Séance 4 Dynamique des structures, Systems

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

Oscillateur non linéaire à un degré de liberté

L'équation de mouvement:

$$\ddot{q} - \epsilon w_0 \left[1 - \left(\frac{q}{y_0}\right)^2\right] \dot{q} + w_0^2 q = 0 \tag{1}$$

1 Runge Kutta

1.1.a) Pour utiliser le schéma Runge Kutta, il faut transformer l'équation (1) aux premier ordre. On introduit $z = \dot{q}$, Donc on a

$$\dot{z} = \epsilon w_0 [1 - (\frac{q}{y_0})^2] z - w_0^2 q$$

1.1.b) Vecteur d'état est toujours $[q, \dot{q}]^T$

1.2) Pour simplifier les équation, on note seulement dans cette question que q_0, z_0 sont des état entrée et q_1, z_1 sont des états sorties. q_0, z_0 est en effet $q_j, \dot{q}_j$ et q_1, z_1 est en effet $q_{j+1}, \dot{q}_{j+1}$. Donc D'arès Runge Kutta, on a:

$$\begin{cases} k_1 = z_0 = \dot{q}_0 \\ j_1 = \epsilon w_0 [1 - (\frac{q_0}{y_0})^2] z_0 - w_0^2 q_0 \\ k_2 = z_0 + \frac{j_1}{2} \Delta t \\ j_2 = \epsilon w_0 [1 - (\frac{q_0 + \frac{k_1}{2} \Delta t}{y_0})^2] (z_0 + \frac{j_1}{2} \Delta t) - w_0^2 (q_0 + \frac{k_1}{2} \Delta t) \\ k_3 = z_0 + \frac{j_2}{2} \Delta t \\ j_3 = \epsilon w_0 [1 - (\frac{q_0 + \frac{k_2}{2} \Delta t}{y_0})^2] (z_0 + \frac{j_2}{2} \Delta t) - w_0^2 (q_0 + \frac{k_2}{2} \Delta t) \\ k_4 = z_0 + j_3 \Delta t \\ j_4 = \epsilon w_0 [1 - (\frac{q_0 + k_3 \Delta t}{y_0})^2] (z_0 + j_3 \Delta t) - w_0^2 (q_0 + k_3 \Delta t) \end{cases}$$

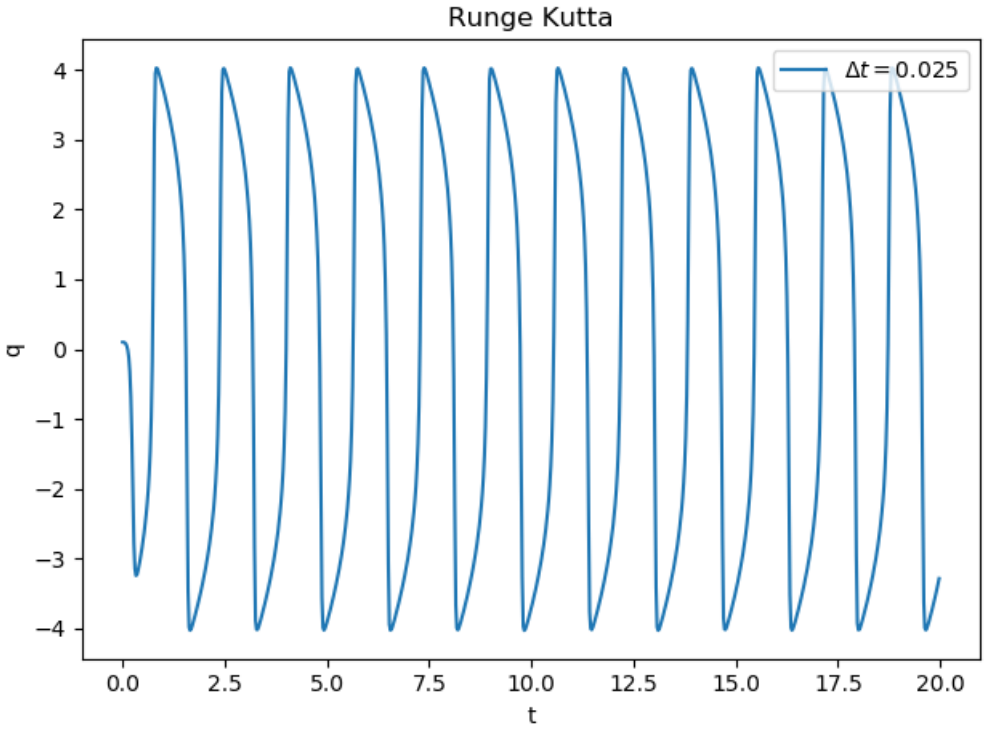
Puis on a :

