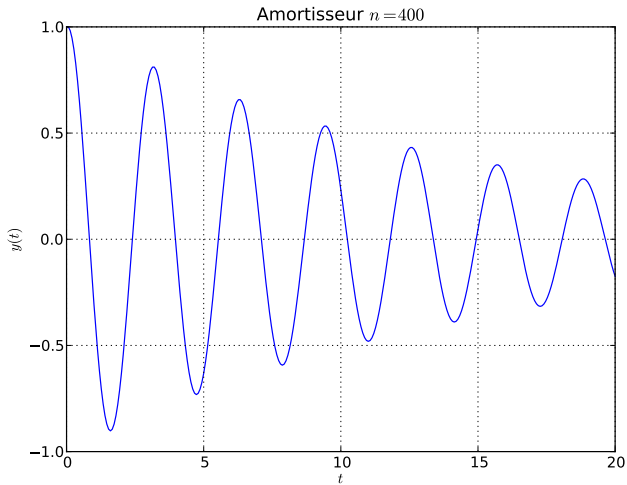
3 Influence du pas de discrétisation

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

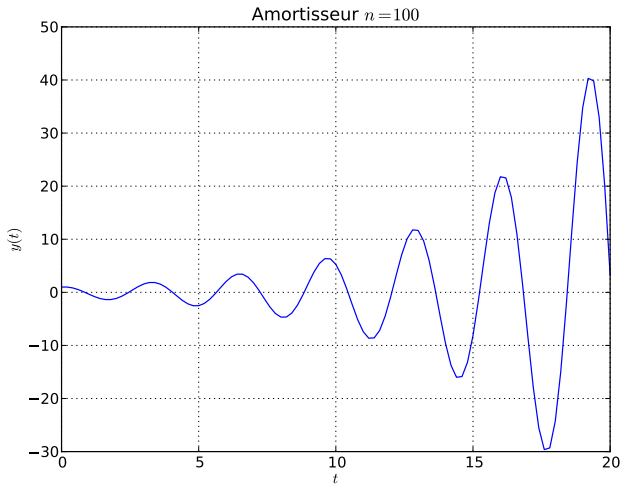
Équations différentielles (suite)



Équations différentielles (suite)



Équations différentielles (suite)



Le dernier graphe est clairement délirant sur le plan physique... (voir livre, notamment fig 9.19 p243)

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, quand n tend vers $+\infty$ (donc quand h tend vers 0), l'erreur de méthode tend vers 0.

Il y a convergence si

1. certaines conditions de stabilité de la méthode employée sont respectée
2. et la somme $e(h)$ des petites erreurs commises à chaque étape tend vers 0 quand h tend vers 0 ($e(h)$: erreur de consistance).

On dit qu'une méthode est d'ordre p si l'erreur de consistance est un $O(Kh^p)$.

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* :

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

Méthode d'Euler implicite

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

Inconvénient : pour trouver y_{k+1} , il faut résoudre une équation (numériquement).

Avantage : 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 $y'(c)$.

Idée de la méthode d'Euler explicite :

$$y'(c) \approx y'(t_n) = k_1 \quad \text{où } k_1 = y'(t_n) = F(t_n, y_n)$$

Idée de la méthode de Heun :

$$\begin{aligned} y'(c) &\approx \frac{y'(t_n) + y'(t_{n+1})}{2} \\ &\approx \frac{F(t_n, y(t_n)) + F(t_{n+1}, y(t_{n+1}))}{2} && (y' = F(t, y)) \\ &\approx \frac{k_1 + F(t_{n+1}, y_n + hk_1)}{2} && (y(t_{n+1}) \approx y_n + hk_1) \end{aligned}$$

On pose donc :

$$y_{n+1} = y_n + \frac{h}{2} \left(F(t_n, y_n) + F(t_{n+1}, y_n + hF(t_n, y_n)) \right)$$

La méthode de Heun est d'ordre (de convergence) deux.

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 $y(t_n + h)$.

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

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

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

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

Il s'agit d'une méthode d'ordre de convergence 4.

Elle est facile à programmer et donc très populaire.