Imports

```
import numpy as np
import matplotlib.pyplot as plt
from tqdm.auto import tqdm

sigma = 10
r = 28
b = 8/3

def lorenz_step(t, x1, x2, x3, dt, sigma, rho, beta):
    """Calculate the next step in the Lorenz system."""
    dx1 = sigma * (x2 - x1) * dt
    dx2 = (x1 * (rho - x3) - x2) * dt
    dx3 = (x1 * x2 - beta * x3) * dt

    return t + dt, x1 + dx1, x2 + dx2, x3 + dx3

def lorenz(t0, x1_0, x2_0, x3_0, dt, sigma, rho, beta, iter_num):
    """Calculate the evolution of the Lorenz system."""
    t = np.zeros(iter_num + 1)
    x1 = np.zeros(iter_num + 1)
    x2 = np.zeros(iter_num + 1)
    x3 = np.zeros(iter_num + 1)

    t[0], x1[0], x2[0], x3[0] = t0, x1_0, x2_0, x3_0
    for i in range(iter_num):
        t[i + 1], x1[i + 1], x2[i + 1], x3[i + 1] = (
            lorenz_step(t[i], x1[i], x2[i], x3[i], dt, sigma, rho,
beta)
        )

    return t, x1, x2, x3
```

Generate training set

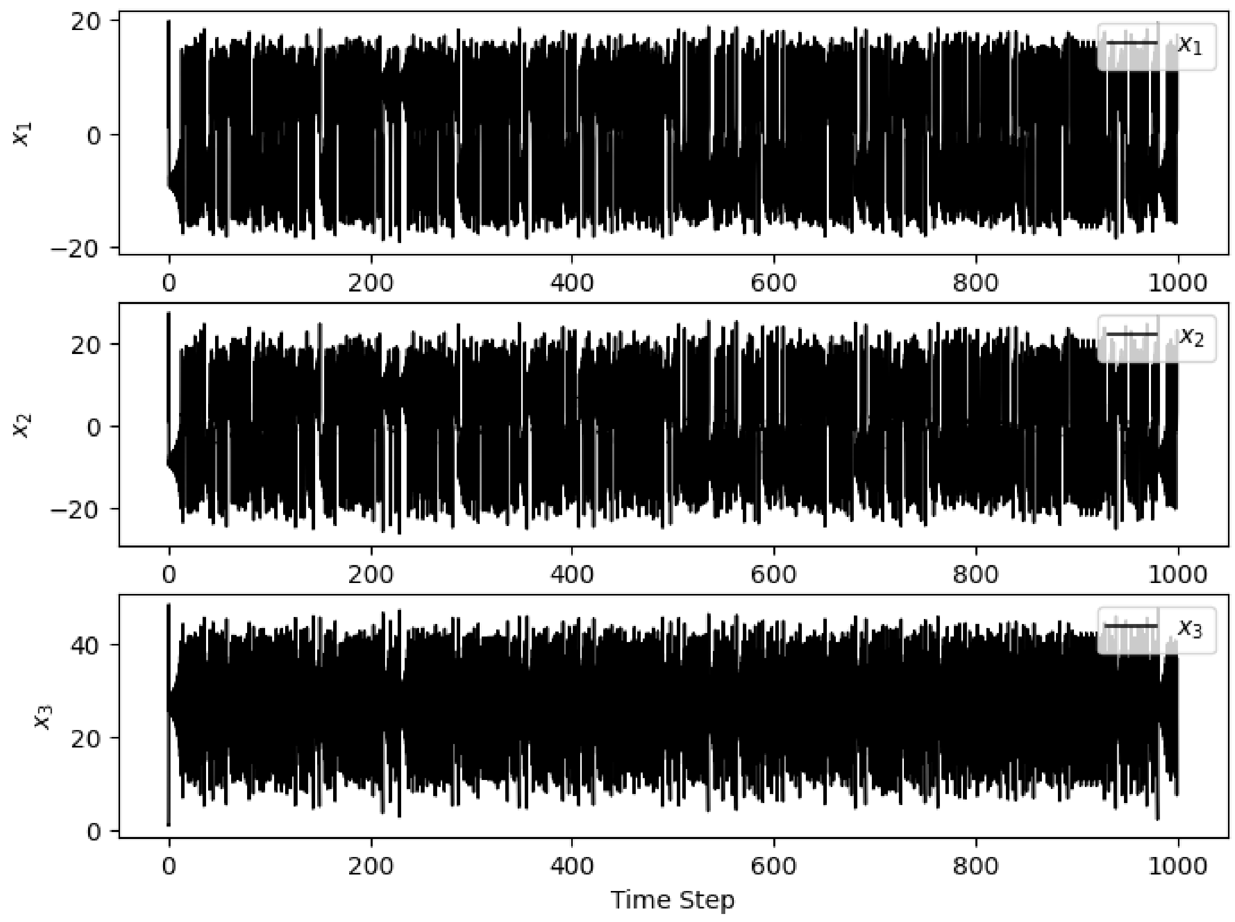
```
num_series = 1
t_max = 1000
dt = 0.001
iter_num = int(t_max/dt)
