Oscillateur linéaire amorti à un degré de liberté

Table of Contents

Sébastien SY1924130

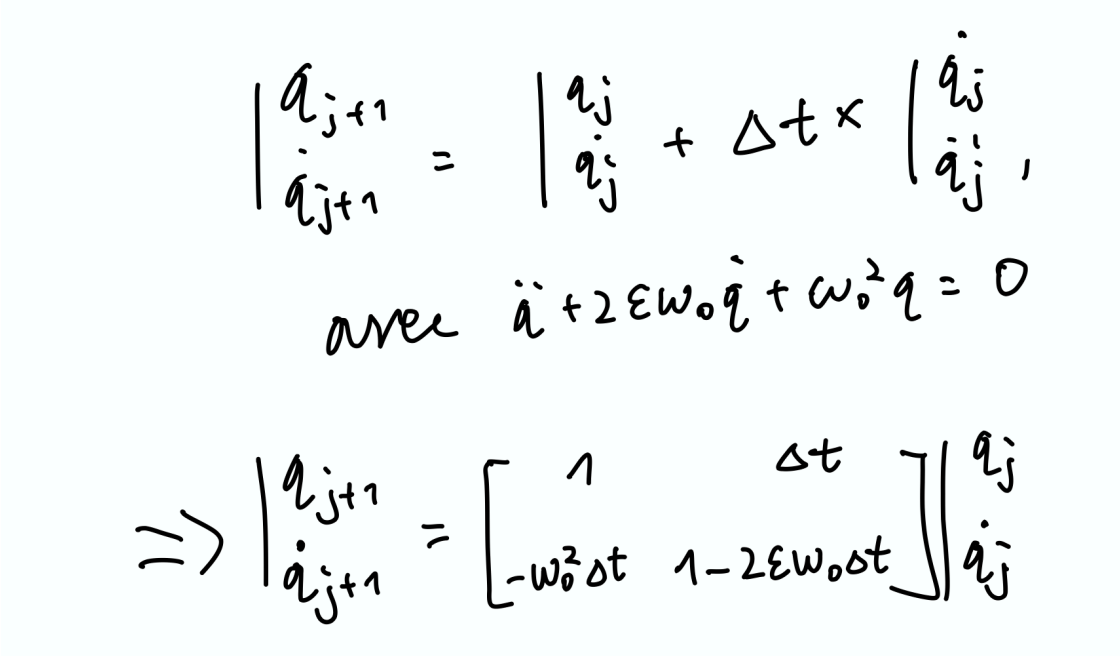
Paramètres

```
clear all
T0 = 1;
global x0 dx0
x0 = 0.01;
dx0 = 0;
omega_0 = 2*pi;
epsilon = 0.02;
warning('off','all');
Gx = gcf;
Gx.Position(3:4) = Gx.Position(3:4)*5;
```

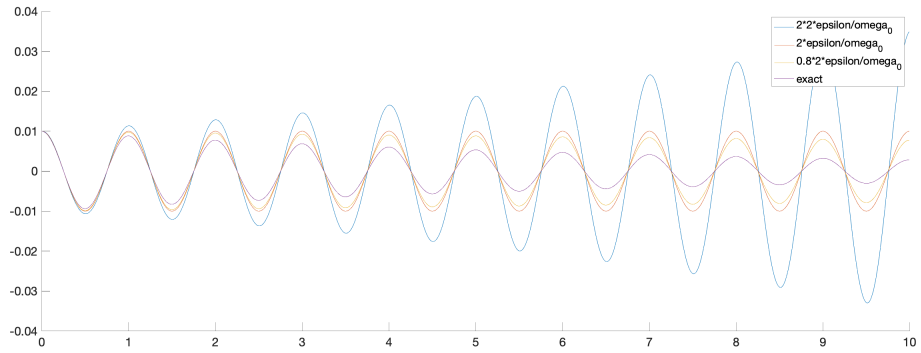
1.1

On utilise la matrice d'amplification d'Euler explicite On peut analyser la précision de ces solutions numériques en comparant avec la solution exact de deux aspects: la précision en période et la précision en amplitude. D'après les résultats on peut voir que les solutions numériques sont tous précises en période, mais pas précises en amplitude. Et celle avec un pas de temps plus petit a une précision plus élevée. En