$$\begin{cases} q_1 = q_0 + \frac{k_1 + 2k_2 + 2k_3 + k_4}{6} \Delta t \\ z_1 = z_0 + \frac{j_1 + 2j_2 + 2j_3 + j_4}{6} \Delta t \end{cases} \tag{2}$$

1.3) D'après la question précédente, on peut le programmer.
code python

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


def one_step(etat, epsilon, w0, y0, dt):
    # unzip
    q0 = etat[0]
    z0 = etat[1]

    k1 = z0
    j1 = epsilon * w0 * (1 - (q0 / y0) ** 2) * z0 - w0 ** 2 * q0
    k2 = z0 + j1 * dt / 2.
    j2 = epsilon * w0 * (1 - ((q0 + k1 * dt / 2.) / y0) ** 2) * (z0 + j1 * dt / 2.) - w0 ** 2 * (q0 + k1 * dt / 2.)
    k3 = z0 + j2 * dt / 2.
    j3 = epsilon * w0 * (1 - ((q0 + k2 * dt / 2.) / y0) ** 2) * (z0 + j2 * dt / 2.) - w0 ** 2 * (q0 + k2 * dt / 2.)
    k4 = z0 + j3 * dt
    j4 = epsilon * w0 * (1 - ((q0 + k3 * dt) / y0) ** 2) * (z0 + j3 * dt) - w0 ** 2 * (q0 + k3 * dt)

    q1 = q0 + (k1 + 2 * k2 + 2 * k3 + k4) * dt / 6.
    z1 = z0 + (j1 + 2 * j2 + 2 * j3 + j4) * dt / 6.

    etat = [q1, z1] # zip
    return etat


def runge_kutta(etat_init, epsilon, w0, y0, T, dt):
    t_list = []
    q_list = []

    i = 0
    etat_i = etat_init
    while (i * dt < T):
        t = i*dt
        t_list.append(t)
        q_list.append(etat_i[0])

        etat_i = one_step(etat_i, epsilon, w0, y0, dt)
        i = i + 1

    return t_list, q_list


if __name__ == '__main__':
    # init params
    q0 = 0.1
    q0_dot = 0
    w0 = 2 * np.pi
    y0 = 2

    # default settings
    T0 = 20
    h = 0.025  # le pas de temps
    epsilon = 4

    etat_init = [q0, q0_dot]
    t_list, q_list = runge_kutta(etat_init, epsilon, w0, y0, T0, h)

