```python
import numpy as np

def RK4(t0, y0, f, h, tf):
    tRange = np.arange(t0, tf, h)
    y = np.zeros((len(tRange), len(y0)))  # Store results for all time steps
    y[0] = y0
    k = np.zeros((len(y0), 4))  # Store the four k values for each equation

    for i, t in enumerate(tRange[:-1]):  # Iterate through the time steps
        #Find k constants for each dependent variable
        k[:, 0] = h * np.array([f_i(t, *y[i]) for f_i in f])
        k[:, 1] = h * np.array([f_i(t + h / 2, *(y[i] + k[:, 0] / 2)) for f_i in f])
        k[:, 2] = h * np.array([f_i(t + h / 2, *(y[i] + k[:, 1] / 2)) for f_i in f])
        k[:, 3] = h * np.array([f_i(t + h, *(y[i] + k[:, 2])) for f_i in f])
        y[i + 1] = y[i] + (k[:, 0] + 2 * k[:, 1] + 2 * k[:, 2] + k[:, 3]) / 6  # Update y

    return tRange, y
```