Behiye Erdemir
240206013

EE430 – Midterm

The expression of genes Y and Z are as determined according to the logical expressions in my student idetification number is given below.

Y=(NOT)((NOT)X1 OR (NOT)X2), Z=(NOT) ((NOT)Y AND (NOT)X3)

a) The gene transcription network is drawn as seen on the left side of Figure 1.

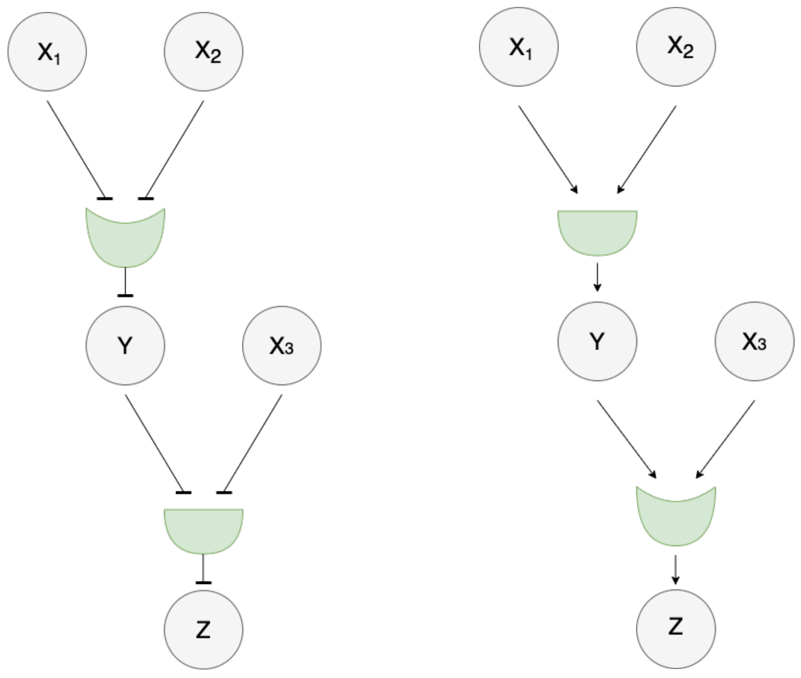


Figure 1: Original and simplified version of the gene transcription network

However, I will continue with a simplified version based on De Morgan's Rule of my network.

The rules are:

- The complement of the union of two sets is the same as the intersection of their complements. That means; NOT (A OR B) = (NOT A) AND (NOT B).
- The complement of the intersection of two sets is the same as the union of their complements. That means; NOT (A AND B) = (NOT A) OR (NOT B).

According to these rules the simplified version of my network is as given in below.

Y= (X1 AND X2) , Z= (Y OR X3)

And the new network is as seen on the right side of Figure 1.

b) The gene Y requires the binding of X_1 and X_2 is expressed with logic approximation as;

$$Y \simeq \beta_Y \,.\, (u([X_1]) - \kappa_{x_1}).\,(u([X_2]) - \kappa_{x_2})$$

The expression becomes without approximation;

$$Y \simeq \beta_Y \,.\left(\frac{([X_1])^{n_{X_1}}(t)}{\left(\kappa_{X_1}\right)^{n_{X_1}} + ([X_1])^{n_{X_1}}(t)}\right).\left(\frac{([X_2])^{n_{X_2}}(t)}{\left(\kappa_{X_2}\right)^{n_{X_2}} + ([X_2])^{n_{X_2}}(t)}\right)$$

The differential equation governing the concentration of gene Y as a function of time is

$$\frac{d([Y])}{dt} \simeq \beta_Y \,.\left(\frac{([X_1])^{n_{X_1}}(t)}{\left(\kappa_{X_1}\right)^{n_{X_1}} + ([X_1])^{n_{X_1}}(t)}\right).\left(\frac{([X_2])^{n_{X_2}}(t)}{\left(\kappa_{X_2}\right)^{n_{X_2}} + ([X_2])^{n_{X_2}}(t)}\right) - \alpha_Y([Y])(t)$$

And the gene Z requires the binding of either Y or X_3 is expressed with logic approximation as;

$$Z \simeq \beta_Z \,.\, \max((u([Y]) - \kappa_Y)\,,(u([X_3]) - \kappa_{x_3}))$$

The expression becomes without approximation;

