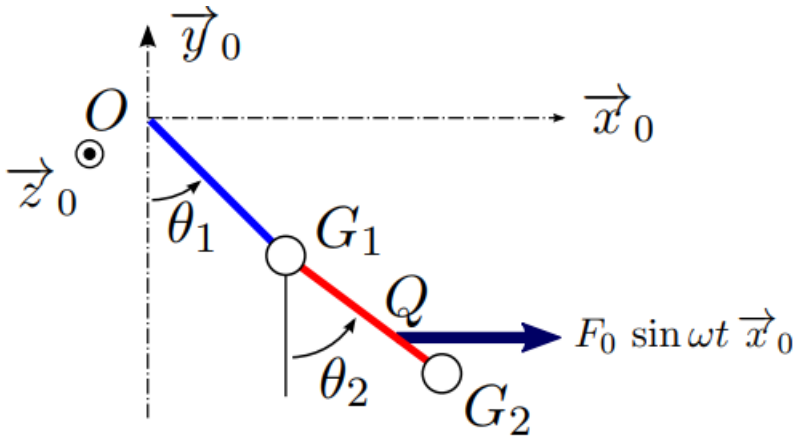
Séance 4 Dynamique des structures, Systems

> Nom Chinois: CAI Pengfei
> Pénom françaus: Vincent
> Numéro d'étudiant: SY1724107

On fournie deux version de code, les codes matlab sont juste après les codes python.

Etude d'un pendule avec l'hypothèse des petits mouvement.



1

1.1) D'abord on a l'équation de mouvement:

$$ma^2 \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix} + mga \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} = F_0 \sin wt \begin{bmatrix} a \\ \frac{a}{\sqrt{2}} \end{bmatrix} \tag{1}$$

Proposon que:

$$M = ma^2 \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}, K = mga \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}, \Phi = F_0 \sin wt \begin{bmatrix} a \\ \frac{a}{\sqrt{2}} \end{bmatrix}, q = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$$

Donc on a :

$$M\ddot{q} + Kq = \Phi$$

Pour simplifier, on note $h = \Delta t$. D'après la méthode Newmark, on a:

$$q_{n+1} = q_n + h\dot{q}_n + h^2(0.5 - \beta)\ddot{q}_n + h^2\beta\ddot{q}_{n+1} \tag{2}$$

$$\dot{q}_{n+1} = \dot{q}_n + h(1 - \gamma)\ddot{q}_n + h\gamma\ddot{q}_{n+1} \tag{3}$$

Donc on a

$$\begin{bmatrix} 1 & 0 & -h^2\beta \\ 0 & 1 & -h\gamma \\ K & 0 & M \end{bmatrix} \begin{bmatrix} q_{n+1} \\ \dot{q}_{n+1} \\ \ddot{q}_{n+1} \end{bmatrix} = \begin{bmatrix} 1 & h & h^2(0.5 - \beta) \\ 0 & 1 & h(1 - \gamma) \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} q_n \\ \dot{q}_n \\ \ddot{q}_n \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \Phi \end{bmatrix}$$

ainsi que

$$B \begin{bmatrix} q_{n+1} \\ \dot{q}_{n+1} \\ \ddot{q}_{n+1} \end{bmatrix} = C \begin{bmatrix} q_n \\ \dot{q}_n \\ \ddot{q}_n \end{bmatrix} + D$$

Donc on peut calculer le matrice d'amplification A et matrice D'

$$A = B^{-1}C, D' = B^{-1}D$$

Pour chercher la matrice d'amplification, on utilise `sympy` pour trouver cette matrice avec sympole h et $h = \Delta t$.

code python:

```
from IPython.display import display
from sympy import Symbol, Matrix, sqrt
import sympy
sympy.init_printing(use_latex='mathjax')

# m = Symbol("m")
# a = Symbol("a")
# g = Symbol("g")
# F0 = Symbol("F_0")
# h = Symbol("h")
# gamma = Symbol("\gamma")
# beta = Symbol("\\beta")
m = 2
a = 0.5
g = 9.81
F0 = 20
beta = 0
gamma = 0.5
h = Symbol("h")

M = m*(a**2)*Matrix([[2, 1],[1,1]])
K = m*g*a*Matrix([[2,0],[0,1]])
Phi = F0*a*Matrix([[1], [1/sqrt(2)]])

I = sympy.eye(2,2)
# I = sympy.ones(2,2)
b11 = I
b12 = 0*I
b13 = -(h**2)*beta*I
b_row1 = Matrix(b11.row_join(b12).row_join(b13))

b21 = 0*I
b22 = I
b23 = -h*gamma*I
b_row2 = Matrix(b21.row_join(b22).row_join(b23))

b31 = K
b32 = 0*I
b33 = M
b_row3 = Matrix(b31.row_join(b32).row_join(b33))

B = Matrix(b_row1.col_join(b_row2).col_join(b_row3))

c11 = I
c12 = h*I
c13 = h**2*(0.5-beta)*I
c_row1 = Matrix(c11.row_join(c12).row_join(c13))

c21 = 0*I
c22 = I
c23 = h*(1-gamma)*I
c_row2 = Matrix(c21.row_join(c22).row_join(c23))

c_row3 = sympy.zeros(2,6)

C = Matrix(c_row1.col_join(c_row2).col_join(c_row3))

A = B.inv()*C
A = sympy.simplify(A)
display(A)
```

