```python
import os
from scipy.integrate import odeint
import numpy as np
import matplotlib.pyplot as plt


def plot_dynamical_system(fun):
    fig = plt.figure()
    ax = plt.axes(projection='3d')
    ax.plot3D(fun[:, 0], fun[:, 1], fun[:, 2], 'green')
    ax.set_title('Lorenz flow')
    plt.show()


def model(t, state, sigma, b, r):
    x, y, z = state
    xdot = sigma*(y-x)
    ydot = r*x - y - x*z
    zdot = x*y - b*z
    return [xdot, ydot, zdot]


def compute_eig(points, sigma, b, r, dt, t_max, N):
    eigenvalues = np.zeros(3)
    Q = np.eye(3)
    lambda_history = np.zeros((N, 3))

    for index in range(N):
        M = np.eye(3) + np.array([[-sigma, sigma, 0],
                                 [r-points[index, 2], -1, -points[index, 0]],
                                 [points[index, 1], points[index, 0], -b]])*dt
        Q, R = np.linalg.qr(np.matmul(M, Q))
        eigenvalues += np.log(np.abs(np.array([R[0, 0], R[1, 1], R[2, 2]])))
        lambda_history[index] = eigenvalues/(index+1)
    eigenvalues /= t_max
    lambda_history /= dt
    print(sorted(eigenvalues, reverse=True))

    return eigenvalues, lambda_history


def plot_eigenvalues(lambda_history):
    fig, ax = plt.subplots(1, 1, figsize=(10, 10))
    ax.set_xscale('log')
    ax.plot(lambda_history[:, 0], 'b', label=r'$\lambda_1$')
    ax.plot(lambda_history[:, 1], 'b', label=r'$\lambda_2$')
    ax.plot(lambda_history[:, 2], 'b', label=r'$\lambda_3$')
    plt.title("Eigenvalues over time")
    plt.xlabel("Time t")
    plt.ylabel("Eigenvalues")
    fig.tight_layout()
    plt.show()


def main():
    t_max = 100
    dt = 10**-4
    sigma = 10
    b = 3
    r = 28
    N = int(t_max/dt)

    # Create initial transient and take last point
    start_point = np.array([1, 1, 1])
    t = np.linspace(0, 30, 1000)
    points = odeint(model, start_point, t, (sigma, b, r), tfirst=True, full_output=0)
    start_point = points[-1]

    t = np.linspace(0, t_max, N)
    points = odeint(model, start_point, t, (sigma, b, r), tfirst=True)
    plot_dynamical_system(points)
    eigenvalues, lambda_history = compute_eig(points, sigma, b, r, dt, t_max, N)
```

```python
    plot_eigenvalues(lambda_history)


if __name__ == '__main__':
    os.chdir(os.path.dirname(__file__))
    main()
```