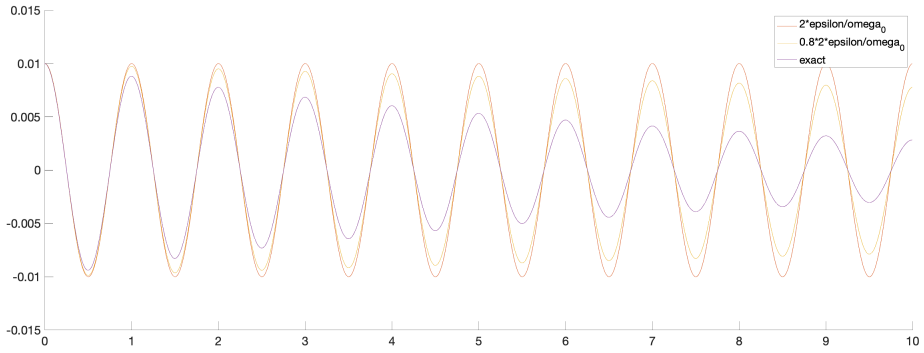
plus, la solution de pas de temps > $\frac{2\epsilon}{\omega_0}$ diverge; la solution de pas de temps = $\frac{2\epsilon}{\omega_0}$ oscille entre 0,01 et -0,01; la solution de pas de temps = $0.8\frac{2\epsilon}{\omega_0}$ converge vers 0.




```
Ax = gca;
Ax.FontSize = Ax.FontSize *3;
syms delta_t
A = [1 delta_t; -omega_0^2*delta_t 1-2*epsilon*omega_0*delta_t];
hold on
for delta_ti=[2*2*epsilon/omega_0 2*epsilon/omega_0 0.8*2*epsilon/
omega_0]
    t = 0:delta_ti:10*T0;
    A_num = double(subs(A,delta_t,delta_ti));
    q = iterate(A_num, t);
    plot(t,q(1,:));
end
Ohm = omega_0*sqrt(1-epsilon^2);
x_exact = exp(-
epsilon.*omega_0.*t).*(x0.*cos(Ohm.*t)+(epsilon.*omega_0.*x0+dx0)/
Ohm.*sin(Ohm.*t));
plot(t,x_exact);
legend('2*2*epsilon/omega_0', '2*epsilon/omega_0', '0.8*2*epsilon/
omega_0', 'exact');
```




```
children = get(gca, 'children');
delete(children(4));
```