code matlab

```
m = 2;
a = 0.5;
g = 9.81;
F0 = 20;
beta = 0;
gamma = 0.5;
syms h;

M = m*a^2.*[2, 1; 1, 1];
K = m*g*a.*[2, 0; 0, 1];
Phi = F0*a.*[1; 1/sqrt(2)];

I = eye(2);
# I = ones(2,2);
b11 = I;
b12 = 0.*I;
b13 = -(h^2).*beta*I;
b_row1 = [b11, b12, b13];

b21 = 0.*I;
b22 = I;
b23 = -h*gamma.*I;
b_row2 = [b21, b22, b23];

b31 = K;
b32 = 0.*I;
b33 = M;
b_row3 = [b31, b32, b33];

B = [b_row1; b_row2; b_row3];

c11 = I;
c12 = h.*I;
c13 = h^2*(0.5-beta).*I;
c_row1 = [c11, c12, c13];

c21 = 0.*I;
c22 = I;
c23 = h*(1-gamma).*I;
c_row2 = [c21, c22, c23];

c_row3 = zeros(2,6);

C = [c_row1; c_row2; c_row3];

A = inv(B)*C;
A = simplify(A);
```

Qunand $\beta = 0, \gamma = 0.5$, on a:

$$A = \begin{bmatrix} 1.0 & 0 & 1.0h & 0 & 0.5h^2 & 0 \\ 0 & 1.0 & 0 & 1.0h & 0 & 0.5h^2 \\ -19.62h & 9.81h & 1.0 - 19.62h^2 & 9.81h^2 & h\left(0.5 - 9.81h^2\right) & 4.905h^3 \\ 19.62h & -19.62h & 19.62h^2 & 1.0 - 19.62h^2 & 9.81h^3 & h\left(0.5 - 9.81h^2\right) \\ -39.24 & 19.62 & -39.24h & 19.62h & -19.62h^2 & 9.81h^2 \\ 39.24 & -39.24 & 39.24h & -39.24h & 19.62h^2 & -19.62h^2 \end{bmatrix}$$

1.2) Pour déterminer le temps critique, on calcule les valeur propres de la matrice d'amplification, On test Δt de 0s à 1s. On déssine la module de la plus grande valeurs propre.

```python
import numpy as np
import matplotlib.pyplot as plt


def compute_AD(h, beta, gamma, M, K, Phi):
    I = np.eye(2, 2)
    b11 = I
    b12 = 0 * I
    b13 = -h * h * beta * I
    b_row1 = np.hstack((b11, b12, b13))

    b21 = 0 * I
    b22 = 1 * I
    b23 = -h * gamma * I
    b_row2 = np.hstack((b21, b22, b23))

    b31 = K
    b32 = 0 * I
    b33 = M
    b_row3 = np.hstack((b31, b32, b33))

    B = np.vstack((b_row1, b_row2, b_row3))

    c11 = I
    c12 = h * I
    c13 = h * h * (0.5 - beta) * I
    c_row1 = np.hstack((c11, c12, c13))

    c21 = 0 * I
    c22 = 1 * I
    c23 = h * (1 - gamma) * I
    c_row2 = np.hstack((c21, c22, c23))

    c31 = 0 * I
    c32 = 0 * I
    c33 = 0 * I
    c_row3 = np.hstack((c31, c32, c33))

    C = np.vstack((c_row1, c_row2, c_row3))

    B_inv = np.linalg.inv(B)

    D = np.vstack((np.zeros_like(Phi), np.zeros_like(Phi), Phi))

    A = B_inv.dot(C)
    D_prime = B_inv.dot(D)

    return A, D_prime

if __name__ == '__main__':
    # initial params
    m = 2.
    a = 0.5
    g = 9.81
    F0 = 20.
    w = 2 * np.pi
    theta1_0 = 0.
    theta2_0 = 0.
    theta1_0_dot = -1.31519275
    theta2_0_dot = -1.85996342
    theta1_0_ddot = 0
    theta2_0_ddot = 0

    # default settings
    h = 0.02
    h_list = np.arange(0, 1, 0.01)

    T0 = 8
```

```
beta = 0.
gamma = 0.5

M = m * a * a * np.array([[2, 1], [1, 2]])
K = m * g * a * np.array([[2, 0], [0, 1]])
Phi = F0 * np.array([a, a / np.sqrt(2)]).reshape(-1, 1)

v_list = []
for h in h_list:
    A, D_prime = compute_AD(h, beta, gamma, M, K, Phi)
    valeurs_propres, v = np.linalg.eig(A)
    modules = np.abs(valeurs_propres)
    max_v = np.max(modules)
    if abs(max_v - 1) > 0.01 and flag == True:
      print(h)
      flag = False
    v_list.append(max_v)
plt.plot(h_list, v_list)
plt.title("valeurs propres")
plt.xlabel("$\Delta t$")
plt.ylabel("max_v")
plt.show()
```

