```
import numpy as np
from numpy import sin, cos
from matplotlib import animation
import pylab

def RungeKutta(f1, f2, f3, f4, a_init, b_init, c_init, d_init, t0, tf, h=0.1):
    A = []
    B = []
    C = []
    D = []
    T = np.arange(t0, tf, h)
    a = a_init
    b = b_init
    c = c_init
    d = d_init
    for t in T:
        A.append(a)
        B.append(b)
        C.append(c)
        D.append(d)
        k1 = h * f1(a, b, c, d, t)
        l1 = h * f2(a, b, c, d, t)
        m1 = h * f3(a, b, c, d, t)
        n1 = h * f4(a, b, c, d, t)
        k2 = h * f1(a + k1 / 2, b + l1 / 2, c + m1 / 2, d + n1 / 2, t + h / 2)
        l2 = h * f2(a + k1 / 2, b + l1 / 2, c + m1 / 2, d + n1 / 2, t + h / 2)
        m2 = h * f3(a + k1 / 2, b + l1 / 2, c + m1 / 2, d + n1 / 2, t + h / 2)
        n2 = h * f4(a + k1 / 2, b + l1 / 2, c + m1 / 2, d + n1 / 2, t + h / 2)
        k3 = h * f1(a + k2 / 2, b + l2 / 2, c + m2 / 2, d + n2 / 2, t + h / 2)
        l3 = h * f2(a + k2 / 2, b + l2 / 2, c + m2 / 2, d + n2 / 2, t + h / 2)
        m3 = h * f3(a + k2 / 2, b + l2 / 2, c + m2 / 2, d + n2 / 2, t + h / 2)
        n3 = h * f4(a + k2 / 2, b + l2 / 2, c + m2 / 2, d + n2 / 2, t + h / 2)
        k4 = h * f1(a + k3, b + l3, c + m3, d + n3, t + h)
        l4 = h * f2(a + k3, b + l3, c + m3, d + n3, t + h)
        m4 = h * f3(a + k3, b + l3, c + m3, d + n3, t + h)
        n4 = h * f4(a + k3, b + l3, c + m3, d + n3, t + h)
        a += (k1 + 2 * k2 + 2 * k3 + k4) / 6
        b += (l1 + 2 * l2 + 2 * l3 + l4) / 6
        c += (m1 + 2 * m2 + 2 * m3 + m4) / 6
        d += (n1 + 2 * n2 + 2 * n3 + n4) / 6
    return T, A, B, C, D

m = 1.0
g = 9.8
l = 0.4

def f1(w1, w2, t1, t2, t):
    return -((w1**2)*sin(2*t1-2*t2)+2*(w2**2)*sin(t1-t2)+(g/l)*(sin(t1-2*t2)+3*sin(t1)))/(3-cos(2*t1-2*t2))

def f2(w1, w2, t1, t2, t):
    return (4*(w1**2)*sin(t1-t2)+(w2**2)*sin(2*t1-2*t2)+2*(g/l)*(sin(2*t1-t2)-sin(t2)))/(3-cos(2*t1-2*t2))

def f3(w1, w2, t1, t2, t):
    return w1

def f4(w1, w2, t1, t2, t):
    return w2

def main():
    h = 0.0018
    T, W1, W2, T1, T2 = RungeKutta(f1, f2, f3, f4, 0,0,np.pi/2, np.pi/2, 0, 100, h)
```

```python
    X1 = [-l*sin(t1) for t1 in T1]
    Y1 = [-l*cos(t1) for t1 in T1]
    X2 = [-l*(sin(T1[i]) + sin(T2[i])) for i in range(len(T1))]
    Y2 = [-l*(cos(T1[i]) + cos(T2[i])) for i in range(len(T1))]
    E = [m*l**2*(W1[i]**2+0.5*W2[i]**2+W1[i]*W2[i]*cos(T1[i]-T2[i]))-m*g*l*(2*cos(T1[i]) + cos(T2[i])) for i in range(len(T))]
    pylab.plot(T, E, 'k')
    pylab.show()
    fig, ax = pylab.subplots()
    ax.set_ylim(-1,1)
    ax.set_xlim(-1,1)
    line, = ax.plot([0, X1[0], X2[0]], [0, Y1[0], Y2[0]], 'k')
    ax.plot(0, 0, 'ro')
    p1, = ax.plot(X1[0], Y1[0], 'go')
    p2, = ax.plot(X2[0], Y2[0], 'bo')
    t1, = ax.plot(X1[0], Y1[0], 'g-', alpha=0.7)
    t2, = ax.plot(X2[0], Y2[0], 'b-', alpha=0.7)
    t1x = []
    t1y = []
    t2x = []
    t2y = []

    def init():
        return line,

    def anim(t):
        line.set_data([0, X1[t*10], X2[t*10]], [0, Y1[t*10], Y2[t*10]])
        p1.set_data(X1[t*10], Y1[t*10])
        p2.set_data(X2[t*10], Y2[t*10])
        t1x.append(X1[t*10])
        t2x.append(X2[t*10])
        t1y.append(Y1[t*10])
        t2y.append(Y2[t*10])
        if len(t1x) >= 100:
            t1x.pop(0)
            t1y.pop(0)
            t2x.pop(0)
            t2y.pop(0)
        t1.set_data(t1x, t1y)
        t2.set_data(t2x, t2y)
        return line,

    ani = animation.FuncAnimation(fig, anim, init_func =init, interval=1000 * h, frames=range(len(T) // 10))
    pylab.show()

if __name__ == "__main__":
    main()
```