9. Linear dynamical systems

9.1. Linear dynamical systems

Let's simulate a time-invariant linear dynamical system

$$x_{t+1} = Ax_t, \quad t = 1, \dots, T,$$

with dynamics matrix

$$A = \begin{bmatrix} 0.97 & 0.10 & -0.05 \\ -0.30 & 0.99 & 0.05 \\ 0.01 & -0.04 & 0.96 \end{bmatrix}$$

and initial state $x_1 = (1, 0, -1)$. We store trajectory in the $n \times T$ matrix `state_traj`, with the ith column x_t. We plot the result in Figure 9.1.

In [ ]:
```
# initial state
x_1 = np.array([1,0,-1])
n = len(x_1)
T = 50
A = np.array([[0.97, 0.1, -0.05], [-0.3, 0.99, 0.05], [0.01,
↪  -0.04, 0.96]])
state_traj = np.column_stack([x_1, np.zeros((n,T-1))])
# dynamics recursion
for t in range(T-1):
    state_traj[:, t+1] = A @ state_traj[:,t]
import matplotlib.pyplot as plt
plt.ion()
plt.plot(np.arange(T), state_traj.T)
plt.legend(['(x_t)_1','(x_t)_2','(x_t)_3'])
plt.xlabel('t')
```

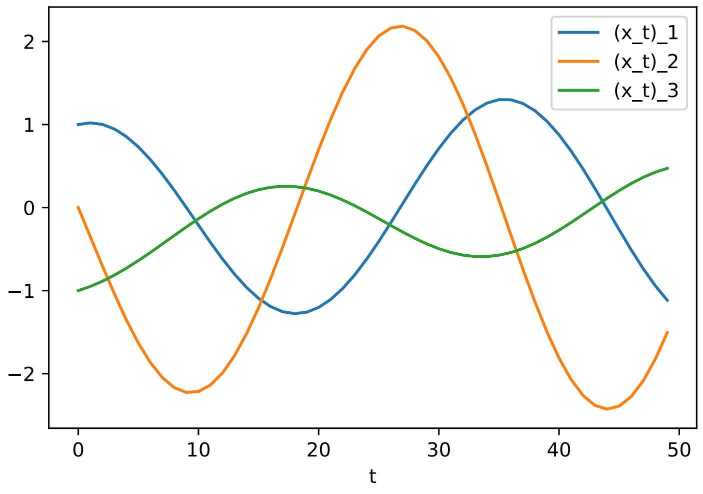

Figure 9.1.: Linear dynamical system simulation.

9.2. Population dynamics

We can create a population dynamics matrix with just one simple line of Python code. The following code predicts the 2020 population distribution in the US using the data of Section 9.2 of VMLS in the `population_data` dataset.

```
In [ ]: # Import 3 100-vector: population, birth_rate, death_rate
D = population_data()
b = D['birth_rate']
d = D['death_rate']
A = np.vstack([b, np.column_stack([np.diag(1 - d[:-1]),
↪  np.zeros((len(d) - 1))])])
x = D['population']
x = np.power(A,10) @ x
import matplotlib.pyplot as plt
plt.ion()
plt.plot(x)
plt.xlabel('Age')
plt.ylabel('Population (millions)')
plt.show()
```