code matlab

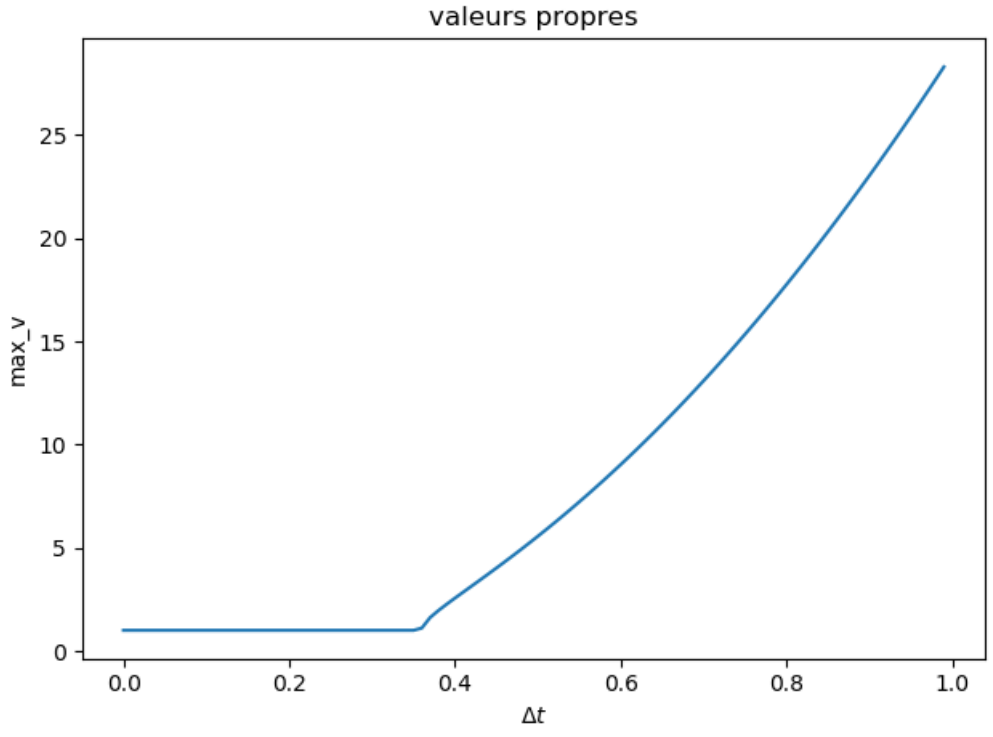
compute_AD.m

```
function [A, D_prime] = compute_AD(h, beta, gamma, M, K, Phi)

I = eye(2, 2);
b11 = I;
b12 = 0 .* I;
b13 = -h * h * beta .* I;
b_row1 = [b11, b12, b13];

b21 = 0 .* I;
b22 = 1 .* I;
b23 = -h * gamma .* I;
b_row2 = [b21, b22, b23];

b31 = K;
b32 = 0 .* I;
b33 = M;
b_row3 = [b31, b32, b33];

B = [b_row1; b_row2; b_row3];

c11 = I;
c12 = h .* I;
c13 = h * h * (0.5 - beta) .* I;
c_row1 = [c11, c12, c13];

c21 = 0 .* I;
c22 = 1 .* I;
c23 = h .* (1 - gamma) * I;
c_row2 = [c21, c22, c23];

c31 = 0 .* I;
c32 = 0 .* I;
c33 = 0 .* I;
c_row3 = [c31, c32, c33];

C = [c_row1; c_row2, c_row3];

B_inv = inv(B);

D = [zeros(size(Phi)); zeros(size(Phi)); Phi];

A = B_inv*C;
D_prime = B_inv*D;

end
```