    plt.plot(t_list, q_list, label="$\Delta t = %.3f$"%h)
    plt.title("Runge Kutta")
    plt.xlabel("t")
    plt.ylabel("q")
    plt.legend(loc="upper right")
    plt.show()
```

Et on peut déssiner la figure:



1.4)
Quand $t = 0$, $q(t) = 0.1$
Quand $t = \Delta t$, $q(t) = 0.09847070681354071$
Quand $t = 2\Delta t$, $q(t) = 0.09224029284401843$
Quand $t = 5\Delta t$, $q(t) = -0.011491881177363904$

2 Newmark Explicite

2.1) On a l'équation de mouvement

$$\ddot{q} - \epsilon w_0 \left[1 - \left(\frac{q}{y_0}\right)^2\right] \dot{q} + w_0^2 q = 0 \tag{1}$$

Pour Newmark Explicite on a $\gamma = 0.5. \beta = 0$, donc

$$q_{j+1} = q_j + \Delta t \dot{q}_j + \frac{\Delta t^2}{2} \ddot{q}_j \tag{3}$$

$$\dot{q}_{j+1} = \dot{q}_j + \frac{\Delta t}{2}(\ddot{q}_j + \ddot{q}_{j+1}) \tag{4}$$

D'après l'équation (1), on a:

$$\ddot{q}_{j+1} = \epsilon w_0 [1 - (\frac{q_{j+1}}{y_0})^2] \dot{q}_{j+1} - w_0^2 q_{j+1} \tag{5}$$

D'après (4) et (5), on obtien:

$$(1 - \frac{\epsilon w_0 \Delta t}{2}(1 - (\frac{q_{j+1}}{y_0})^2)) \dot{q}_{j+1} = \dot{q}_j + \frac{\Delta t}{2} \ddot{q}_j - \frac{w_0^2 \Delta t}{2} q_{j+1} \tag{6}$$

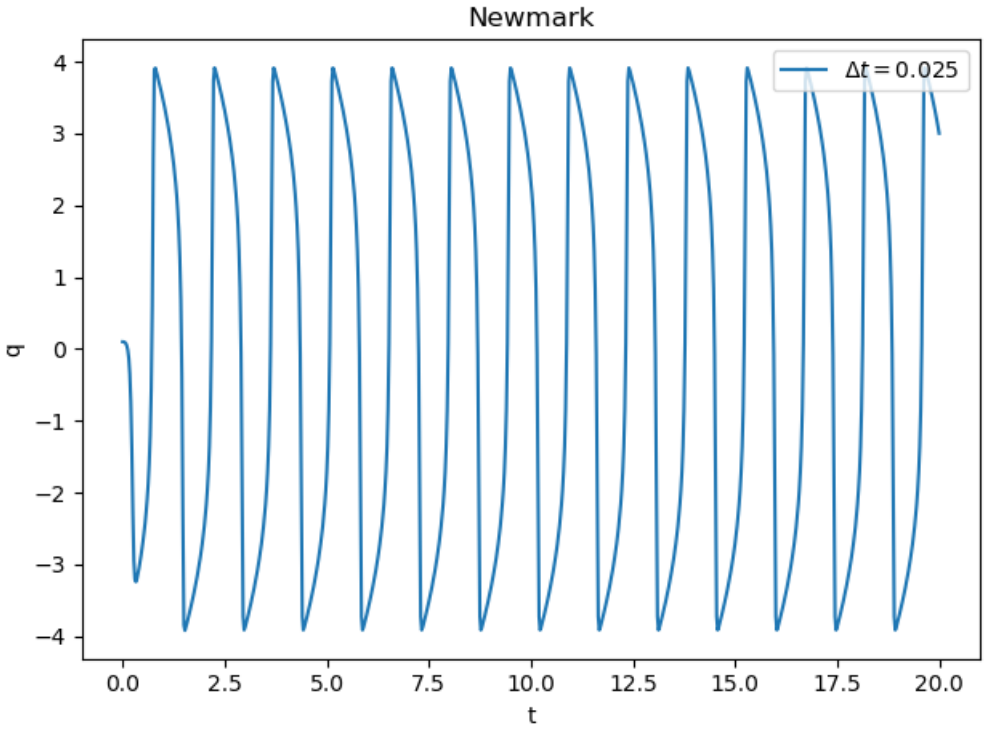
En utilisant l'équation (3), on peut calculer q_{j+1} à partir des valeurs de $q_j, \dot{q}_j, \ddot{q}_j$. Après on a q_{j+1}, on peut calculer $\dot{q}_{j+1}$ avec l'équation (6). Alors on a $\ddot{q}_{j+1}$ avec l'équation (5).

2.2) D'après la question **2.1**, on peut le programmer

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


def one_step(etat, epsilon, w0, y0, dt):
    # unzip
    q0 = etat[0]
    z0 = etat[1]

    k1 = z0
    j1 = epsilon * w0 * (1 - (q0 / y0) ** 2) * z0 - w0 ** 2 * q0
    k2 = z0 + j1 * dt / 2.
    j2 = epsilon * w0 * (1 - ((q0 + k1 * dt / 2.) / y0) ** 2) * (z0 + j1 * dt / 2.) - w0 ** 2 * (q0 + k1 * dt / 2.)
    k3 = z0 + j2 * dt / 2.
    j3 = epsilon * w0 * (1 - ((q0 + k2 * dt / 2.) / y0) ** 2) * (z0 + j2 * dt / 2.) - w0 ** 2 * (q0 + k2 * dt / 2.)
    k4 = z0 + j3 * dt
    j4 = epsilon * w0 * (1 - ((q0 + k3 * dt) / y0) ** 2) * (z0 + j3 * dt) - w0 ** 2 * (q0 + k3 * dt)

    q1 = q0 + (k1 + 2 * k2 + 2 * k3 + k4) * dt / 6.
    z1 = z0 + (j1 + 2 * j2 + 2 * j3 + j4) * dt / 6.

    etat = [q1, z1]  # zip
    return etat


