Assignment - 3

Python is used for this assignment.

Q1. Stochastic binary decisions

Gillespie's algorithm is used to solve these problem.

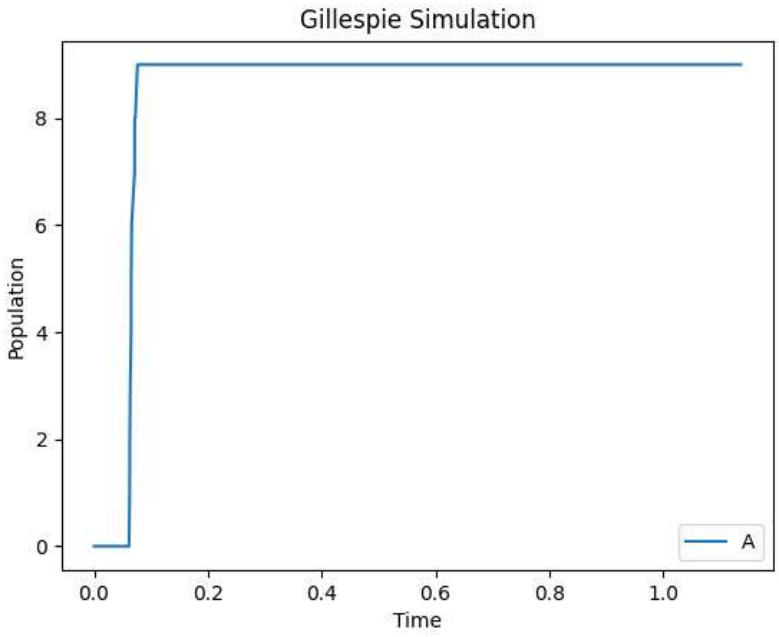

Figure 1: A1* Population Steps 30

Fig 1: Population of A1* for 30 step simulation

Fig 2: Histogram of A1* Population for 30 step simulation

Fig 3: A1* Population for 900 step simulation

Fig 4: Histogram of A1* Population for 900 step simulation

Fig 5:Population of all species for 900 step simulation

Differnetial Rate Equations: dA1*/dt = k3.E.A1 .... [1]

dA1/dt = -k3.A1.E - A1.k5.S ..... [2]

dE/dt = k1.A1 +k4.A1* -kd.E ..... [3]

dS/dt = k2.A2 -kd.S .... [4]