main.m

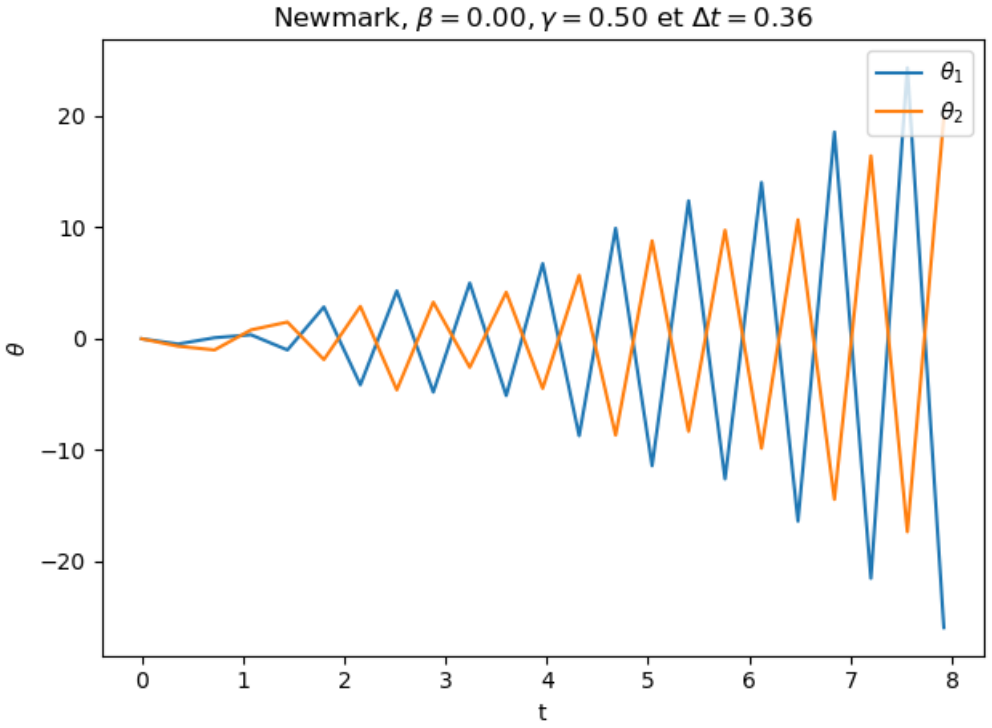
```
% initial params
m = 2.0;
a = 0.5;
g = 9.81;
F0 = 20;
w = 2 * pi;
theta1_0 = 0;
theta2_0 = 0;
theta1_0_dot = -1.31519275;
theta2_0_dot = -1.85996342;
theta1_0_ddot = 0;
theta2_0_ddot = 0;

% default settings
h = 0.02;
h_list = 0:0.01:1;

T0 = 8;
beta = 0;
gamma = 0.5;

M = m * a * a .* [2, 1; 1, 2];
K = m * g * a .* [2, 0; 0, 1];
Phi = F0 .* [a; a / sqrt(2)];

v_list = [];
for h = h_list
    [A, D_prime] = compute_AD(h, beta, gamma, M, K, Phi);
    [valeurs_propres, v] = eig(A);
    modules = abs(valeurs_propres);
    max_v = max(modules);
    v_list = [v_list, max_v];
end
plot(h_list, v_list)
title('valeurs propres')
xlabel('delta_t')
ylabel('max_v'")
```

On a la figure:



Et on trouve le temps de pas critique est 0.36s

En utilisant le code après, on peut le vérifier, quand $\Delta t = 0.36$, le courbe n'est pas stable:



1.3) Comme on a

$$M\ddot{q} + Kq = \Phi$$

Donc on peut obtenir:

$$\ddot{q}_0 = -M^{-1}Kq_0 + M^{-1}\Phi$$

$$\ddot{q}_0 = \begin{bmatrix} 39.24 & -19.62 \\ -39.24 & 39.24 \end{bmatrix} q_0 + \begin{bmatrix} 20.0 - 10.0\sqrt{2} \\ -20.0 + 20.0\sqrt{2} \end{bmatrix} \sin wt$$

Quand t = 0, on a $\ddot{q}_0 = 0$.