def runge_kutta(etat_init, epsilon, w0, y0, T, dt):
    assert len(etat_init) == 2, "Runge kutta only accpet etat like [q, q_dot]"
    t_list = []
    q_list = []

    i = 0
    etat_i = etat_init
    while (i * dt < T):
        t = i * dt
        t_list.append(t)
        q_list.append(etat_i[0])
        if i in [0, 1, 2, 5]:
            print("Quand $t = {}\Delta t$, $q(t) = {}$".format(i, etat_i[0]))

        etat_i = one_step(etat_i, epsilon, w0, y0, dt)
        i = i + 1

    return t_list, q_list


def compute_q_ddot(epsilon, w0, y0, q, q_dot):
    return epsilon * w0 * (1 - (q / y0) ** 2) * q_dot - w0 ** 2 * q


def newmark(etat_init, epsilon, w0, y0, T, dt, mode="explicite", precision=0.01):
    assert len(etat_init) == 3, "Newmark only accpet etat like [q, q_dot, q_ddot]"
    t_list = []
    q_list = []

    i = 0
    etat_i = etat_init

    while (i * dt <= T):
        t = i * dt
        t_list.append(t)
        q_list.append(etat_i[0])

        if mode == "explicite":
            q_ip1 = etat_i[0] + dt * etat_i[1] + dt ** 2 * etat_i[2] / 2
            numerator = etat_i[1] + dt * etat_i[2] / 2 - w0 ** 2 * dt * q_ip1 / 2
            denominator = 1 - epsilon * w0 * dt / 2 * (1 - (q_ip1 / y0) ** 2)
```

```
            q_dot_ip1 = numerator / denominator
            q_ddot_ip1 = epsilon * w0 * (1 - (q_ip1 / y0) ** 2) * q_dot_ip1 - w0 ** 2 * q_ip1
            etat_i = [q_ip1, q_dot_ip1, q_ddot_ip1]


        elif mode == "implicite":
            raise NotImplementedError
        else:
            raise NotImplementedError
        i = i + 1

    return t_list, q_list


if __name__ == '__main__':
    # init params
    q0 = 0.1
    q0_dot = 0
    w0 = 2 * np.pi
    y0 = 2

    # default settings
    T0 = 20
    h = 0.025  # le pas de temps
    epsilon = 4

    # Runge Kutta
    # etat_init = [q0, q0_dot]
    # t_list, q_list = runge_kutta(etat_init, epsilon, w0, y0, T0, h)

    # Newmark
    q0_ddot = compute_q_ddot(epsilon, w0, y0, q0, q0_dot)
    etat_init = [q0, q0_dot, q0_ddot]

    t_list, q_list = newmark(etat_init, epsilon, w0, y0, T0, h, mode="explicite")

    plt.plot(t_list, q_list, label="$\Delta t = %.3f$" % h)
    plt.title("Newmark")
    plt.xlabel("t")
    plt.ylabel("q")
    plt.legend(loc="upper right")
    plt.show()
```