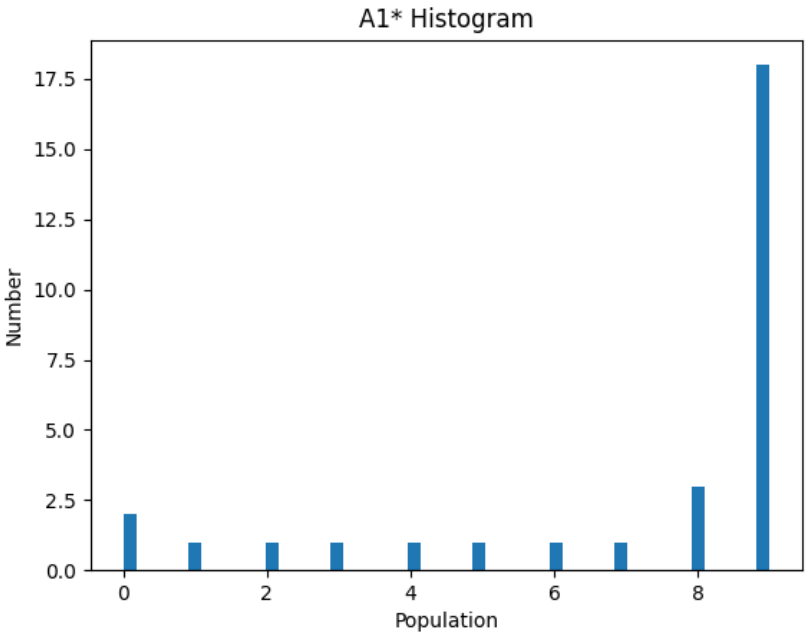

Figure 2: A1* Histogram Steps 30

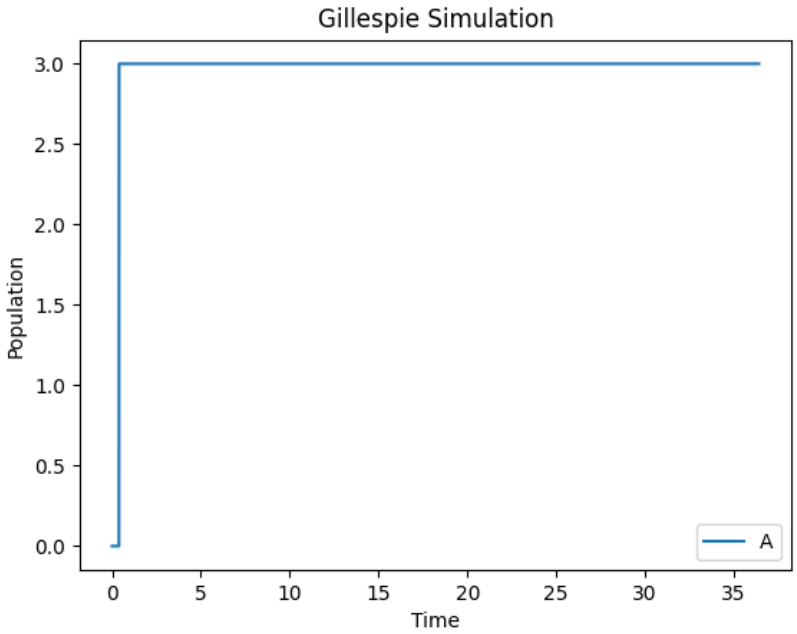

Figure 3: A1* Population Steps 900

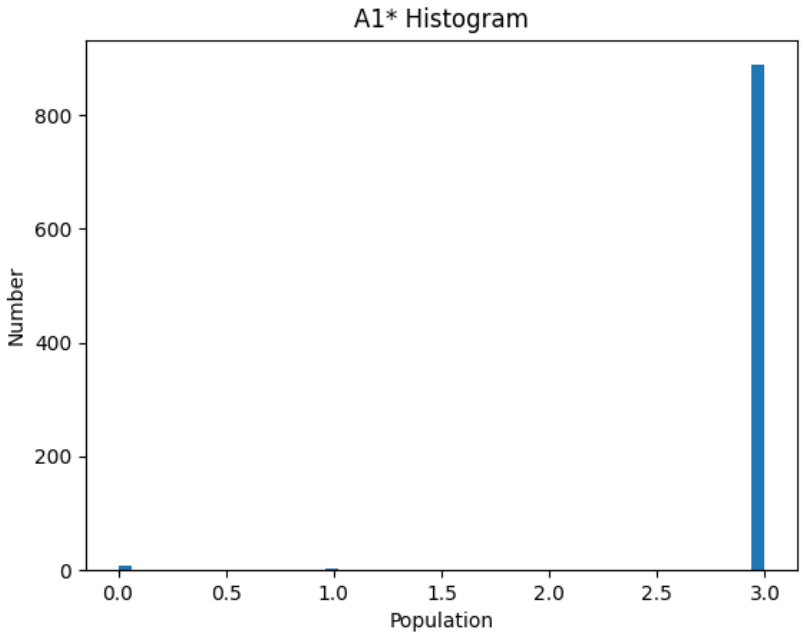

Figure 4: A1* Histogram Steps 9000

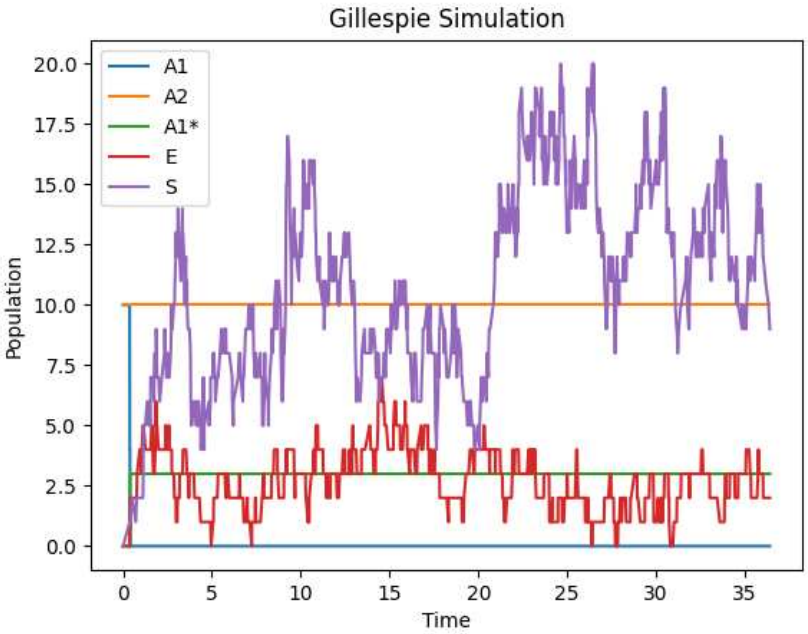

Figure 5: All Population Steps 900

Also k1=k2=k4=kd=1 and k3=100 and k5=100.

Looking at the above equation the rate of A1* is dependant upon the concentration of E and A1. Also the rate is always increasing and there is no destruction. Thus one expect a monotoneously increasing value of A1*. Also we expect a rapid growth of E at the begining as only -kd.E contribute to its destruction and should be low at the begining. Thus we expect a very large growth rate at the beginning but soon the rate is expected to drop as the concentration of A1 would deplete fast with the growth of S and E. We observe a similar situation in the grah.

Also we expect to observer a case where there is no growth of A1* if A1 decreases very rapidly. Then the concentration of E will reduce as well the A1 and we expect to observer almost no growth in the value of A1. *This is the numerical instability of the problem. Thuse we expect a constant value for the A1* or almost 0 very soon the simulation ater the begining of the simulation.