initialpart = int(20/dt)
std = 0.05
series = []
for _ in tqdm(range(num_series)):
    t, x1, x2, x3 = lorenz(t0=0, x1_0=np.random.normal(1, std),
x2_0=np.random.normal(1, std), x3_0=np.random.normal(1, std), dt=dt,
                        sigma=sigma, rho=r, beta=b, iter_num=iter_num)
    series.append([x1, x2, x3])
series = np.array(series)
```

{"model_id":"d2fbd089554a46c69aa4b91b68bdfa96","version_major":2,"version_minor":0}

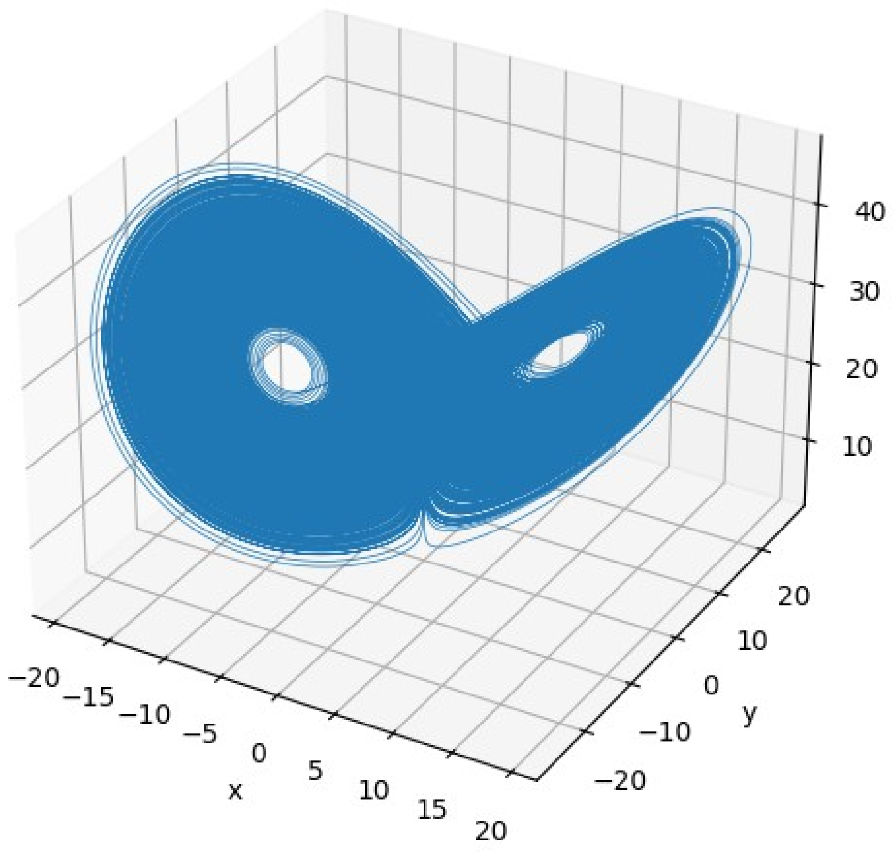
```python
x1 = series[0, 0]
x2 = series[0, 1]
x3 = series[0, 2]
x_zip = list(zip(x1, x2, x3))
X = np.array(x_zip)

fig, axs = plt.subplots(3, 1, figsize=(8, 6))
X = np.array(x_zip)
for i in range(3):
    axs[i].plot(t, X[:,i], label=fr"$x_{i+1}$", color="k", lw=1.1)
    axs[i].set_ylabel(fr"$x_{i+1}$")
    axs[i].legend()
axs[2].set_xlabel("Time Step")
```

```
Text(0.5, 0, 'Time Step')
```

```
/usr/local/lib/python3.10/dist-packages/IPython/core/
pylabtools.py:151: UserWarning: Creating legend with loc="best" can be
slow with large amounts of data.