2.3) Quand on choisit $\epsilon = 4, \Delta t = 0.025$, on peut déssiner la figure:



et on a quelques valeurs numériques:

Quand $t = 0$, $q(t) = 0.1$
Quand $t = \Delta t$, $q(t) = 0.09876629944986384$
Quand $t = 2\Delta t$, $q(t) = 0.09285710346130857$
Quand $t = 5\Delta t$, $q(t) = -0.013400637689010692$

3) On s'intéresse au cas $\epsilon = 0.1$

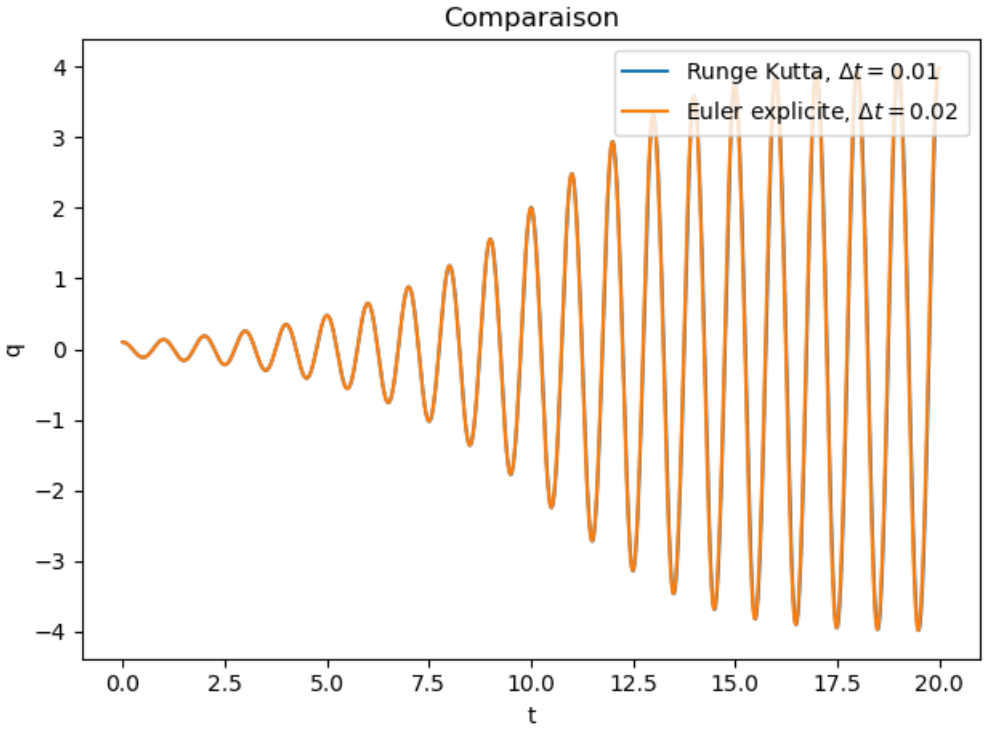
3.1.1)
Pour le schéma Runge Kutta
Quand $t = 0\Delta t$, $q(t) = 0.1$
Quand $t = 1\Delta t$, $q(t) = 0.09980225982071861$
Quand $t = 2\Delta t$, $q(t) = 0.09920816567665768$
Quand $t = 5\Delta t$, $q(t) = 0.09505420471322289$

Pour le schéma Euler explicite
Quand $t = 0\Delta t$, $q(t) = 0.1$
Quand $t = 1\Delta t$, $q(t) = 0.09921043164791285$
Quand $t = 2\Delta t$, $q(t) = 0.0968343535557966$
Quand $t = 5\Delta t$, $q(t) = 0.08048981985591919$

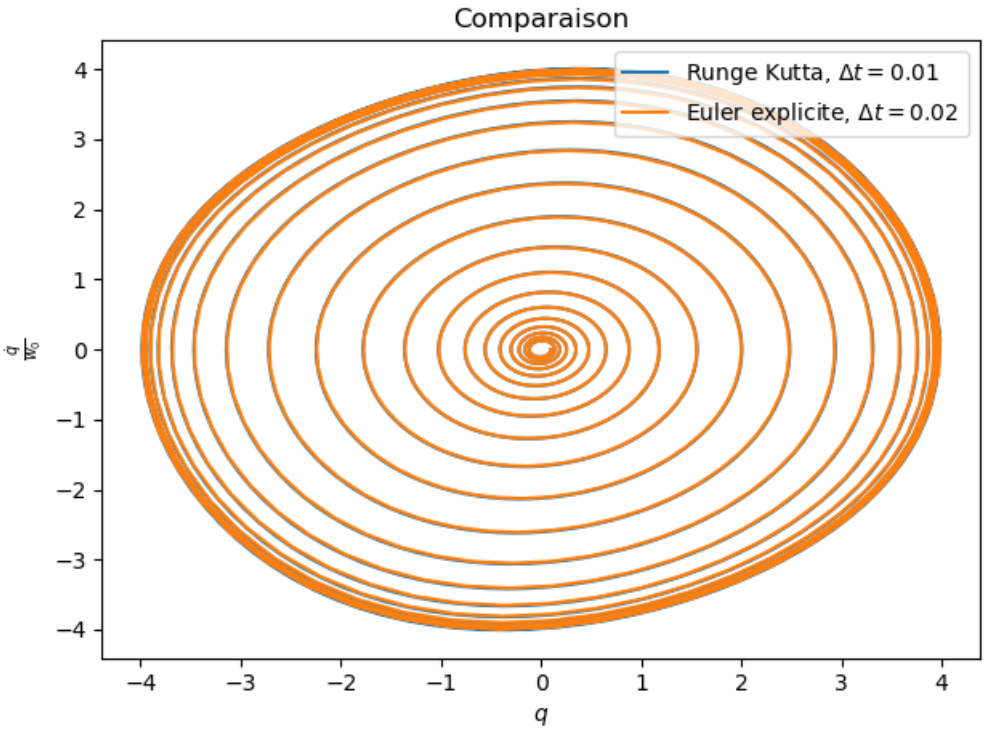
Les valeurs sont très proches.