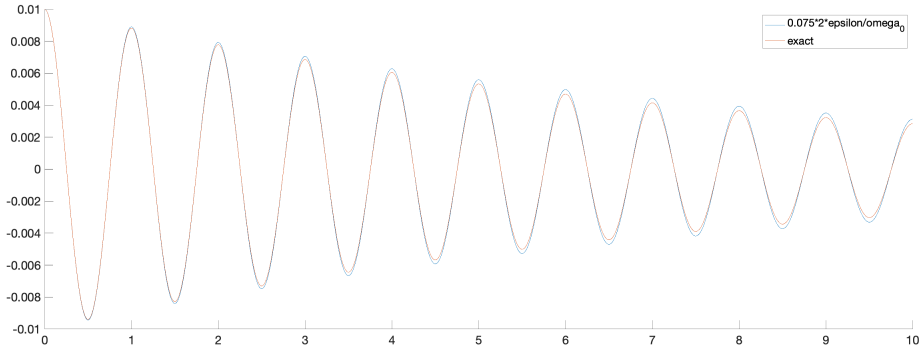
1.1(3d

On définie "une précision suffisante" par "l'erreur relative maximum de l'amplitude est inférieure à 0,1 pour l'intervalle de temps donné". Donc on cherche tous les rapport de delta_t et 2*epsilon/omega_0 inférieur à 1 d'un pas de 0,001 pour trouver le bon rapport. On compare les pics des deux solutions pour calculer les erreurs relatives.

```
clf
hold on
Ax = gca;
Ax.FontSize = Ax.FontSize *3;
for rapport=1:-0.001:0.001
    delta_ti=rapport*2*epsilon/omega_0;
    t = 0:delta_ti:10*T0;
    A_num = double(subs(A,delta_t,delta_ti));
    q = iterate(A_num, t);
    x_exact = exp(-
epsilon.*omega_0.*t).*(x0.*cos(Ohm.*t)+(epsilon.*omega_0.*x0+dx0)/
Ohm.*sin(Ohm.*t));
    erreur = abs((x_exact - q(1,:))./x_exact);
    erreur = erreur(1:floor(10*T0/delta_ti):length(erreur));
    if max(erreur) <= 0.1
```

```
        disp(['À partir de cette valeur du rapport la
 solution calculée présente-t-elle une précision suffisante: '
 num2str(rapport)])
        break
    end
end
plot(t,q(1,:));
plot(t,x_exact);
legend([num2str(rapport) '*2*epsilon/omega_0'], 'exact');
```

À partir de cette valeur du rapport la solution calculée présente-t-elle une précision suffisante: 0.075



1.2

$$\begin{vmatrix} q_{j+1} \\ \dot{q}_{j+1} \end{vmatrix} = \begin{vmatrix} q_j \\ \dot{q}_j \end{vmatrix} + \Delta t \times \begin{vmatrix} \dot{q}_j \\ \ddot{q}_j \end{vmatrix},$$

avec $\ddot{q} + 2\varepsilon\omega_0\dot{q} + \omega_0^2 q = 0$

$$\Rightarrow \begin{vmatrix} q_{j+1} \\ \dot{q}_{j+1} \end{vmatrix} = \begin{bmatrix} 1 & \Delta t \\ -\omega_0^2 \Delta t & 1 - 2\varepsilon\omega_0 \Delta t \end{bmatrix} \begin{vmatrix} q_j \\ \dot{q}_j \end{vmatrix}$$

```
A_imp_inv = [1 -delta_t; omega_0^2*delta_t
 1+2*delta_t*epsilon*omega_0];
A_imp = inv(A_imp_inv);
```

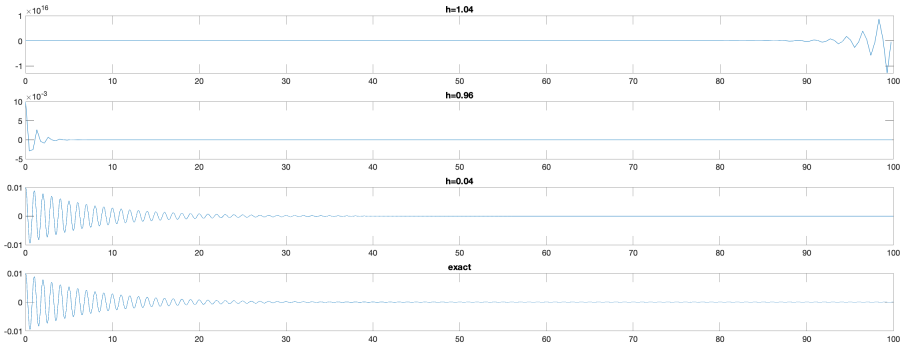
```
[~,d] = eig(A_imp);
vp = simplify(abs(d));
assume(delta_t>0);
solution = solve([vp(1,1)<=1
 vp(2,2)<=1],delta_t,'ReturnConditions',true);
disp(solution.conditions);
% D'après la solution, on peut voir qu'il n'y a de pas de temps
 critique.
% Pour tous les delta_t > 0 les modules des valeurs propres sont
 inférieurs
% à 1.
% Ce schéma est inconditionellement stable.
```