  fig.canvas.print_figure(bytes_io, **kw)
```



```
fig= plt.figure(figsize=(8, 6))
ax = fig.add_subplot(projection="3d")
ax.plot(x1[initialpart:], x2[initialpart:], x3[initialpart:], lw=0.5)

ax.set_xlabel("x")
ax.set_ylabel("y")
ax.set_zlabel("z")

plt.show()
```



```
t, x1,x2,x3 = t[initialpart:], x1[initialpart:], x2[initialpart:],
x3[initialpart:]
X = np.column_stack((x1, x2, x3))

def jac(x1, x2, x3, sigma, r, b):
    return np.array([[-sigma, sigma, 0], [r-x3, -1, -x1], [x2, x1, -
b]])

Q = np.eye(3)
Id = np.eye(3)
lambdas = np.zeros(3)
lambdas_history = np.zeros((len(t),3))

for n in tqdm(range(len(t))):
    x1s, x2s, x3s = X[n,0], X[n,1], X[n,2]

    M = Id + jac(x1s, x2s, x3s, sigma, r, b)*dt
    Q, R = np.linalg.qr(np.matmul(M, Q))

    lambdas[0] += np.log(np.abs(R[0, 0]))
    lambdas[1] += np.log(np.abs(R[1, 1]))
    lambdas[2] += np.log(np.abs(R[2, 2]))
    current_time = (n+1)*dt
    lambdas_history[n] = lambdas / current_time

lambdas = lambdas/(len(t)*dt)
print(f'Lyapunov exponents in x1: {lambdas[0]:.2f}, in x2:
{lambdas[1]:.2f} and x3: {lambdas[2]:.2f}')
print(f'The largest Lyapunov exponent is found to be approximatly
{max(lambdas):.2f}')
```

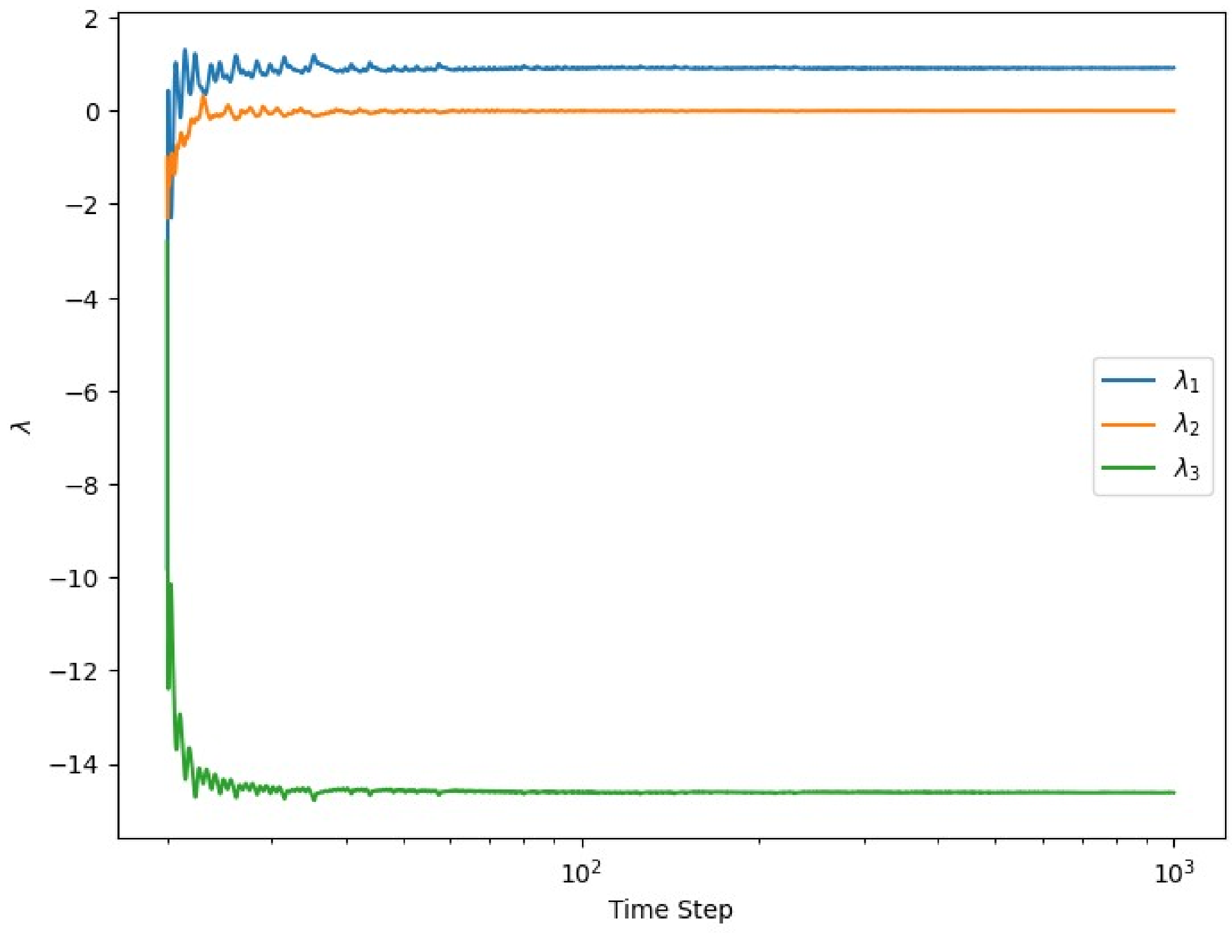
