Diffusive and Stochastic Processes Programming Assignment

Georgios Sevastakis

June 2024

Note: For all cases, the number of samples is 200.

Exercise 1

Our goal in the first exercise is to numerically simulate the trajectory $X(t)$ of a brownian particle in the overdamped limit and under the effect of force from the following potential by using the Euler method

$$U(X) = \frac{1}{4}X^4 - \frac{1}{2}X^2 + X \tag{1}$$

Then, the force is given by

$$F(X) = -U'(X) = -X^3 + X - 1 \tag{2}$$

The Langevin equation in this case is given by the following equation

$$\frac{dX}{dt} = -\frac{1}{\eta}(X^3 - X + 1) + \xi(t) \tag{3}$$

where η is the drag coefficient and $\xi(t)$ is a Gaussian white noise with the following properties

$$\langle \xi(t) \rangle = 0 \tag{4}$$

$$\langle\, \xi(t_1)\, \xi(t_2)\, \rangle = 2D\delta(t_1 - t_2) \tag{5}$$

where D is the diffusion coefficient.

Before continuing any further, we note that the σ to be used in the Gaussian noise is not readily available and therefore, we resort to equation 5 and compute the autocorrelation from which we ultimately get σ

$$\begin{aligned} \langle \xi^2(t) \rangle &= \sigma^2 = 2D \\ \implies \sigma &= \sqrt{2D} \end{aligned} \tag{6}$$

With the initial conditions and the parameter values given, we are now ready to integrate equation 3. We proceed to the results section

Results

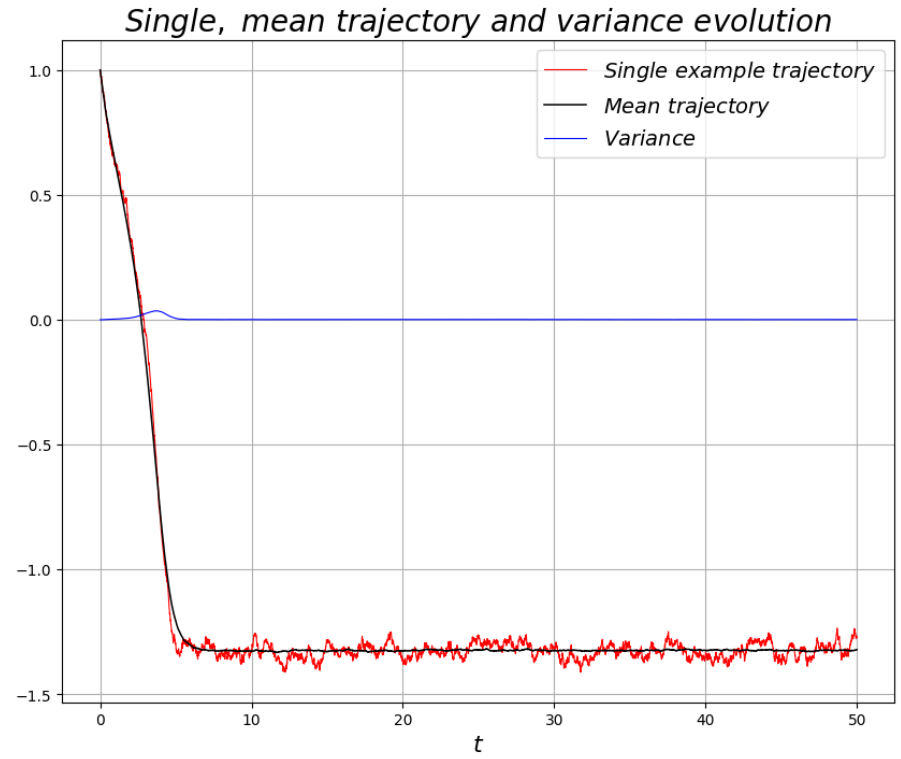

Figure 1: Single and mean trajectory, as well as the variance with respect to time.