$$0 < x$$

1.3

On peut voir que pour h=1,04, le résultat n'est pas précis ou stable; pour h=0,96, le résultat est stable car il ne diverge pas, mais il n'est pas précis; pour h=0,04, le résultat est stable et précis.

```
clf;
hold on;
Ax = gca;
Ax.FontSize = Ax.FontSize *3;
f = @(y,t) [0 1; -omega_0^2 -2*epsilon*omega_0]*y;
i=1;
for h=[1.04 0.96 0.04]
    delta_t = h * 2*sqrt(2)/omega_0;
    t = 0:delta_t:100*T0;
    q = rk(f, t, delta_t);
    subplot(4,1,i);
    plot(t,q(1,:));
    Ax = gca;
    Ax.FontSize = Ax.FontSize * 2;
    title(['h=' num2str(h)]);
    i=i+1;
end
x_exact = exp(-
epsilon.*omega_0.*t).*(x0.*cos(Ohm.*t)+(epsilon.*omega_0.*x0+dx0)/
Ohm.*sin(Ohm.*t));
subplot(4,1,4);
plot(t,x_exact);
Ax = gca;
Ax.FontSize = Ax.FontSize * 2;
title('exact');
```



1.3b

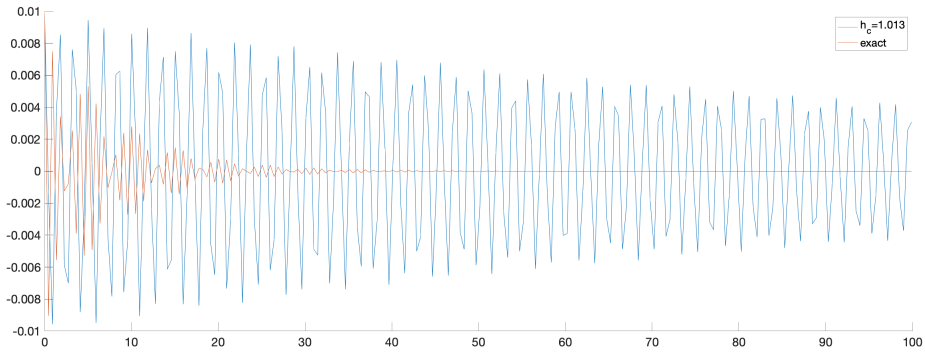
D'après 1.3, on sait que 0,96<h_c<1,04. Donc on essaie tous les h dans cet intervalle avec un pas de 0,001 et examine pour chaque h si la solution diverge ou pas.

```
clf;
hold on;
Ax = gca;
Ax.FontSize = Ax.FontSize *3;
for h=1.04:-0.001:0.96
    delta_t = h * 2*sqrt(2)/omega_0;
    t = 0:delta_t:100*T0;
    q = rk(f, t, delta_t);
    if max(q(1,1:length(t)/2)) >= max(q(1,length(t)/2+1:length(t)))
        disp(['h_c=' num2str(h)]);
        break
    end
    i=i+1;
end
plot(t,q(1,:));
x_exact = exp(-
epsilon.*omega_0.*t).*(x0.*cos(Ohm.*t)+(epsilon.*omega_0.*x0+dx0)/
Ohm.*sin(Ohm.*t));
plot(t,x_exact);
legend(['h_c=' num2str(h)],'exact');
delta_t_c = h*2*sqrt(2)/omega_0
```

```
h_c=1.013

delta_t_c =

    0.4560
```



Fonctions

```
function [U] = iterate(A,t)
%ITERATE U_j+1 = A*U_j, for j in t
i = 0;
U = [];
global x0 dx0
for t_i=t
    if i==0
        U(:,1) = [x0;dx0];
    else
        U(:,i+1)=A*U(:,i);
    end
    i=i+1;
end

end

function [U] = rk(f, t, delta_t)
i = 0;
global x0 dx0
for t_i=t
    if i==0
        U(:,1) = [x0;dx0];
    else
        k1 = f(U(:,i), t_i);
        k2 = f(U(:,i) + k1 * delta_t/2, t_i+delta_t/2);
        k3 = f(U(:,i) + k2 * delta_t/2, t_i+delta_t/2);
        k4 = f(U(:,i) + k3 * delta_t, t_i+delta_t);
        K = (k1 + 2*k2 + 2*k3 + k4) / 6;
        U(:,i+1)=U(:,i) + K * delta_t;
    end
    i=i+1;
end
end
```

Published with MATLAB® R2019b