{"model_id":"38d289d5f1804af1bc42e71b8d164e8f","version_major":2,"vers
ion_minor":0}

```
Lyapunov exponents in x1: 0.92, in x2: -0.00 and x3: -14.62
The largest Lyapunov exponent is found to be approximatly 0.92
```

```
theortic_sum = -(sigma+1+b)
simulation_sum = np.sum(lambdas)
print(f"Theoretic Sum of Lyapunov Exponents: {theortic_sum:.2f}")
print(f"Simulation Sum of Lyapunov Exponents: {simulation_sum:.2f}")
```

```
Theoretic Sum of Lyapunov Exponents: -13.67
Simulation Sum of Lyapunov Exponents: -13.70
```

```
plt.figure(figsize=(8, 6))
for i in range(3):
    plt.semilogx(t, lambdas_history[:, i], label=fr"$\lambda_{i+1}$")
plt.ylabel(fr"$\lambda$")
plt.xlabel("Time Step")
plt.legend()
plt.show()
```



New parameters

```
sigma = 10
r = 28
b = 17/6

num_series = 1
t_max = 1000
dt = 0.001
iter_num = int(t_max/dt)
initialpart = int(20/dt)
std = 0.05
series = []
for _ in tqdm(range(num_series)):
    t, x1, x2, x3 = lorenz(t0=0, x1_0=np.random.normal(1, std),
x2_0=np.random.normal(1, std), x3_0=np.random.normal(1, std), dt=dt,
                        sigma=sigma, rho=r, beta=b, iter_num=iter_num)
    series.append([x1, x2, x3])
series = np.array(series)
```

