Équation du pendule : méthodes d'Euler

1. Introduction

On considère un système dynamique conservatif dont l'équation différentielle a la forme suivante :

$$\frac{dx}{dt} = v \tag{1}$$
$$\frac{dv}{dt} = a(x) \tag{2}$$

x est une variable de position, v la vitesse correspondante, et $a(x)$ l'accération en fonction de la position.

Un exemple d'équation de ce type est celle du pendule :

$$\frac{dx}{dt} = v \tag{3}$$
$$\frac{dv}{dt} = -4\pi^2 \sin(x) \tag{4}$$

Pour des petites valeurs de x, la linéarisation de cette équation conduit à l'oscillateur harmonique de pulsation 2π.

On se propose d'appliquer différentes méthodes d'Euler à ce système.

2. Méthode d'Euler explicite

Soit (x_0, v_0) une condition initiale pour $t = 0$. Les méthodes d'Euler consistent à calculer des valeurs approchées x_n, y_n de la solution pour cette condition initiale, aux instants suivants :

$$t_n = nh \tag{5}$$

La méthode d'Euler explicite est définie par :

$$x_{n+1} = x_n + hv_n \tag{6}$$
$$v_{n+1} = v_n + ha(x_n) \tag{7}$$

La fonction suivante calcule N points par cette méthode pour l'équation du pendule, et renvoit 3 tableaux numpy :

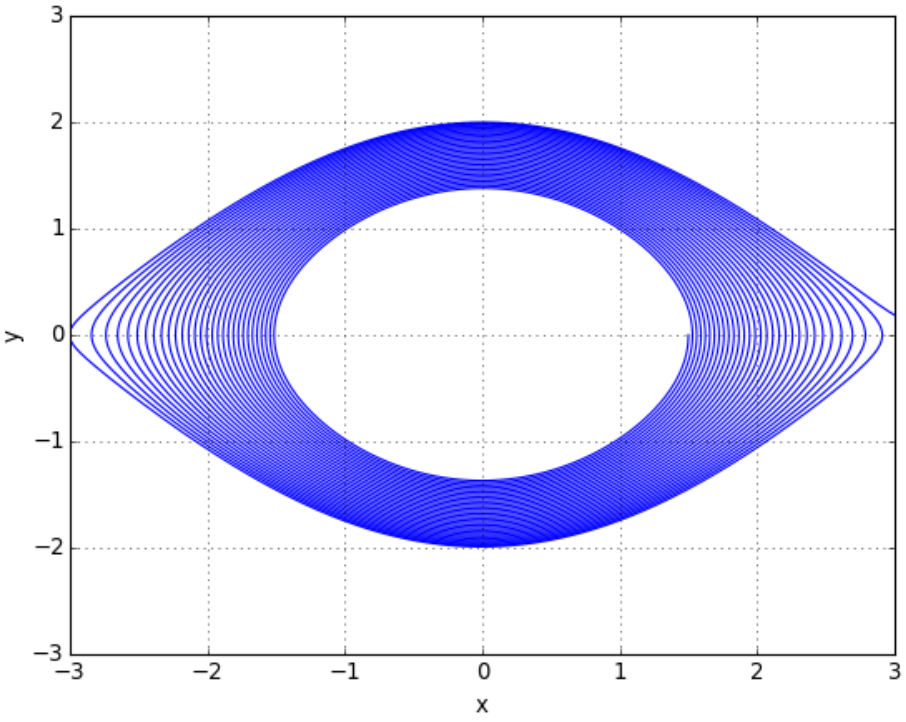
```
def euler(x0,v0,h,N):
    tab_x = numpy.zeros(N)
    tab_v = numpy.zeros(N)
    tab_t = numpy.arange(N)*h
    c=4*math.pi**2
```

```
    x = x0
    v = v0
    for i in range(N):
        tab_x[i] = x
        tab_v[i] = v
        (x,v) = (x+v*h,v-c*math.sin(x)*h)
    return (tab_t,tab_x,tab_v)
```

