```python
import matplotlib.animation as animation
import numpy as np
import pylab

L = 5e-10
hb = 1.054571800e-34
mass = 9.1094e-31
# a = 0
a = 1.60218e-18
PSI = None


def simpson(func, a, b, n=50):
    """Approximates integral using simpson method"""
    h = abs(b - a) / n
    return (h / 3.0) * (
        func(a) + func(b) +
        (4.0 * sum([func(a + (k * h)) for k in range(1, n, 2)])) +
        (2.0 * sum([func(a + (k * h)) for k in range(2, n - 1, 2)])))


def H(n, m, v):
    """Generates n.m element of H matrix"""
    n += 1
    m += 1
    pre_1 = ((hb**2) * (n**2) * (np.pi**2)) / (mass * (L**3))
    pre_2 = 2 / L
    int_1 = simpson(
        lambda x: np.sin(m * np.pi * x / L) * np.sin(n * np.pi * x / L), 0, L)
    int_2 = simpson(
        lambda x: v(x) * np.sin(m * np.pi * x / L) * np.sin(n * np.pi * x / L),
        0, L)
    return pre_1 * int_1 + pre_2 * int_2


def phi(n, x):
    return np.sqrt(2 / L) * np.sin(n * np.pi * x / L)


def psi(p, A):
    return lambda x: sum(A[n] * phi(n, x) for n in range(len(A)))


def B(F_Psi, f_psi):
    return simpson(lambda x: F_Psi(x) * f_psi(x), 0, L)


def PSI_init(x):
    if 0 < x < L / 2:
        return np.sqrt(12 / L**3) * x
    elif L / 2 < x < L:
        return np.sqrt(12 / L**3) * (L - x)
    else:
        return 0

X = np.linspace(0, L, 50)
T = np.linspace(0, 1.17e-16, 40)

fig, ax = pylab.subplots()
plot, = pylab.plot(X, [np.nan] * len(X), animated=True)


def initialize():
```

```
    ax.set_xlim(0, L)
    plot.set_ydata([np.nan] * len(X))
    plot.set_xdata(X)
    return plot,


def animate(t):
    # print(t)
    y_vals = [np.fabs(np.real(PSI(x, t)))**2 for x in X]
    pylab.title("{:e}".format(t))
    plot.set_ydata(y_vals)
    plot.set_xdata(X)
    # print(simpson(lambda x: PSI(x, 0)**2, 0, L, 500))
    return plot,


def main():
    global PSI
    V = lambda x: a * x / L
    eigen_1 = []
    eigen_2 = []
    vec_1 = []
    vec_2 = []

    def cd():
        arr1 = np.array([[H(x, y, V) for x in range(0, 11)] for y in range(0, 11)])
        arr2 = np.array([[H(x, y, V) for x in range(0, 30)] for y in range(0, 30)])
        E1, Vec1 = np.linalg.eigh(arr1)
        E2, Vec2 = np.linalg.eigh(arr2)
        E1 /= 1.6022e-19
        E2 /= 1.6022e-19
        count = 0
        Vec1 = np.transpose(Vec1)
        Vec2 = np.transpose(Vec2)
        for i in range(len(E1)):
            if np.fabs(E1[i]) > 1e-10:
                eigen_1.append(E1[i])
                vec_1.append(Vec1[i])
                count += 1
            if count == 10:
                break
        count = 0
        for i in range(len(E2)):
            if np.fabs(E2[i]) > 1e-10:
                eigen_2.append(E2[i])
                vec_2.append(Vec2[i]);
                count += 1
        print(eigen_1)
        print(eigen_2)
        print("The much larger matrix makes very little difference to the accuracy")

    def e():
        psis = [psi(p, vec_2[p]) for p in range(3)]
        X = np.linspace(0, L, 100)
        pylab.plot(X, [psis[0](x)**2 for x in X])
        pylab.plot(X, [psis[1](x)**2 for x in X])
        pylab.plot(X, [psis[2](x)**2 for x in X])
        pylab.show()
        # PSI = lambda x, t: sum([B(init, psi(n, Vec[n]))*np.exp(-1j*E[n]*t/hb)*psi(n,Vec[n])(x) for n in range(len(E))])
        # pass

    cd()
```

```
    # e()
    # V = lambda x: 0
    # Vec = np.transpose(Vec)
    # E /= 1.6022e-19
    # print(E)
    # psis = [psi(p, Vec[p]) for p in range(len(E))]
    init = lambda x: np.sqrt(12 / L**3) * x if x < L / 2 else np.sqrt(12 / L**3) * (L - x)
    PSI = lambda x, t: sum([B(init, psi(n, vec_2[n]))*np.exp(-1j*eigen_2[n]*t/hb)*psi(n,vec_2[n])(x) for n in range(len(eigen_2))])
    # PSI = lambda x, t: sum([B(init, psi(n, vec_1[n]))*np.exp(-1j*eigen_1[n]*t/hb)*psi(n,vec_1[n])(x) for n in range(len(eigen_1))])
    pylab.plot(X, [PSI(x, 0)**2 for x in X])

    # print(PSI(L/2, 0))
    # print(simpson(lambda x: PSI(x, 0)**2, 0, L, 500))
    # pylab.show()
    #
    # print(T)
    for i, t in enumerate(T):
        print(i, t)
        pylab.plot(X, [np.abs(PSI(x, t))**2 for x in X])
        # print(simpson(lambda x: np.abs(PSI(x, t))**2, 0, L, 100))
        pylab.savefig("anim/{}.png".format(i))
        pylab.clf()
        # pylab.show()
    # ani = animation.FuncAnimation(
    #     fig, animate, init_func=initialize, interval=60, blit=True, frames=T)
    # ani.save("anim.mp4")
    # pylab.show()


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