{"model_id":"084920bbfbf340f0a315b1ef09e02bc8","version_major":2,"version_minor":0}

```
x1 = series[0, 0]
x2 = series[0, 1]
x3 = series[0, 2]
x_zip = list(zip(x1, x2, x3))
X = np.array(x_zip)

t, x1,x2,x3 = t[initialpart:], x1[initialpart:], x2[initialpart:],
x3[initialpart:]
X = np.column_stack((x1, x2, x3))

def jac(x1, x2, x3, sigma, r, b):
    return np.array([[-sigma, sigma, 0], [r-x3, -1, -x1], [x2, x1, -
b]])

Q = np.eye(3)
Id = np.eye(3)
lambdas = np.zeros(3)
lambdas_history = np.zeros((len(t),3))

for n in tqdm(range(len(t))):
    x1s, x2s, x3s = X[n,0], X[n,1], X[n,2]

    M = Id + jac(x1s, x2s, x3s, sigma, r, b)*dt
    Q, R = np.linalg.qr(np.matmul(M, Q))

    lambdas[0] += np.log(np.abs(R[0, 0]))
    lambdas[1] += np.log(np.abs(R[1, 1]))
    lambdas[2] += np.log(np.abs(R[2, 2]))
    current_time = (n+1)*dt
    lambdas_history[n] = lambdas / current_time

lambdas = lambdas/(len(t)*dt)
print(f'Lyapunov exponents in x1: {lambdas[0]:.2f}, in x2:
{lambdas[1]:.2f} and x3: {lambdas[2]:.2f}')
print(f'The largest Lyapunov exponent is found to be approximatly
{max(lambdas):.2f}')
```

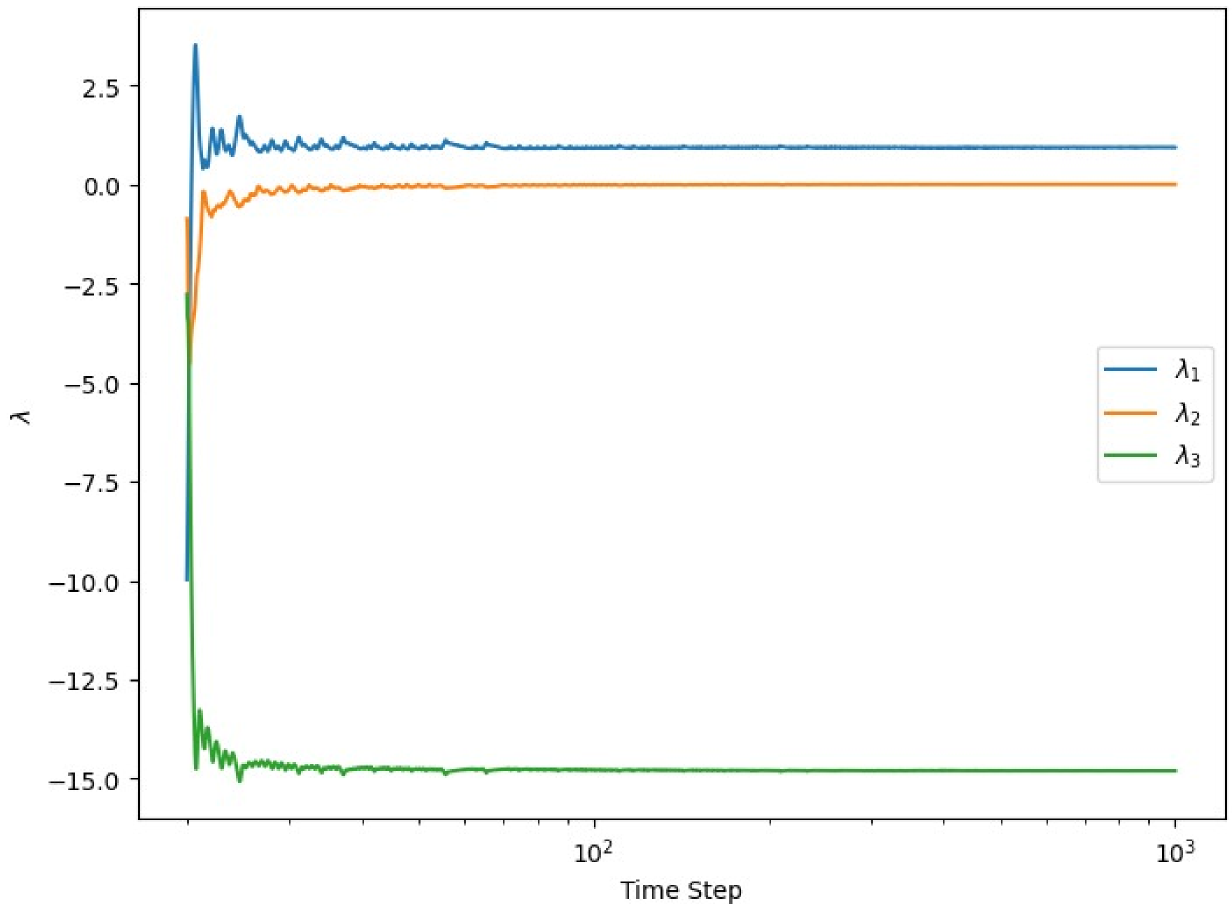
{"model_id":"c60653d49f014211ab65217e7d6b55a8","version_major":2,"version_minor":0}