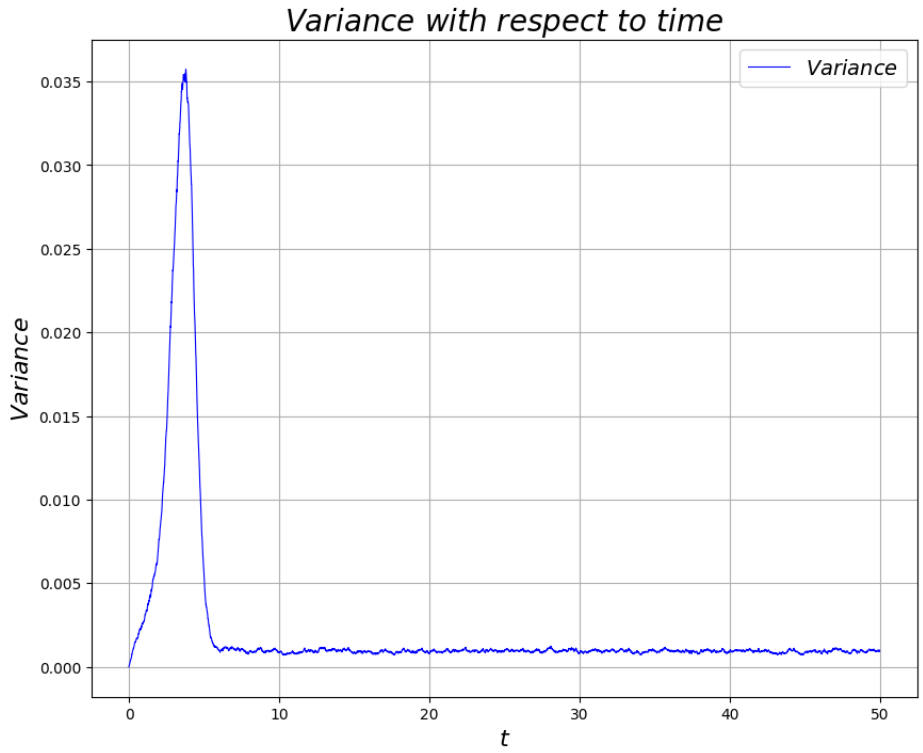

Figure 2: Variance with respect to time (Added for better visualization).

```
def Euler(dt, eta, Nt, N_trajectories):
    X_0 = np.zeros((N_trajectories, Nt))
    X_0[:, 0] = 1
    t = np.zeros(Nt)
    X = X_0
    for i in range(1, Nt):
        t[i] = t[i - 1] + dt
        xi = np.random.normal(loc = gaussian_mean, scale = gaussian_std, size =
    N_trajectories)
        X[:, i] = X[:, i - 1] - dt * (1/eta*(X[:, i - 1]**3 - X[:, i - 1] + 1) + xi)
    return X, t
```

Listing 1. Function that implements the Euler method to integrate the given Langevin equation.

Exercise 2

The goal of this exercise is to simulate an SIR–like model utilizing the Gillespie method. For each sample of the ensemble we firstly compute the propensity functions. Next, we generate a random number r_1 to determine the time step τ (essentially sampling from the exponential) and a random number r_2, which determines the fate of the evolution of N. With the initial conditions and the parameter values given, we proceed to the results section

Results

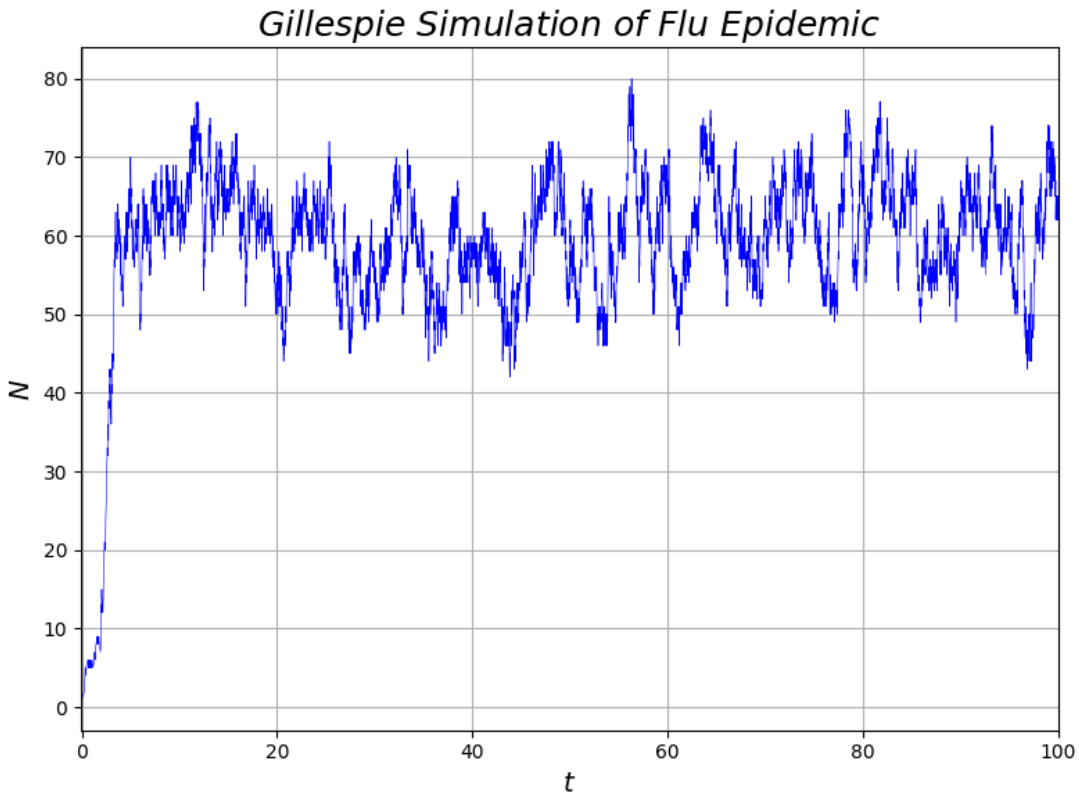

Figure 3: Gillespie simulation of a flu epidemic for a single trajectory, namely the number of infected people with respect to time.

