In [39]:

```python
import numpy as np
import matplotlib.pyplot as plt

# Defining system parameters
G = np.array([[1, 0.2, 0.1],
              [0.1, 2, 0.1],
              [0.3, 0.1, 3]])
G_prime = np.array([[0, G[0, 1]/G[0, 0], G[0, 2]/G[0, 0]],
                    [G[1, 0]/G[1, 1], 0, G[1, 2]/G[1, 1]],
                    [G[2, 0]/G[2, 2], G[2, 1]/G[2, 2], 0]])
B = np.array([3.6, 1.8, 1.2])
u = 0.01

# Defining simulation parameters (states and other quantities of interest)
num_steps = 50
p = np.zeros((3, num_steps + 1))  # Extra index accounting for initial state
s = np.zeros((3, num_steps))
q = np.zeros((3, num_steps))
S = np.zeros((3, num_steps))

# Initial condition
p[:, 0] = np.array([0.1, 0.1, 0.1])

# Simulating the system
for k in range(num_steps):
    p[:, k + 1] = G_prime @ p[:, k] + B * u  # Updating p value for each step
    s[:, k] = np.array([G[0, 0] * p[0, k],  # s - signal power
                        G[1, 1] * p[1, k],
                        G[2, 2] * p[2, k]])
    q[:, k] = u**2 + np.array([G[0, 1] * p[1, k] + G[0, 2] * p[2, k],  # q - noise plus interfere
                       G[1, 0] * p[0, k] + G[1, 2] * p[2, k],
                       G[2, 0] * p[2, k] + G[2, 1] * p[1, k]])
    S[:, k] = np.divide(s[:, k], q[:, k])  # SINR


# Plotting the values of p
time = np.arange(num_steps + 1)
plt.figure(figsize=(10, 6))
plt.plot(time, p[0, :], label='p1')
plt.plot(time, p[1, :], label='p2')
plt.plot(time, p[2, :], label='p3')
plt.xlabel('Time Step')
plt.ylabel('State Variables')
plt.title('State Variables over Time | Initial p1 = p2 = p3 = 0.1, gamma = 3')
plt.legend()
plt.grid(True)
plt.show()

# Plotting the values of S
plt.figure(figsize=(10, 6))
plt.plot(time[0: -1], S[0, :], label='S1')
plt.plot(time[0: -1], S[1, :], label='S2')
plt.plot(time[0: -1], S[2, :], label='S3')
plt.xlabel('Time Step')
plt.ylabel('SINR for each Receiver')
plt.title('SINR over Time | Initial p1 = p2 = p3 = 0.1, gamma = 3')
plt.legend()
plt.grid(True)
plt.show()
```