```
Lyapunov exponents in x1: 0.94, in x2: -0.00 and x3: -14.80
The largest Lyapunov exponent is found to be approximatly 0.94
```

```
theortic_sum = -(sigma+1+b)
simulation_sum = np.sum(lambdas)
print(f"Theoretic Sum of Lyapunov Exponents: {theortic_sum:.2f}")
print(f"Simulation Sum of Lyapunov Exponents: {simulation_sum:.2f}")
```

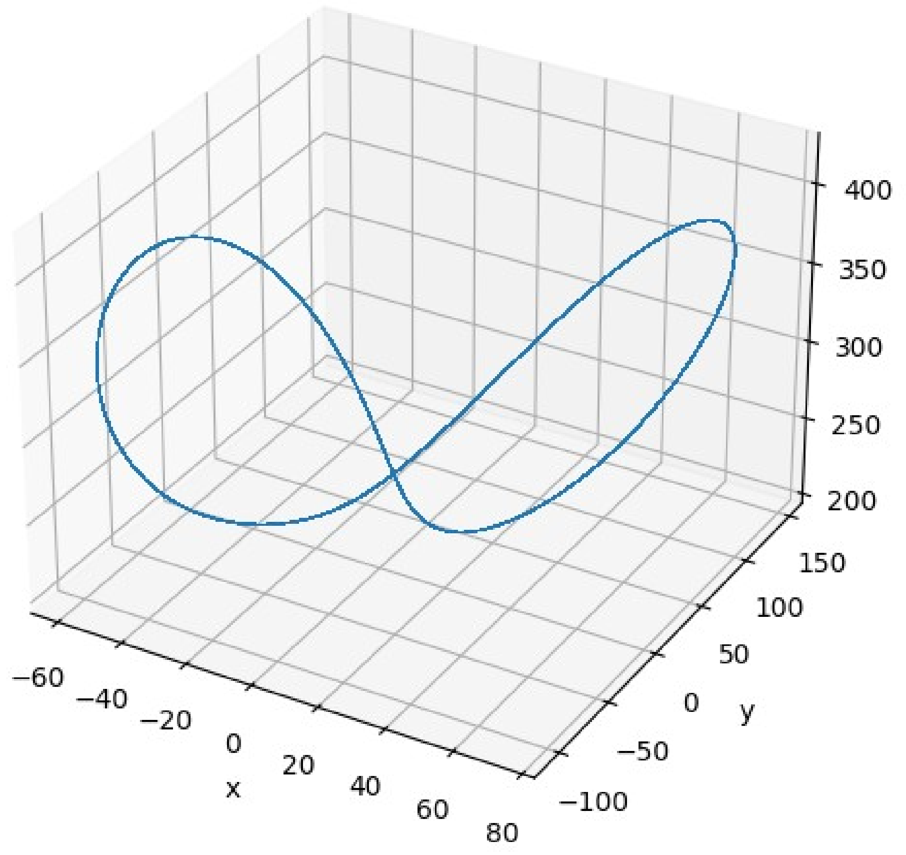
```
Theoretic Sum of Lyapunov Exponents: -13.83
Simulation Sum of Lyapunov Exponents: -13.87
```

```
plt.figure(figsize=(8, 6))
for i in range(3):
    plt.semilogx(t, lambdas_history[:, i], label=fr"$\lambda_{i+1}$")
plt.ylabel(fr"$\lambda$")
plt.xlabel("Time Step")
plt.legend()
plt.show()
```



New again

```
from scipy.integrate import solve_ivp

# current lorenz system where not accurate enough with the new set of
parameters. Therefore we instead solve the system with scipy
def lorenz_system(t, state, sigma, rho, beta):
    x1, x2, x3 = state[:3]

    dx1 = sigma * (x2 - x1)
    dx2 = (x1 * (rho - x3) - x2)
    dx3 = (x1 * x2 - beta * x3)

    J = jac(x1, x2, x3, sigma, rho, beta)

    M = state[3:].reshape((3, 3))

    dMdt = J @ M
    return np.concatenate(([dx1, dx2, dx3], dMdt.flatten()))