Voici un exemple avec tracé de la courbe dans le plan de phase (x, v) :

```
h=1e-3
N=100000
x0 = 1.5
v0 = 0
(t,x,v) = euler(x0,v0,h,N)

figure()
plot(x,v/(2*math.pi))
xlabel("x")
ylabel("y")
axis([-3,3,-3,3])
grid()
```



L'amplitude des oscillations augmente au cours du temps (le mouvement n'est pas périodique). Cela met en évidence l'instabilité de la méthode d'Euler explicite pour ce type

de problème. On démontre en effet que cette méthode est instable pour l'oscillateur harmonique, en considérant les valeurs propres se son propagateur P, qui est la matrice permettant de calculer :

$$\begin{pmatrix} x_{n+1} \\ v_{n+1} \end{pmatrix} = P \begin{pmatrix} x_n \\ v_n \end{pmatrix} \tag{8}$$

Cette matrice n'est définie que si le système est linéaire. Pour un système non linéaire, on l'obtient en linéarisant au voisinage d'une position d'équilibre. Pour l'oscillateur harmonique, qui correspond au voisinage de la position $x = 0$ de l'équation du pendule, le propagateur est :

$$P = \begin{pmatrix} 1 & h \\ -\omega^2 h & 1 \end{pmatrix} \tag{9}$$

Nous avons choisi $\omega = 2\pi$.

La méthode est stable si et seulement si toutes les valeurs propres du propagateur (en général complexes) ont un module inférieur ou égal à 1. Pour la justification de cette propriété, voir Intégration des équations du mouvement : méthodes d'Euler.

Dans le cas présent, les deux valeurs propres ont un module strictement supérieur à 1, ce qui démontre l'instabilité de la méthode d'Euler explicite.

3. Méthode d'Euler asymétrique

Pour les système dynamiques conservatifs, il existe une modification de la méthode d'Euler qui permet de la rendre stable :

$$x_{n+1} = x_n + hv_n \tag{10}$$
$$v_{n+1} = v_n + ha(x_{n+1}) \tag{11}$$

La différence avec la méthode explicite est l'évaluation de l'accélération, qui se fait avec la position x_{n+1} et non pas x_n. Cette méthode est dite asymétrique en raison du traitement différent des variables positions et vitesses.

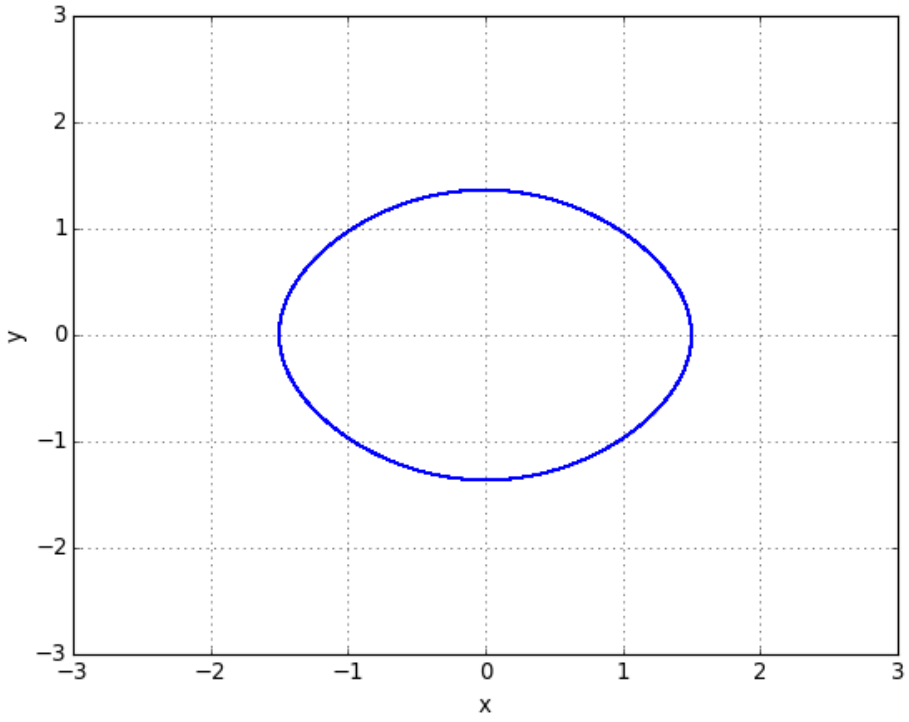
Voici une fonction réalisant cette méthode pour l'équation du pendule, et le résultat du calcul pour les mêmes paramètres que plus haut :

```
def euler_asym(x0,v0,h,N):
    tab_x = numpy.zeros(N)
    tab_v = numpy.zeros(N)
    tab_t = numpy.arange(N)*h
    c=4*math.pi**2
    x = x0
    v = v0
    for i in range(N):
        tab_x[i] = x
        tab_v[i] = v
        x= x+v*h
        v = v-c*math.sin(x)*h
```

```
    return (tab_t,tab_x,tab_v)

(t,x,v) = euler_asym(x0,v0,h,N)

figure()
plot(x,v/(2*math.pi))
xlabel("x")
ylabel("y")
axis([-3,3,-3,3])
grid()
```