CASE-2

Very high value of A1 and A2

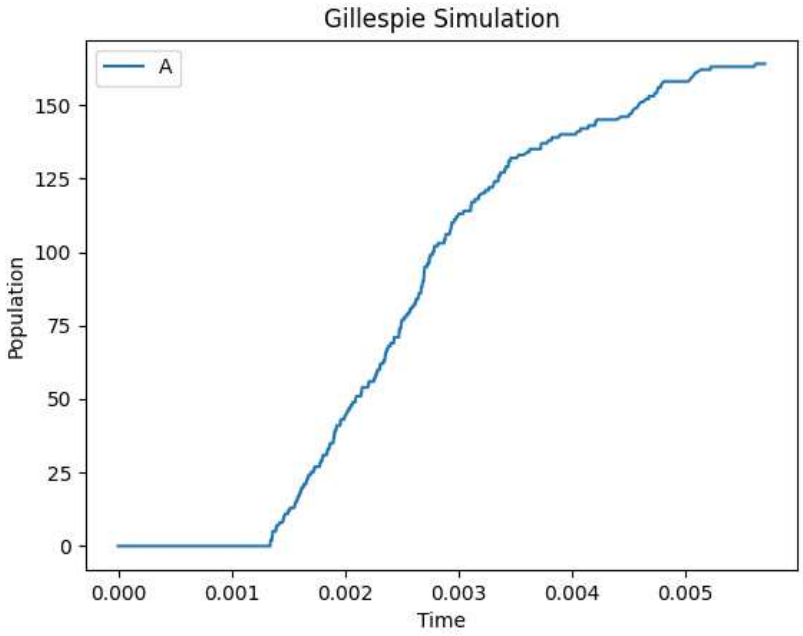

Figure 6: A1* Population for high A1 and A2

Fig 6: Population of A1* for very high value of A1 and A2

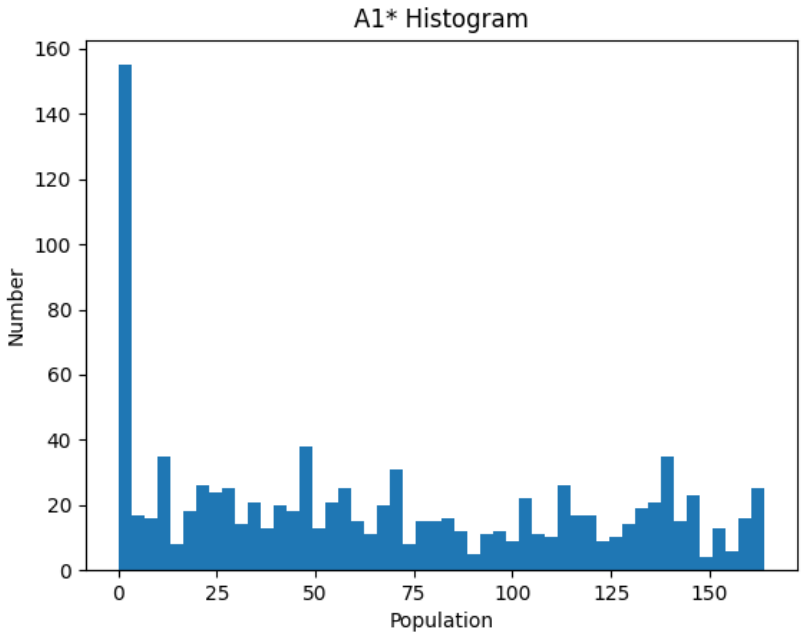

Figure 7: A1* Histogram for large values of A1

Fig 7: Population of A1* for large population of A1 and A2

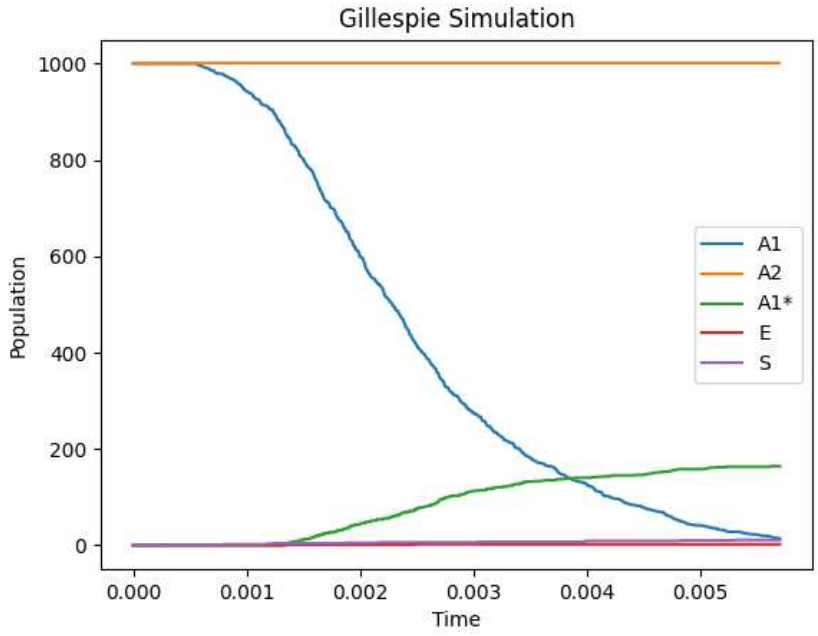

Figure 8: All Population for large values of A1*

Fig 8: Population of A1* for 30

With a very high value of A1 we don't have any instability in A1 and the value of A1 decreases smoothly following almost an exponential distribution. We expect E to have almost same behavious like previous case. Thus we expect the growth rate of A1* to be almost proportional to population of A1. Thus A1* is expected to grow for sometime and then achieve a equilibriu with near zero growth because of low population of A1. We observe similar dynamics in Figure 8.

The instability is clearly shown in the histogram of Fig 4 and Fig 7, where we expect a suddent in crease of A1* for the former case and a stable solution for the later case as depicted by Fig 7, except for the first bin we observe an equal distribution of the population.

In Figure 3 and 1 the legend is A1* instesd of A1

Q2. The genetic toggle switch and stochastic bistability

The constrains:

gA+gA* = 1

gB+gB* = 1

should be followed automatically if we have correct initial value and made no mistakes defining the reactions.

The last constrain of exclusive switch was implemented by reducing the propensity of formation of gB* to 0 if gA* is present and vice-versa for gA*.

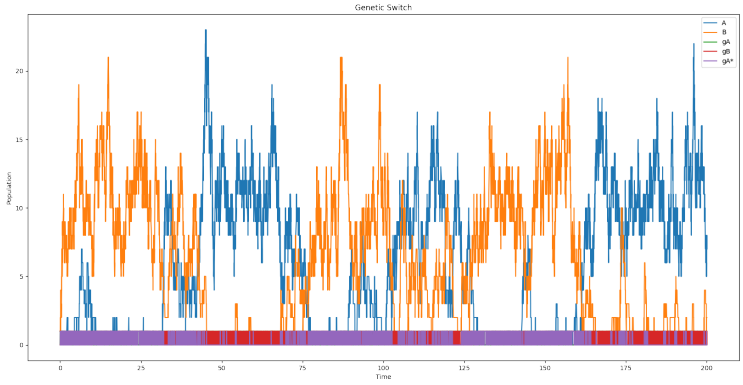

Figure 9: Genetic switch

Fig 9: Genetic switch for kd=1,kp=10,kf=20,kr=15

As can be seen in the Fig 9, a constant switch in concentration of species A and B can be observed.