sigma = 16
r = 310
b = 5

num_series = 1
t_max = 1000
dt = 0.0001
iter_num = int(t_max/dt)
initialpart = int(10/dt)
std = 0.05
t_eval = np.arange(0, t_max, dt)
t_span = (0, t_max)

initial_state = np.array([np.random.normal(1, std),
np.random.normal(1, std), np.random.normal(1, std)] +
list(np.eye(3).flatten()))

solution = solve_ivp(
    lorenz_system, t_span, initial_state, args=(sigma, r, b),
    t_eval=t_eval, method='RK45', max_step=dt
)

t = solution.t
x1, x2, x3 = solution.y[0], solution.y[1], solution.y[2]

t, x1,x2,x3 = t[initialpart:], x1[initialpart:], x2[initialpart:],
x3[initialpart:]
X = np.column_stack((x1, x2, x3))

fig= plt.figure(figsize=(8, 6))
ax = fig.add_subplot(projection="3d")
ax.plot(x1[initialpart:], x2[initialpart:], x3[initialpart:], lw=0.5)

ax.set_xlabel("x")
ax.set_ylabel("y")
ax.set_zlabel("z")

plt.show()
```

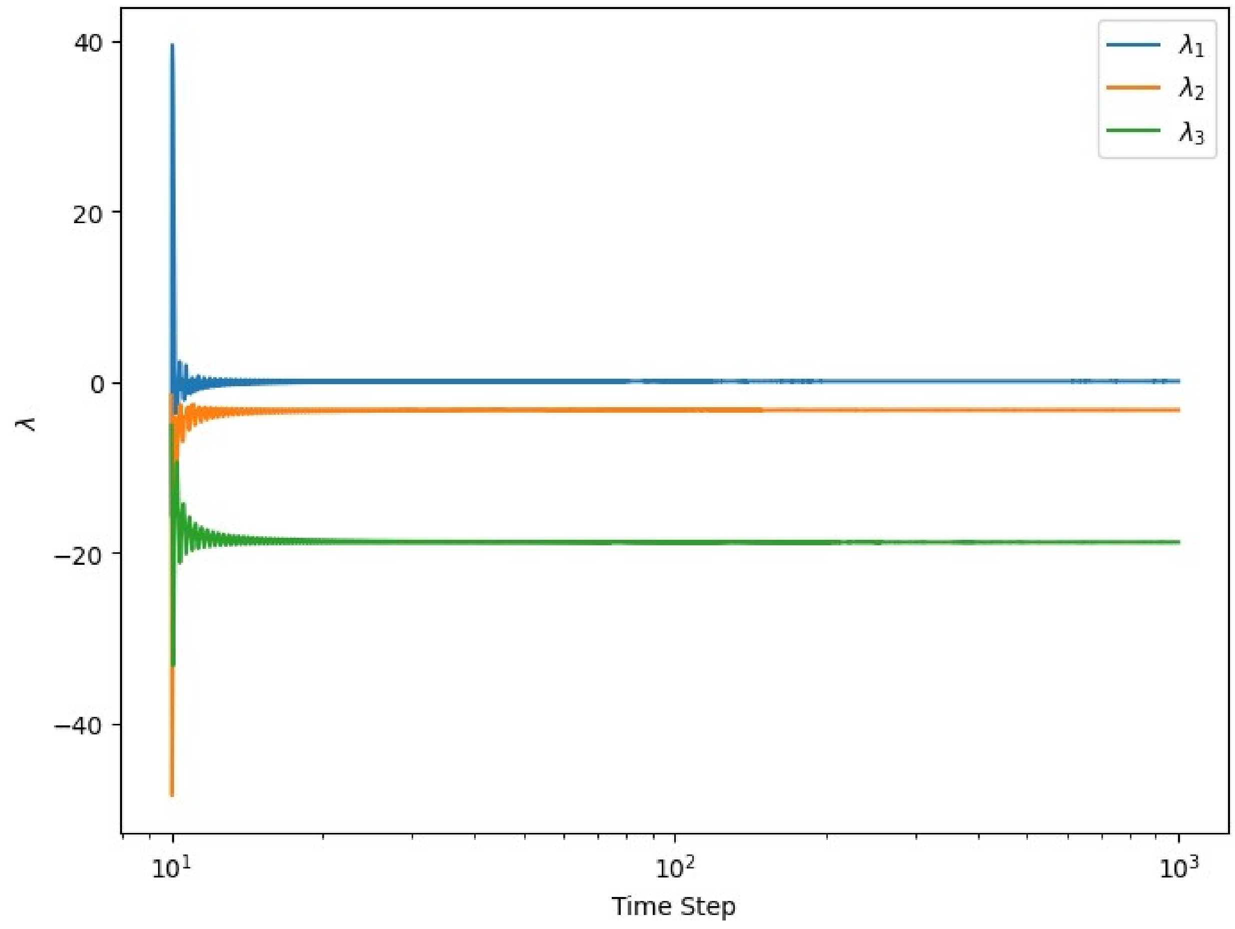


```
def jac(x1, x2, x3, sigma, r, b):
    return np.array([[-sigma, sigma, 0], [r-x3, -1, -x1], [x2, x1, -
b]])

Q = np.eye(3)
Id = np.eye(3)
lambdas = np.zeros(3)
lambdas_history = np.zeros((len(t),3))

for n in tqdm(range(len(t))):
    x1s, x2s, x3s = X[n,:] #######################