By visual inspection, it is safe to assume that the steady state is reached by $t = 20$. Since the ensemble average matches the time average in the simple model in the steady state, we draw our attention to

the time average. After having carefully adjusted the weights due to the uneven time steps, we report the following ensemble/time mean and variance values

$$Mean = 56.51 \tag{7}$$

$$Variance = 39.49 \tag{8}$$

```
for sample in range(N_samples):
    N = N_initial
    t = 0
    times = [t]
    infected = [N]

    while t < T:
        a1 = alpha * N / Omega * (Omega - N) + epsilon * (Omega - N)
        a2 = gamma * N
        a0 = a1 + a2

        r1 = np.random.uniform(0, 1)
        r2 = np.random.uniform(0, 1)

        tau = (1 / a0) * np.log(1 / r1)
        t += tau

        if t > T:
            break

        if r2 * a0 < a1:
            N += 1
        else:
            N -= 1

        times.append(t)
        infected.append(N)

    times_ensemble.append(times)
    infected_ensemble.append(infected)
```

Listing 2. Code snippet of the Gillespie implementation for the SIR model.

Exercise 3

Exercise 3 is in a way the continuance of Exercise 2, since it revolves around the same model with the addition of the immunity into the previous model. By adding another propensity function, adjusting the sum as well as the if statements properly, we plotted the following trajectories for the Infected and Immune folks (one sample).

Results

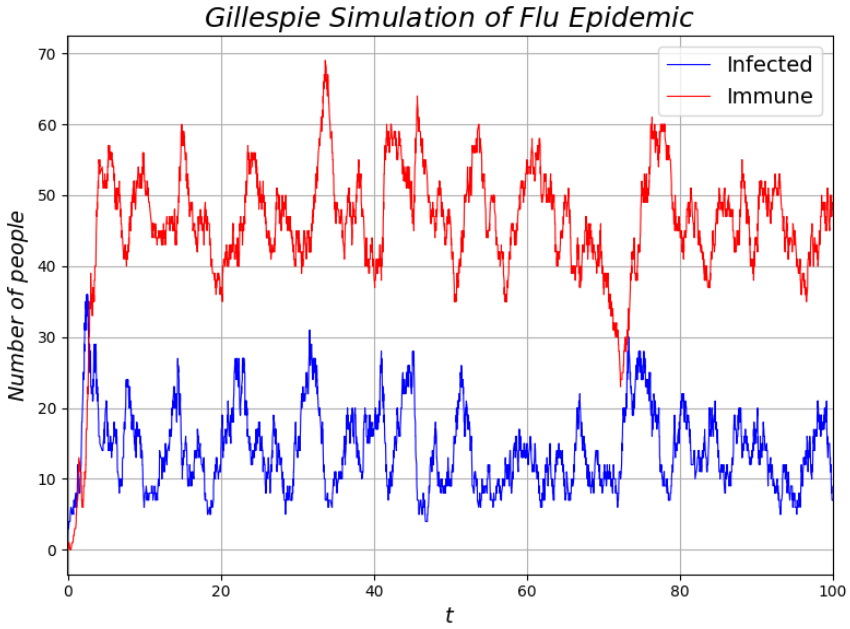

Figure 4: Gillespie simulation of a flu epidemic for a single trajectory, namely the number of infected and immune people with respect to time.

```
for sample in range(N_samples):
    M = M_initial
    N = N_initial
    t = 0
    times = [t]
    infected = [N]
    immune = [M]
    while t < T:
        a1 = alpha * N / Omega * (Omega - N - M) + epsilon * (Omega - N - M)
        a2 = gamma * N
        a3 = beta * M
        a0 = a1 + a2 +a3

        r1 = np.random.uniform(0, 1)
        r2 = np.random.uniform(0, 1)
        tau = (1 / a0) * np.log(1 / r1)
        t += tau
        if t > T:
            break
        if r2 * a0 < a1:
            N += 1
        elif r2 * a0 < a1 + a2:
            N -= 1
            M += 1
        else:
            M -= 1
        times.append(t)
        infected.append(N)
        immune.append(M)
    times_ensemble.append(times)
    infected_ensemble.append(infected)
    immune_ensemble.append(immune)
```

Listing 3. Code snippet of the Gillespie implementation for the modified SIR model.