Fig 10: Genetic Switch with initial population of A and B = 100

Fig 11: Genetic Switch with initial population of A and B = 1000

With a very large population of A and B we expect it to decrease very fast at the beginning and then exhibits switching between two solutions as shown in Fig 10. But if the initial condition is as high as 1000 then in terms of initial condition the hopping would be almost negligible as can be seen from Fig 11.

If one re-simulate with the exclusive switch off the fixed points are thrown off and we observe some random pattern.

Fig 10: Genetic Switch with initial population of A and B = 100

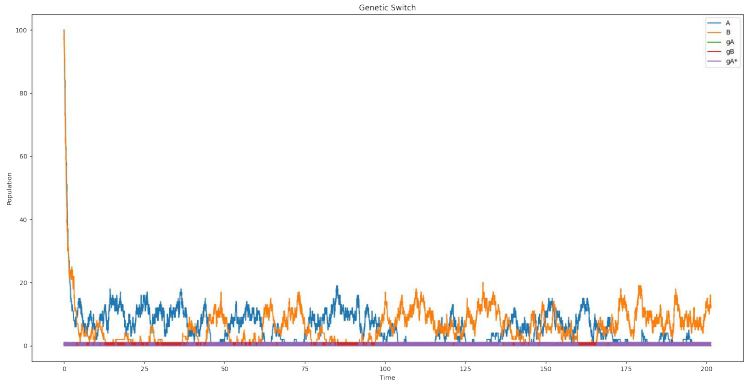

Figure 10: Genetic Switch Large Populations

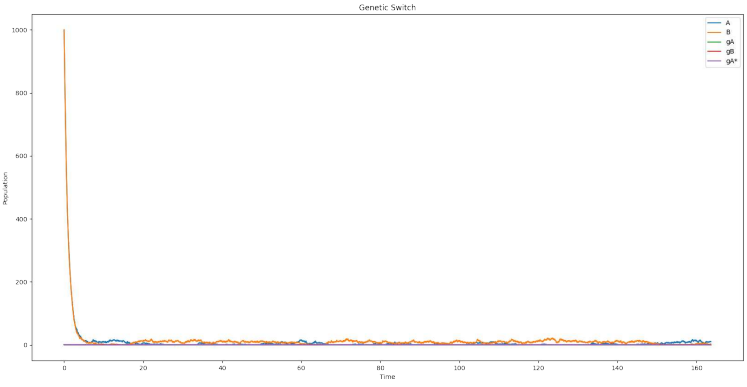

Figure 11: Genetic Switch Large Populations 1000

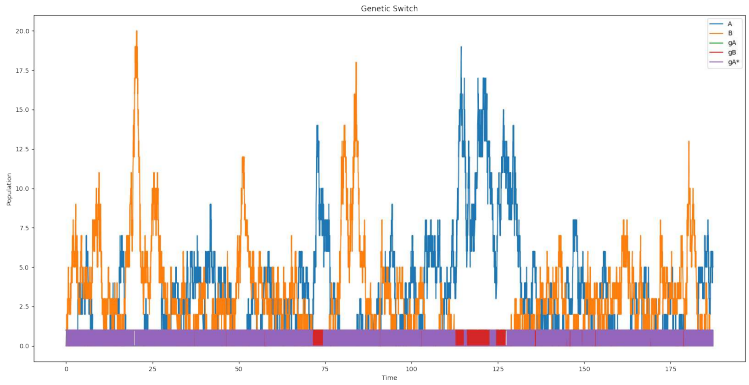

Figure 12: Genetic Switch with Exclusive Switch turned off

```
#Importing required libraries
import numpy as np #Used for mathematical operations
from tqdm import tqdm #Used for progress bar in loops
import matplotlib.pyplot as plt #Plotting software
from time import time; #To obtain system time.
# Initial states.

np.random.seed(int(time())); #Passing time as the random seed to the numpy
# c is the initial populations.
populations = np.array((1000,1000,0,0,0),dtype=int); # Ip = [A1,A2,A1*,E,S]
# rate is the initial rates
rates = np.array((1,1,100,1,100,1),dtype=float); # rates = [k1,k2,k3,k4,k5,kd]

# populations = np.zeros(5,dtype=int) # keeps track of nth population.
propensities = np.zeros(7,dtype=float) #keeps track of nth propensities

steps = 1000; #Number of steps for the simulation

time = np.zeros(steps, dtype=float); # Store the time steps
storeP = np.zeros((steps,5),dtype=int); # Store all the populations at all time step
s in 2D array.
storeP[0] = populations; #Store the initial population at step 0
# Defining propensities
def prop(populations,propensities):
    propensities[0] = populations[0]*rates[0]; # A1 -> E + A1 (k1)
    propensities[1] = populations[1]*rates[1]; # A2 -> S + A2 (k2)
    propensities[2] = populations[3]*populations[0]*rates[2]; # E + A1 -> E + A1* (k
3)
    propensities[3] = populations[2]*rates[3]; # A1* -> E + A1* (k4)
    propensities[4] = populations[4]*populations[0]*rates[4]; # S + A1 -> S + (-A1)
(k5)
    propensities[5] = populations[4]*rates[5]; # S -> (-S) (kd)
    propensities[6] = populations[3]*rates[5]; # E -> (-E) (kd)
    return propensities

def reactions(populations,choice):
    if choice == 0: #For A1 -> E + A1 (k1)
        populations[3] +=1; # Only E increases
    if choice == 1 : # For A2 -> S + A2 (k2)
        populations[4] +=1; # Only S increases
    if choice == 2: # For E + A1 -> E + A1* (k3)
        populations[0] -=1; #A1 decreases
        populations[2] +=1; # A1* increases
        # E stayes the same.
    if choice == 3: #For A1* -> E + A1* (k4)
        populations[3] += 1; #Only E increases
    if choice == 4: # For S + A1 -> S + (-A1) (k5)
        populations[0] -=1; #Only A1 decreases as A1 is inactivated