    M = Id + jac(x1s, x2s, x3s, sigma, r, b)*dt
    Q, R = np.linalg.qr(np.matmul(M, Q))

    lambdas[0] += np.log(np.abs(R[0, 0]))
    lambdas[1] += np.log(np.abs(R[1, 1]))
    lambdas[2] += np.log(np.abs(R[2, 2]))
    current_time = (n+1)*dt
    lambdas_history[n] = lambdas / current_time
```

```
lambdas = lambdas/(len(t)*dt)
print(f'Lyapunov exponents in x1: {lambdas[0]:.2f}, in x2:
{lambdas[1]:.2f} and x3: {lambdas[2]:.2f}')
print(f'The largest Lyapunov exponent is found to be approximatly
{max(lambdas):.2f}')
```

```
100%|██████████| 9900000/9900000 [12:00<00:00, 13748.88it/s]

Lyapunov exponents in x1: 0.11, in x2: -3.26 and x3: -18.75
The largest Lyapunov exponent is found to be approximatly 0.11
```

```
plt.figure(figsize=(8, 6))
for i in range(3):
    plt.semilogx(t, lambdas_history[:, i], label=fr"$\lambda_{i+1}$")
plt.ylabel(fr"$\lambda$")
plt.xlabel("Time Step")
plt.legend()
plt.show()
```



```python
theortic_sum = -(sigma+1+b)
simulation_sum = np.sum(lambdas)
print(f"Theoretic Sum of Lyapunov Exponents: {theortic_sum:.2f}")
print(f"Simulation Sum of Lyapunov Exponents: {simulation_sum:.2f}")
```

```
Theoretic Sum of Lyapunov Exponents: -22.00
Simulation Sum of Lyapunov Exponents: -21.90
```