3.1.2) On observe que les deux courbes se chevauchent parfaitement, seulement dans l'intervalle $[0, 20s]$, on remarque il y a des divergence pour chaque courbe.

D'abord pour (t, q), on a une figure:



Pour $(q, \frac{\dot{q}}{w_0})$ on a une autre figure:



On observe les deux courbes sont en spirale, plus près de l'extérieure, plus la ligne est dense. Donc peut-être le système ne diverge pas, il peut-être stable.

On s'intéresse au cas $\epsilon = 5$

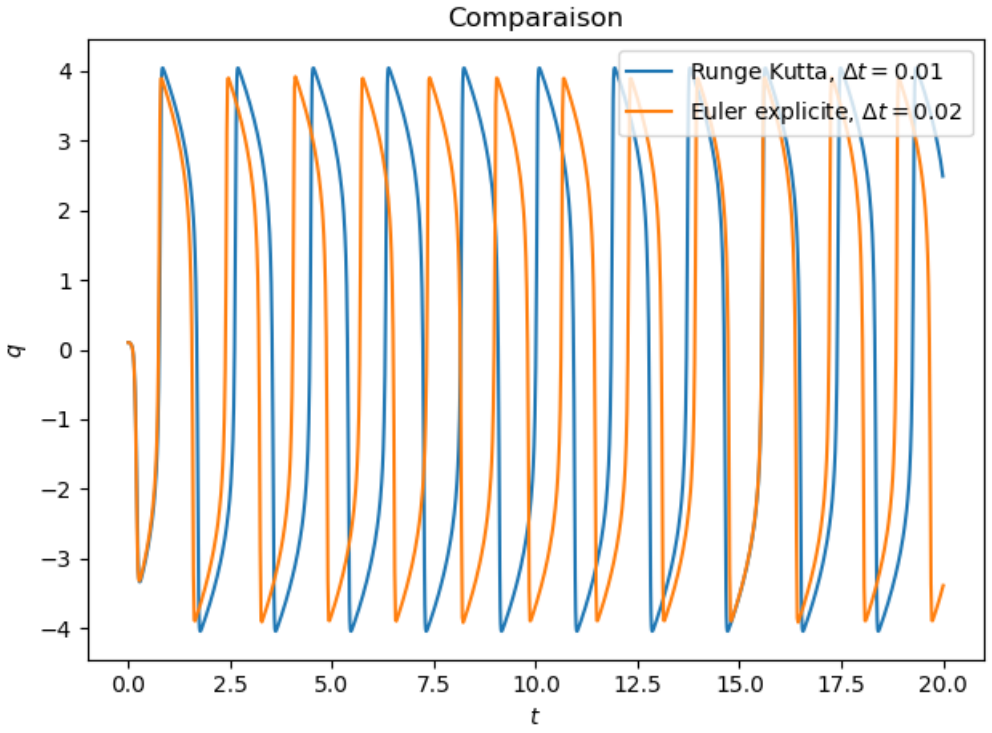
3.2.1) on a des nouvaux solutions:

Pour le schéma Runge Kutta
Quand $t = 0\Delta t$, $q(t) = 0.1$
Quand $t = 1\Delta t$, $q(t) = 0.09978043824084103$
Quand $t = 2\Delta t$, $q(t) = 0.09901762765835839$
Quand $t = 5\Delta t$, $q(t) = 0.09113691943034163$