    if choice == 5: # For S -> (-S) (kd)
        populations[4] -=1; # Only S decreases
    if choice == 6: #For E -> (-E) (kd)
        populations[3] -=1; #Only E decreases
    return populations


for i in tqdm(range(1,steps)): # Loop i will range from 1 to steps-1
    propensities = prop(populations,propensities); #Calculate propensities for all t
he reactions
    sumProp = np.sum(propensities); # sum of all the propensities
    maxTime = 1.0/sumProp; # Maximum time.
    tau = np.random.exponential(maxTime); # Holding time sampled from an exponential
 distribution.
    choice = np.random.choice(7,1,p=propensities/sumProp); # Making choice for the r
eaction to be evaluated
    populations = reactions(populations,choice); # Update the population based on th
e choice made above.
    time[i] = time[i-1]+tau; #Store the time.
    storeP[i]=populations#Store all the populations at step i

#plotting
storeP = storeP.transpose();
#Plotting A1* population as a function of time.
plt.plot(time,storeP[2]);
```

```
plt.title("Gillespie Simulation")
plt.xlabel("Time");
plt.ylabel("Population");
plt.legend("A1*");
plt.savefig("A1*Population-"+str(steps)+"i1000.png",dpi=100)
plt.clf();
#Plotting the histogram of the A1*Population
plt.hist(storeP[2],bins=50);
plt.xlabel("Population");
plt.ylabel("Number");
plt.title("A1* Histogram")
plt.savefig("A1*Histogram-"+str(steps)+"i1000.png",dpi=100)
plt.clf();

#Plotting all the populations
for i in range(5):
    plt.plot(time,storeP[i]);
plt.title("Gillespie Simulation");
plt.xlabel("Time");
plt.ylabel("Population");
plt.legend(["A1","A2","A1*","E","S"])
plt.savefig("All Population-"+str(steps)+"i1000.png",dpi=100)
plt.clf()
```

```
import numpy as np # Importing Numpy which does most of the mathematical operations
import matplotlib.pyplot as plt #Plotting library
from tqdm import tqdm #To obtain a progress bar for the loop.
from time import time; #To obtain the current system time.

np.random.seed(int(time())); #current system time is feeded as the seed for numpy ca
lculation.
steps = 10000 #Number of steps to be iteratted.

populations = np.array((0,0,1,1,0,0),dtype=int); # populations = [nA,nB,gA,gB,gA*,gB
*]
rates = np.array((1,10,20,19),dtype=float) #rates = [kd,kp,kf,kr]

time = np.zeros(steps, dtype=float); # Store the time steps
storeP = np.zeros((steps,6),dtype=int); # Store all the populations at all time step
s in 2D array.
storeP[0] = populations; #Store the initial population at step 0

propensities = np.zeros((8),dtype=float); #Store the propensities


#Defining all the propensities:
def prop(populations,propensities):
    propensities[0]=populations[0]*rates[0]; #A->0 (kd)
    propensities[1]=populations[2]*rates[1]; #gA->gA+A (kp)
    propensities[2]=populations[2]*populations[1]*rates[2]; #gA+B->gA* (kf)
    propensities[3]=populations[4]*rates[3]; #gA*->gA+B (kr)
    propensities[4]=populations[1]*rates[0]; #B->0 (kd)
    propensities[5]=populations[3]*rates[1]; #gB -> gB +B(kp)
    propensities[6]=populations[3]*populations[0]*rates[2] #gB+A->gB* (kf)