1.4) On a déjà la relation dans **1.3**. On a

$$v_{n+1} = Av_n + D'$$

où

$$v = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \dot{\theta}_1 \\ \dot{\theta}_2 \\ \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix}$$

1.5) Programmation :

```python
import numpy as np
import matplotlib.pyplot as plt


def compute_AD(h, beta, gamma, M, K, Phi):
    I = np.eye(2, 2)
    b11 = I
    b12 = 0 * I
    b13 = -h * h * beta * I
    b_row1 = np.hstack((b11, b12, b13))

    b21 = 0 * I
    b22 = 1 * I
    b23 = -h * gamma * I
    b_row2 = np.hstack((b21, b22, b23))

    b31 = K
    b32 = 0 * I
    b33 = M
    b_row3 = np.hstack((b31, b32, b33))

    B = np.vstack((b_row1, b_row2, b_row3))

    c11 = I
    c12 = h * I
    c13 = h * h * (0.5 - beta) * I
    c_row1 = np.hstack((c11, c12, c13))

    c21 = 0 * I
    c22 = 1 * I
    c23 = h * (1 - gamma) * I
    c_row2 = np.hstack((c21, c22, c23))

    c31 = 0 * I
    c32 = 0 * I
    c33 = 0 * I
    c_row3 = np.hstack((c31, c32, c33))

    C = np.vstack((c_row1, c_row2, c_row3))

    B_inv = np.linalg.inv(B)

    D = np.vstack((np.zeros_like(Phi), np.zeros_like(Phi), Phi))

    A = B_inv.dot(C)
    D_prime = B_inv.dot(D)

    return A, D_prime


def newmark(w, delta_t, T, etat_init, A, D_prime):
    t_list = []
    theta1_list = []
    theta2_list = []

    i = 0
    etat_i = etat_init

    while (i * delta_t < T):
        t = i * delta_t
        t_list.append(t)
        theta1_list.append(etat_i[0])
        theta2_list.append(etat_i[1])
        i = i + 1
        # etat_i = A.dot(etat_i)
        etat_i = A.dot(etat_i) + D_prime * np.sin(w * t)

    return t_list, theta1_list, theta2_list
```

```python
if __name__ == '__main__':
    # initial params
    m = 2.
    a = 0.5
    g = 9.81
    F0 = 20.
    w = 2 * np.pi
    theta1_0 = 0.
    theta2_0 = 0.
    theta1_0_dot = -1.31519275
    theta2_0_dot = -1.85996342
    theta1_0_ddot = 0
    theta2_0_ddot = 0

    # default settings
    h = 0.02

    T0 = 8
    beta = 0
    gamma = 0.5

    M = m * a * a * np.array([[2, 1], [1, 2]])
    K = m * g * a * np.array([[2, 0], [0, 1]])
    Phi = F0 * np.array([a, a / np.sqrt(2)]).reshape(-1, 1)

    etat_init = np.array([theta1_0, theta2_0, theta1_0_dot, theta2_0_dot, theta1_0_ddot, theta2_0_ddot]).reshape(-1, 1)
    A, D_prime = compute_AD(h, beta, gamma, M, K, Phi)
    t_list, theta1_list, theta2_list = newmark(w, h, T0, etat_init, A, D_prime)

    plt.plot(t_list, theta1_list, label="$\\theta_1$")
    plt.plot(t_list, theta2_list, label="$\\theta_2$")
    plt.xlabel("t")
    plt.ylabel("$\\theta$")
    plt.legend(loc="upper right")
    plt.title("Newmark, $\\beta = %.2f, \gamma = %.2f$ et $\Delta t = %.2f$"%(beta, gamma, h))
    plt.show()
```

code matlab

compute_AD.m

```
function [A, D_prime] = compute_AD(h, beta, gamma, M, K, Phi)

I = eye(2, 2);
b11 = I;
b12 = 0 .* I;
b13 = -h * h * beta .* I;
b_row1 = [b11, b12, b13];

b21 = 0 .* I;
b22 = 1 .* I;
b23 = -h * gamma .* I;
b_row2 = [b21, b22, b23];

b31 = K;
b32 = 0 .* I;
b33 = M;
b_row3 = [b31, b32, b33];

B = [b_row1; b_row2; b_row3];

c11 = I;
c12 = h .* I;
c13 = h * h * (0.5 - beta) .* I;
c_row1 = [c11, c12, c13];

c21 = 0 .* I;
c22 = 1 .* I;
c23 = h .* (1 - gamma) * I;
c_row2 = [c21, c22, c23];

c31 = 0 .* I;
c32 = 0 .* I;
c33 = 0 .* I;
c_row3 = [c31, c32, c33];

C = [c_row1; c_row2, c_row3];

B_inv = inv(B);

D = [zeros(size(Phi)); zeros(size(Phi)); Phi];

A = B_inv*C;
D_prime = B_inv*D;

end
```