On obtient bien en apparence une solution périodique, avec conservation de l'énergie.

Le propagateur pour l'oscillateur harmonique est :

$$P = \begin{pmatrix} 1 & h \\ -\omega^2 h & 1-\omega^2 h^2 \end{pmatrix} \tag{12}$$

La stabilité est vérifiée si $h\omega; < 2$, soit dans notre cas $h < 1/\pi$. Si cette condition est vérifiée, les valeurs propres ont même un module exactement égal à 1, ce qui explique la périodicité de la solution obtenue. C'est la méthode idéale pour les systèmes conservatifs. La méthode de Verlet est une méthode d'ordre 2 qui repose sur ce même principe ; elle est utilisée pour les sytèmes conservatifs à très grand nombre de degrés de libertés, par exemple en dynamique moléculaire.

4. Méthode d'Euler implicite

4.a. Définition

La méthode d'Euler implicite est définie par :

$$x_{n+1} = x_n + hv_{n+1} \tag{13}$$
$$v_{n+1} = v_n + ha(x_{n+1}) \tag{14}$$

Le calcul de x_{n+1}, v_{n+1} nécessite de résoudre l'équation :

$$x_{n+1} + h^2 a(x_{n+1}) = x_n + hv_n \tag{15}$$

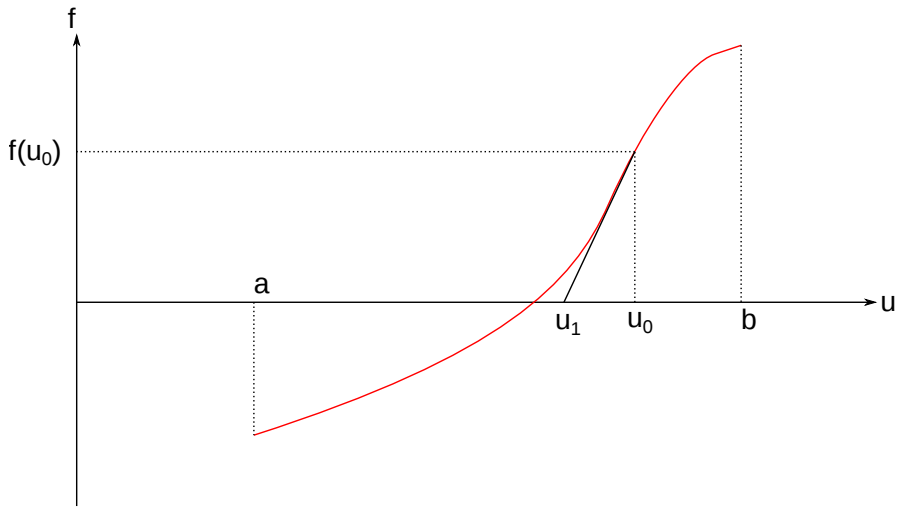
Lorsque le système est non linéaire, comme c'est le cas pour l'équation du pendule, il faut donc résoudre une équation non linéaire à chaque pas de temps. On utilise pour cela la méthode de Newton.