    propensities[7]=populations[5]*rates[3]; #gB*->gB+B (kr)
    return propensities

def reaction(populations,choice):
    if choice == 0: #A->0 (kd)
        populations[0]-=1; #Only A is decreasing by 1
    if choice == 1: #gA->gA+A (kp)
        populations[0]+=1; #Only A is increasing by 1
    if choice == 2: #gA+B->gA* (kf)
        populations[1] -=1; #B decreasing
        populations[2] -=1; #gA decreasing
        populations[4] +=1; #gA* increasing
    if choice == 3: #gA*->gA+B (kr)
        populations[4] -=1; #gA* decreasing
        populations[2] +=1; #gA increasing
        populations[1] +=1; #B increasing

    if choice == 4: #B->0 (kd)
        populations[1] -=1 #Only B is decreasing
    if choice == 5:#gB -> gB +B(kp)
        populations[1] +=1 #Only B is increasing
    if choice == 6: #gB+A->gB* (kf)
        populations[3] -=1 #gB decreasing
        populations[0] -=1 #A is decreasing
        populations[5] +=1 #gB* increasing
    if choice == 7: #gB*->gB+B (kr)
        populations[3] +=1 #gB increasing
        populations[0] +=1 #A is increasing
        populations[5] -=1 #gB* is decreasing
    return populations

if __name__ == "__main__": #Just to show that this is the main function of the code
    for i in tqdm(range(1,steps)): # Loop from 1 to step-1.
        propensities = prop(populations,propensities); #Calculate propensities for a
ll the reactions
        if populations[2]+populations[4] >1: #To check that constrain gA + gA* =1 is
 followed.
            print("something wrong ")
        if populations[3]+populations[5]>1: #To check that constrain gB + gB* =1 is
followed.
            print("something wrong")
        if populations[4] ==1: #can't have a gB* when we already have a gA*
            propensities[6]=0 #setting production of gB* to 0
```

```
        if populations[5]==1: #can't have gA* when gB* =1
            propensities[2]=0 #setting production of gA* to 0
        sumProp = np.sum(propensities); # sum of all the propensities
        maxTime = 1.0/sumProp; # Maximum time.
        tau = np.random.exponential(maxTime); # Holding time sampled from an exponen
tial distribution.
        choice = np.random.choice(8,1,p=propensities/sumProp); # Making choice for t
he reaction to be evaluated
        populations = reaction(populations,choice); # Update the population based on
 the choice made above.
        time[i] = time[i-1]+tau; #Store the time.
        storeP[i]=populations#Store all the populations at step i

    storeP = storeP.transpose(); #Transpoing the matrix to easily plot the graph
    plt.figure(figsize=(20,10)) #Setting large figure size
    for i in range(5): #Loop through all the populations
        plt.plot(time,storeP[i]); #Plot them.
    plt.title("Genetic Switch"); #Title of the plot
    plt.xlabel("Time"); #x-lable of the plot
    plt.ylabel("Population"); #y-label of the plot
    plt.legend(["A","B","gA","gB","gA*","gB*"]); #Legend of the plot
    plt.savefig("geneticSwitch2.png",dpi=100); #Saving the figures
    plt.show() #Display the plot in Qt window.
```