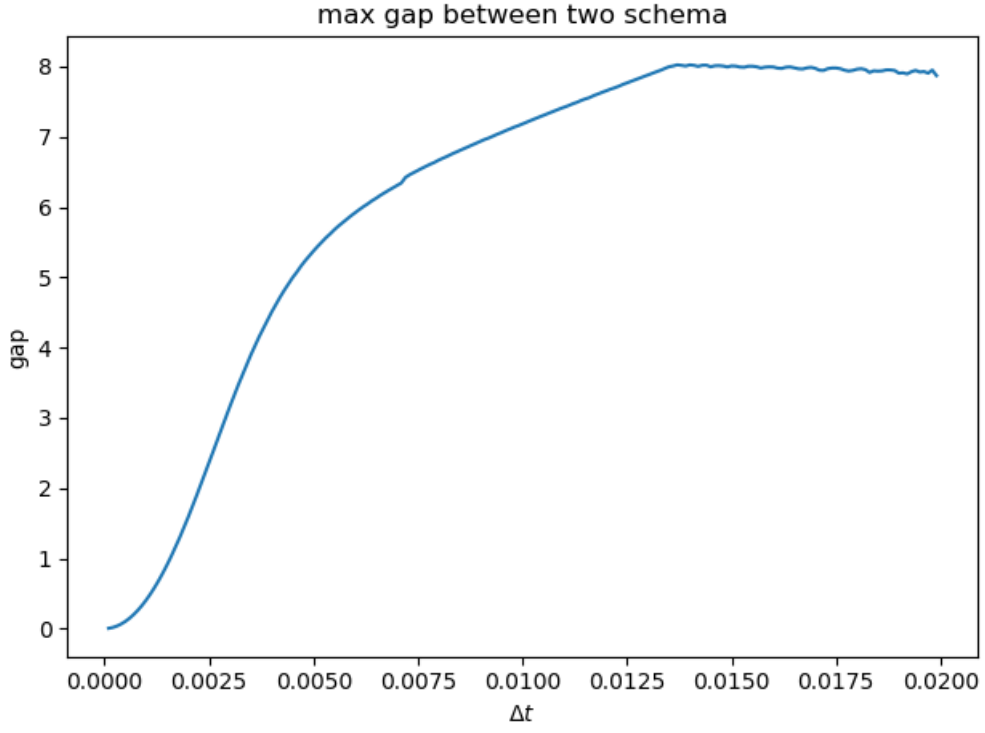
Pour le schéma Euler explicite
Quand $t = 0\Delta t$, $q(t) = 0.1$
Quand $t = 1\Delta t$, $q(t) = 0.09921043164791285$
Quand $t = 2\Delta t$, $q(t) = 0.0954183776385891$
Quand $t = 5\Delta t$, $q(t) = 0.025982627085860993$

Les valeurs sont aussi très proche, sans le dernier terme.

3.2.2) En ce moment, on ne peut pas déterminer ce qui est correcte parce que quand on déssine la figure des deux schéma, on trouve que il n'y a plus la chevauchement entre les deux.