Lorsque x_{n+1} est obtenu, on calcule :

$$v_{n+1} = \frac{x_{n+1} - x_n}{h} \tag{16}$$

4.b. Méthode de Newton

Soit f une fonction continue d'une variable réelle u à valeurs réelles, de classe C^1 sur un intervalle $[a, b]$. On suppose qu'il existe une valeur u_r et une seule sur cet intervalle telle que $f(u_r) = 0$. La méthode de Newton permet d'obtenir une approximation numérique de u_r. On part d'une valeur u_0 dans l'intervalle $[a, b]$ et on considère la tangente à la courbe en ce point. L'intersection de cette tangente avec l'axe Ox donne u_1.



L'opération est répétée, ce qui donne les abcisses u_2, u_3, etc. La suite est définie par la relation de récurrence suivante :

$$u_{n+1} = u_n - \frac{f(u_n)}{f'(u_n)} \tag{17}$$

Cette suite converge vers la racine u_r. La méthode de Newton converge beaucoup plus rapidement que la méthode de dichotomie (mais elle peut dans certains cas échouer). En pratique, on stoppe l'itération lorsque

$$|f(u_p)| < \epsilon \tag{18}$$

où ϵ est une tolérance que l'on choisit en fonction de la précision souhaitée. Pour utiliser la méthode de Newton, il faut connaître la fonction f et sa dérivée f'. Voici une fonction qui implémente la méthode de Newton. On prévoit un nombre maximal d'itérations.

```
def newton(f,df,u0,epsilon):
    u = u0
    y = f(u)
    iter = 0
    while abs(y) > epsilon and iter < 100:
        u = u-y/df(u)
        y = f(u)
        iter += 1
    if iter==100:
        raise Exception("maximum d'iteration atteint")
    print("%d iterations"%iter)
    return u
```

Voici un exemple :

```
def f(u):
    return u*u-2
def df(u):
    return 2*u
r = newton(f,df,1.0,1e-15)


print(r)
--> 1.4142135623730951

print(math.sqrt(2))
--> 1.4142135623730951
```

La convergence vers la solution approchée à la précision du type flottant se fait en seulement 5 itérations. Bien sûr, le nombre d'itérations dépend du point initial. Pour cet exemple, la méthode de Newton échoue si la valeur initiale est $u_0 = 0$, car la tangente en ce point ne coupe pas l'axe Ox.

4.c. Application au pendule

Pour la méthode d'Euler implicite, la fonction dont on cherche la racine est :

$$f(u) = u - h^2 a(u) - (x_n + hv_n) \tag{19}$$

Pour l'équation du pendule, la fonction f et sa dérivée sont :

