```python
import numpy as np
from scipy.fft import fft, ifft, fftfreq, fftshift, ifftshift

# Define phi in 2D
def phi2(s, R):
            return 2*R*np.sinc(R*s)

n = 2**8+1 # Number of discretisation points
N = (n-1)//2
u = np.linspace(-1, 1, n) # Set up velocity space

# Parameters
T = np.pi
R = 2*T/(1+3*np.sqrt(2))*2

# Initial distribution f(u, t=0)
f = np.zeros(n)
f[n//2 - n//10:n//2 + n//10] = 1
f0 = np.sum(f) # Initial density

# Compute Fourier frequencies
freq = fftshift(fftfreq(n))
freq = freq*N/np.max(freq)

# Time step and number of time steps
dt = 1e-2
nt = 1000

# Integrate in Time
for ti in range(nt):

    # Implement spectral algorithm at each time step
            f_hat = fftshift(fft(f))
            Q_hat = np.zeros(n)
            for k in range(n):
                        if k <= N:
                                    f_hat_l = f_hat[:N+k+1]
                                    l = freq[:N+k+1]

                                    f_hat_m = f_hat[N+k::-1]
                                    m = freq[N+k::-1]
                        else:
                                    f_hat_l = f_hat[k-N:]
                                    l = freq[k-N:]

                                    f_hat_m = f_hat[k-N:][::-1]
                                    m = freq[k-N:][::-1]

                        beta_lm = np.pi/2*(phi2(l, R) + phi2(-m, R))
                        Q_hat[k] = np.sum(np.real(beta_lm*f_hat_l*f_hat_m))

    # Inverse FFT to obtain Q(f, f) and update f by Euler
            Q = np.real(ifft(ifftshift(Q_hat)))
            Q = np.roll(Q, -1)
            f += dt*Q

    # Hack to preserve density
            ffac = np.sum(f)/f0
            f /= ffac
```