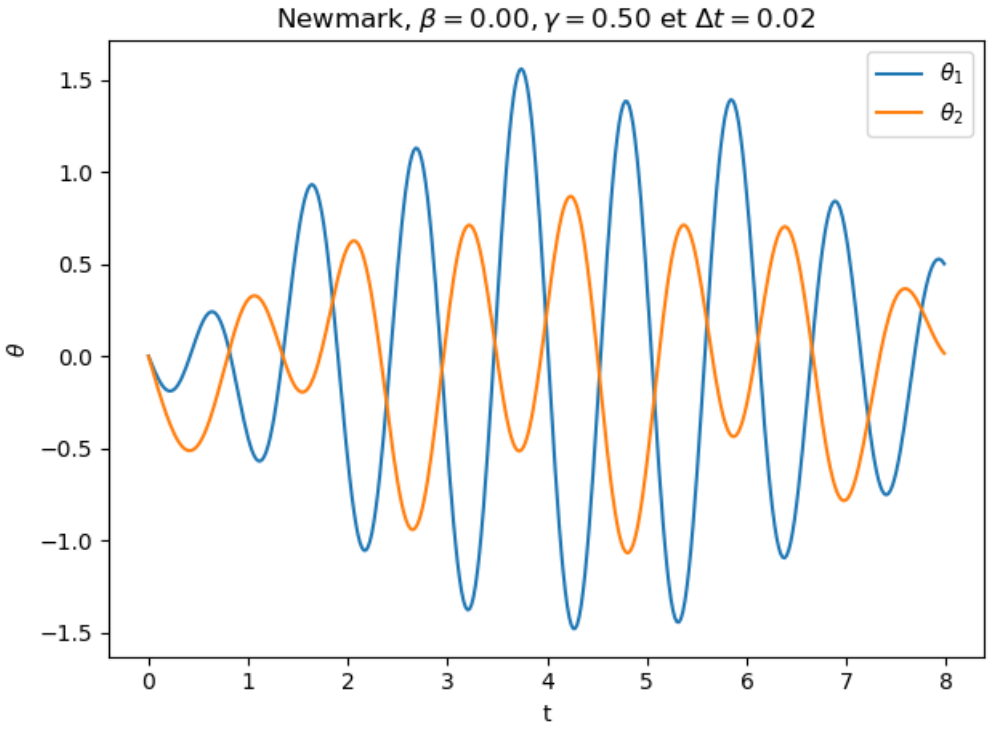
newmark.m

```
function  [t_list, theta1_list, theta2_list] = newmark(w, delta_t, T, etat_init, A, D_prime)
t_list = [];
theta1_list = [];
theta2_list = [];

i = 0;
etat_i = etat_init;

while (i * delta_t < T)
    t = i * delta_t;
    t_list = [t_list, t];
    theta1_list = [theta1_list, etat_i(1)];
    theta2_list = [theta2_list, etat_i(2)];
    i = i + 1;
    % etat_i = A*etat_i;
    etat_i = A*etat_i + D_prime * sin(w*t);
end

end
```

main.m

```
% initial params
m = 2.0;
a = 0.5;
g = 9.81;
F0 = 20;
w = 2 * pi;
theta1_0 = 0;
theta2_0 = 0;
theta1_0_dot = -1.31519275;
theta2_0_dot = -1.85996342;
theta1_0_ddot = 0;
theta2_0_ddot = 0;

% default settings
h = 0.02;

T0 = 8;
beta = 0;
gamma = 0.5;

M = m * a * a .* [2, 1; 1, 2];
K = m * g * a .* [2, 0; 0, 1];
Phi = F0 .* [a; a / sqrt(2)];

etat_init = [theta1_0; theta2_0; theta1_0_dot; theta2_0_dot; theta1_0_ddot; theta2_0_ddot];
[A, D_prime] = compute_AD(h, beta, gamma, M, K, Phi);
[t_list, theta1_list, theta2_list] = newmark(w, h, T0, etat_init, A, D_prime);

plot(t_list, theta1_list, 'DisplayName', 'theta_1')
hold on
plot(t_list, theta2_list, 'DisplayName', 'theta_2')
xlabel('t')
ylabel('theta')
title(sprintf('Newmark, beta = %0.2f, gamma=%0.2f, delta_t = %0.2f', beta, gamma, h))
hold off
legend
```

1.6) D’après le code, on a la figure:



Si $\Delta t = 0.02s$
Quand $t = 0s$, on a:

$$q(t) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \dot{q}(t) = \begin{bmatrix} -1.32 \\ -1.86 \end{bmatrix}, \ddot{q}(t) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Quand $t = \Delta t$, on a:

$$q(t) = \begin{bmatrix} -0.0263 \\ -0.0372 \end{bmatrix}, \dot{q}(t) = \begin{bmatrix} -1.31 \\ -1.86 \end{bmatrix}, \ddot{q}(t) = \begin{bmatrix} 0.445 \\ 0.143 \end{bmatrix}$$

Quand $t = 2\Delta t$, on a:

$$q(t) = \begin{bmatrix} -0.0524 \\ -0.0743 \end{bmatrix}, \dot{q}(t) = \begin{bmatrix} -1.29 \\ -1.85 \end{bmatrix}, \ddot{q}(t) = \begin{bmatrix} 1.97 \\ 0.633 \end{bmatrix}$$

Quand $t = 0.5s$, on a:

$$q(t) = \begin{bmatrix} 0.130 \\ -0.481 \end{bmatrix}, \dot{q}(t) = \begin{bmatrix} 1.43 \\ 0.739 \end{bmatrix}, \ddot{q}(t) = \begin{bmatrix} -5.46 \\ 8.33 \end{bmatrix}$$

2

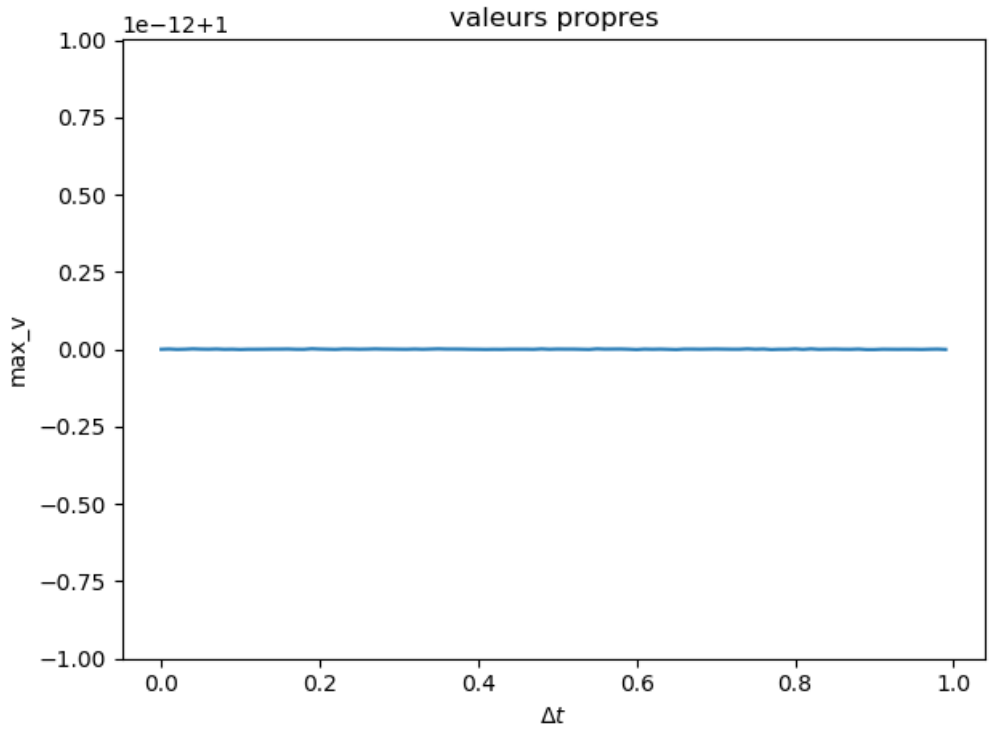
2.1)

$$A = \begin{bmatrix} \frac{1.0\left(24.059025h^2+2.4525\right)}{118.009517625h^4+48.11805h^2+2.4525} & \cdots & \frac{3.007378125h^4}{118.009517625h^4+48.11805h^2+2.4525} \\ \vdots & \ddots & \vdots \\ \frac{9.81}{12.0295125h^4+4.905h^2+0.25} & \cdots & -\frac{h^2\left(12.0295125h^2+2.4525\right)}{12.0295125h^4+4.905h^2+0.25} \end{bmatrix}$$

2.2) Il n'est pas possible de trouver des valeurs propres sans donner une valeur de Δt. Donc on va évaluer de pas à pas numériquement.

En utilisant le code de question **2.2** on a la figure ci-dessous:

On a une ligne horizontale.



Donc la plus grande valeur propre (il s'agit son module) est 1. Donc le courbe est toujours stable.