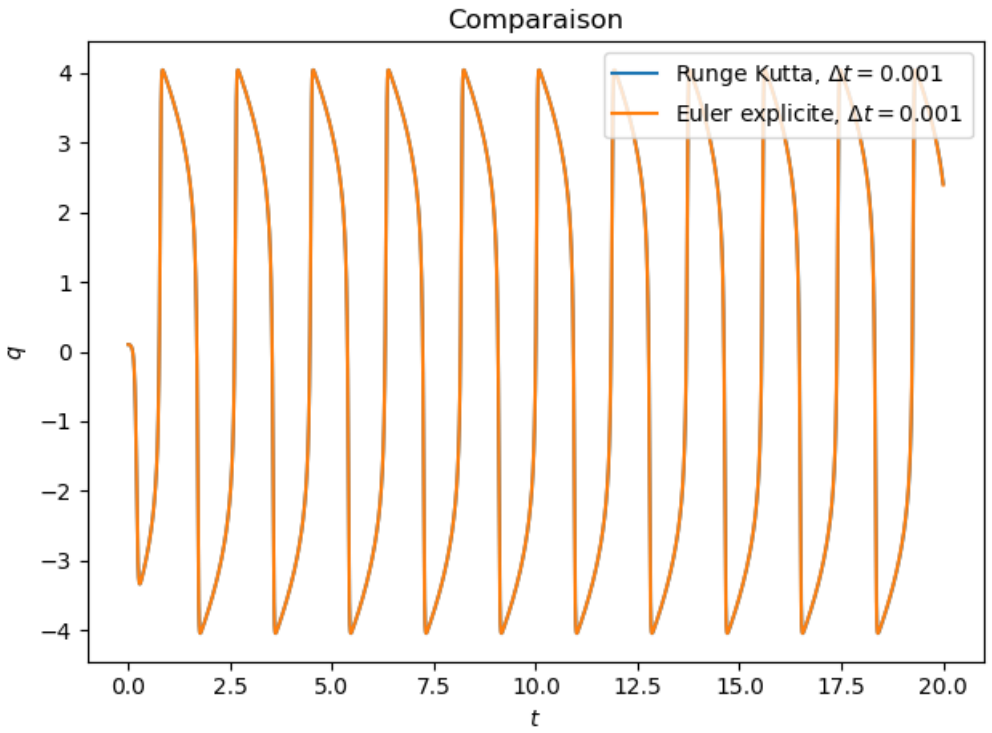
3.2.3) On teste toutes les valeurs entre $[0.001s, 0.02s]$ pour le pas de temps, intervalle est 0.001s, et on déssine le plus grand écart entre deux schéma.



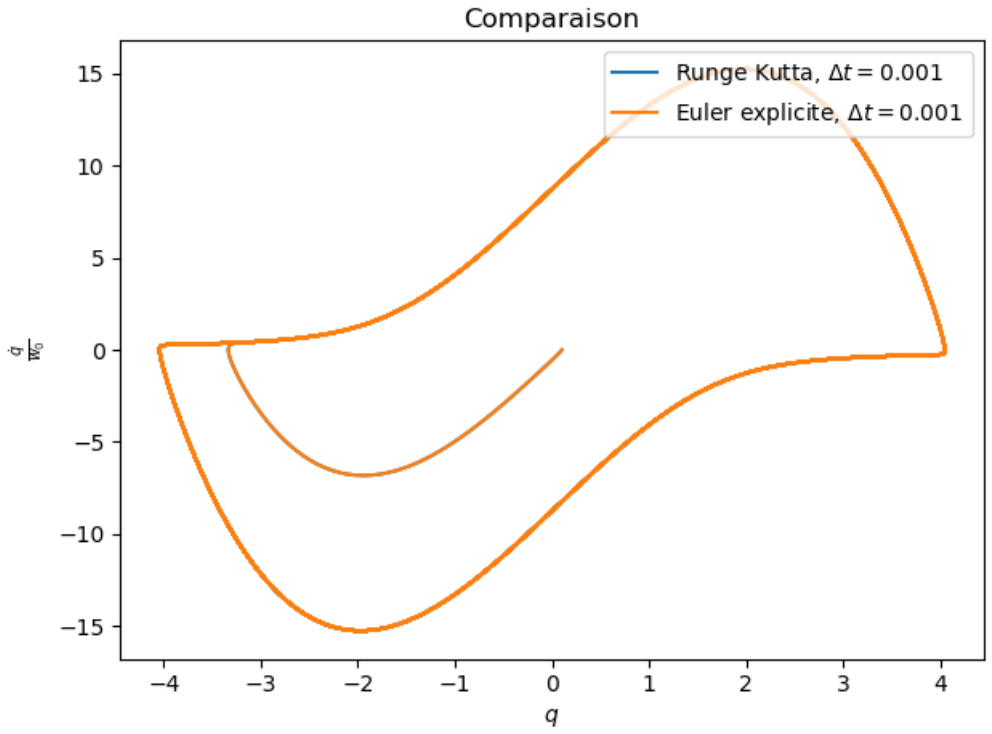
Donc quand le pas de temps tends vers 0, les deux schémas plus proche. Et on sait que les deux schéma va tends vers la solution analytique. Donc on va utiliser le pas de temps qui a le plus petit valeur. C'est à dire $\Delta t = 0.001s$

3.2.4)

D'abord pour (t, q), on a une figure:



Pour $(q, \frac{\dot{q}}{w_0})$ on a une autre figure:



Les deux courbes se chevauchent parfaitement