$$f(u) = u + 4\pi^2 h^2 \sin(u) - (x_n + hv_n) \tag{20}$$

$$f'(u) = 1 + 4\pi^2 h^2 \cos(u) \tag{21}$$

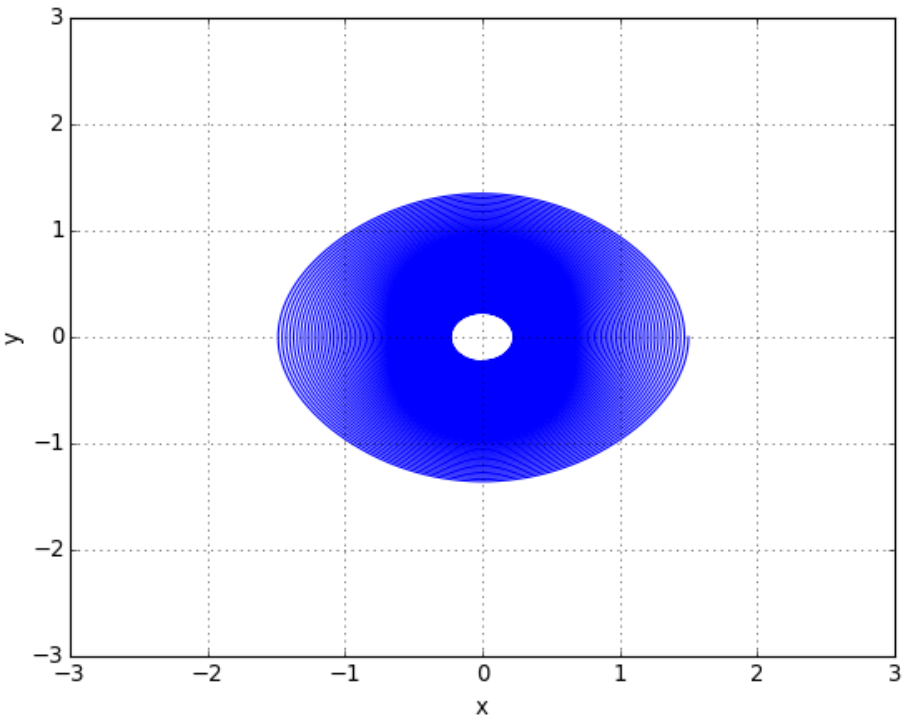
Voici une fonction qui fait le calcul. Elle renvoit aussi le nombre moyen d'itérations dans la méthode de Newton.

```
def euler_implicite(x0,v0,h,N,epsilon):
    tab_x = numpy.zeros(N)
    tab_v = numpy.zeros(N)
    tab_t = numpy.arange(N)*h
    x = x0
    v = v0
    c=4*math.pi**2
    b = c*h**2
    n_iter = 0
    for i in range(N):
        tab_x[i] = x
        tab_v[i] = v
        u = x
        d = x+h*v
        f = u+b*math.sin(u)-d
        while abs(f) > epsilon:
            u = u-f/(1+b*math.cos(u))
            f = u+b*math.sin(u)-d
            n_iter += 1
        v = (u-x)/h
        x = u

    return (tab_t,tab_x,tab_v,n_iter*1.0/N)


(t,x,v,niter) = euler_implicite(x0,v0,h,N,1e-8)

figure()
plot(x,v/(2*math.pi))
xlabel("x")
ylabel("y")
axis([-3,3,-3,3])
grid()
```




```
print(niter)
--> 1.0
```

On voit que la méthode de Newton converge en une seule itération, car la valeur de départ est très proche de la racine.

La solution approchée n'est pas périodique : on observe une réduction de l'amplitude des oscillations, c'est-à-dire une diminution de l'énergie. Pour ce type de problème conservatif, la méthode implicite est donc tout aussi mauvaise que la méthode explicite.

Le propagateur pour l'oscillateur harmonique est :

$$P = \frac{1}{1+\omega^2 h^2} \begin{pmatrix} 1 & h \\ -\omega^2 h & 1-\omega^2 h^2 \end{pmatrix} \tag{22}$$

Ses deux valeurs propres ont un module strictement inférieur à 1. La méthode d'Euler implicite est donc stable (quelque soit le pas de temps). Cependant, elle n'est pas adapté au problème du pendule car la solution obtenue présente une décroissance de l'énergie. La méthode d'Euler asymétrique considérée plus haut n'a pas cet inconvénient, car les valeurs propres ont un module exactement égal à 1.