2.3)
Comme on a

$$M\ddot{q} + Kq = \Phi$$

Donc on peut obtenir:

$$\ddot{q}_0 = -M^{-1}Kq_0 + M^{-1}\Phi$$

$$\ddot{q}_0 = \begin{bmatrix} 39.24 & -19.62 \\ -39.24 & 39.24 \end{bmatrix} q_0 + \begin{bmatrix} 20.0 - 10.0\sqrt{2} \\ -20.0 + 20.0\sqrt{2} \end{bmatrix} \sin wt$$

Quand t = 0, on a $\ddot{q}_0 = 0$.

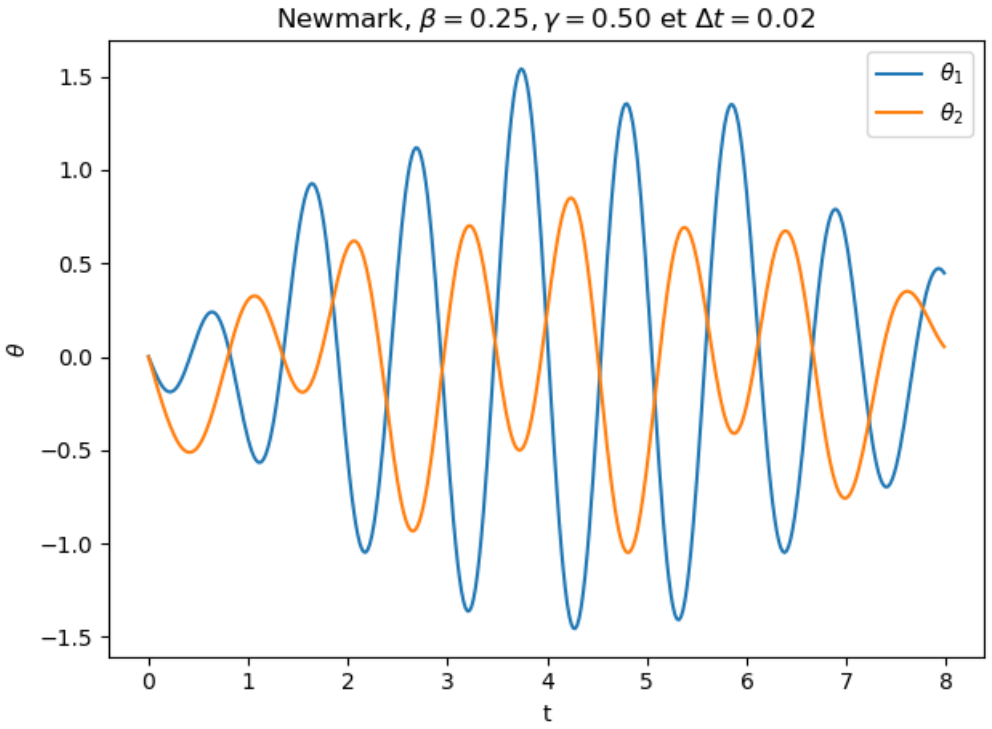
2.4) On a déjà la relation dans **1.3**. On a

$$v_{n+1} = Av_n + D'$$

où

$$v = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \dot{\theta}_1 \\ \dot{\theta}_2 \\ \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix}$$

2.5) On utilise encore le code de question **1.5**, il faut seulement change β de 0 à 0.25.
2.6) Si on choisit $\Delta t = 0.02s$, on a la figure ci-dessous



Si $\Delta t = 0.02s$
Quand $t = 0s$, on a:

$$q(t) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \dot{q}(t) = \begin{bmatrix} -1.32 \\ -1.86 \end{bmatrix}, \ddot{q}(t) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Quand $t = \Delta t$, on a:

$$q(t) = \begin{bmatrix} -0.0263 \\ -0.0372 \end{bmatrix}, \dot{q}(t) = \begin{bmatrix} -1.31 \\ -1.86 \end{bmatrix}, \ddot{q}(t) = \begin{bmatrix} 0.444 \\ 0.143 \end{bmatrix}$$

Quand $t = 2\Delta t$, on a:

$$q(t) = \begin{bmatrix} -0.0522 \\ -0.0743 \end{bmatrix}, \dot{q}(t) = \begin{bmatrix} -1.29 \\ -1.85 \end{bmatrix}, \ddot{q}(t) = \begin{bmatrix} 1.96 \\ 0.634 \end{bmatrix}$$

Quand $t = 0.5s$, on a:

$$q(t) = \begin{bmatrix} 0.128 \\ -0.480 \end{bmatrix}, \dot{q}(t) = \begin{bmatrix} 1.43 \\ 0.736 \end{bmatrix}, \ddot{q}(t) = \begin{bmatrix} -5.42 \\ 8.30 \end{bmatrix}$$