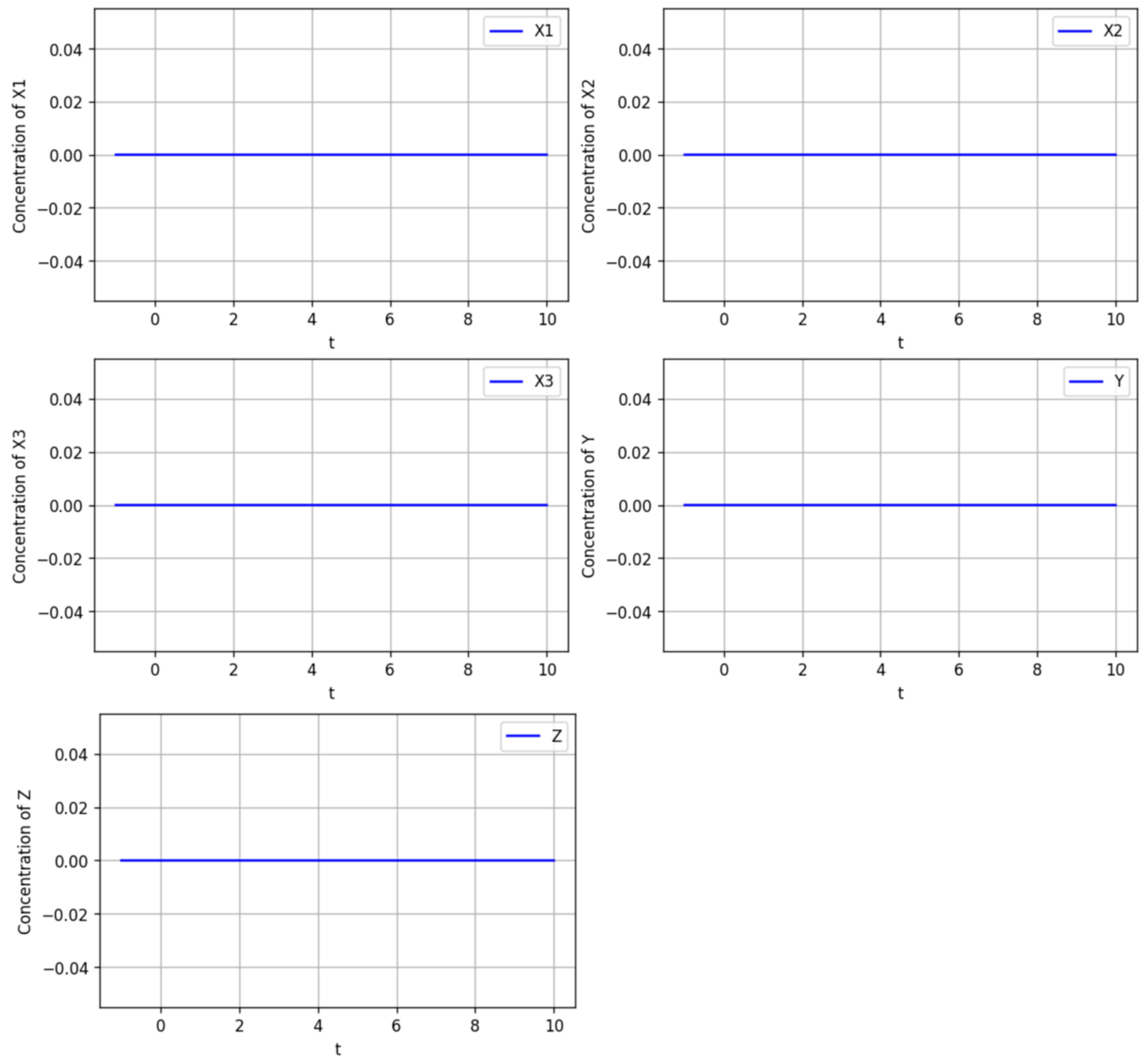
$$Z \simeq \beta_Z \,.\max\left(\left(\frac{([Y])^{n_Y}(t)}{(\kappa_Y)^{n_Y} + ([Y])^{n_Y}(t)}\right).\left(\frac{([X_3])^{n_{X_3}}(t)}{\left(\kappa_{X_3}\right)^{n_{X_3}} + ([X_3])^{n_{X_3}}(t)}\right)\right)$$

The differential equation governing the concentration of gene Z as a function of time is

$$\frac{d([Z])}{dt} \simeq \beta_z \,.\max\left(\left(\frac{([Y])^{n_Y}(t)}{(\kappa_Y)^{n_Y} + ([Y])^{n_Y}(t)}\right).\left(\frac{([X_3])^{n_{X_3}}(t)}{\left(\kappa_{X_3}\right)^{n_{X_3}} + ([X_3])^{n_{X_3}}(t)}\right)\right) - \alpha_z([Z])(t)$$

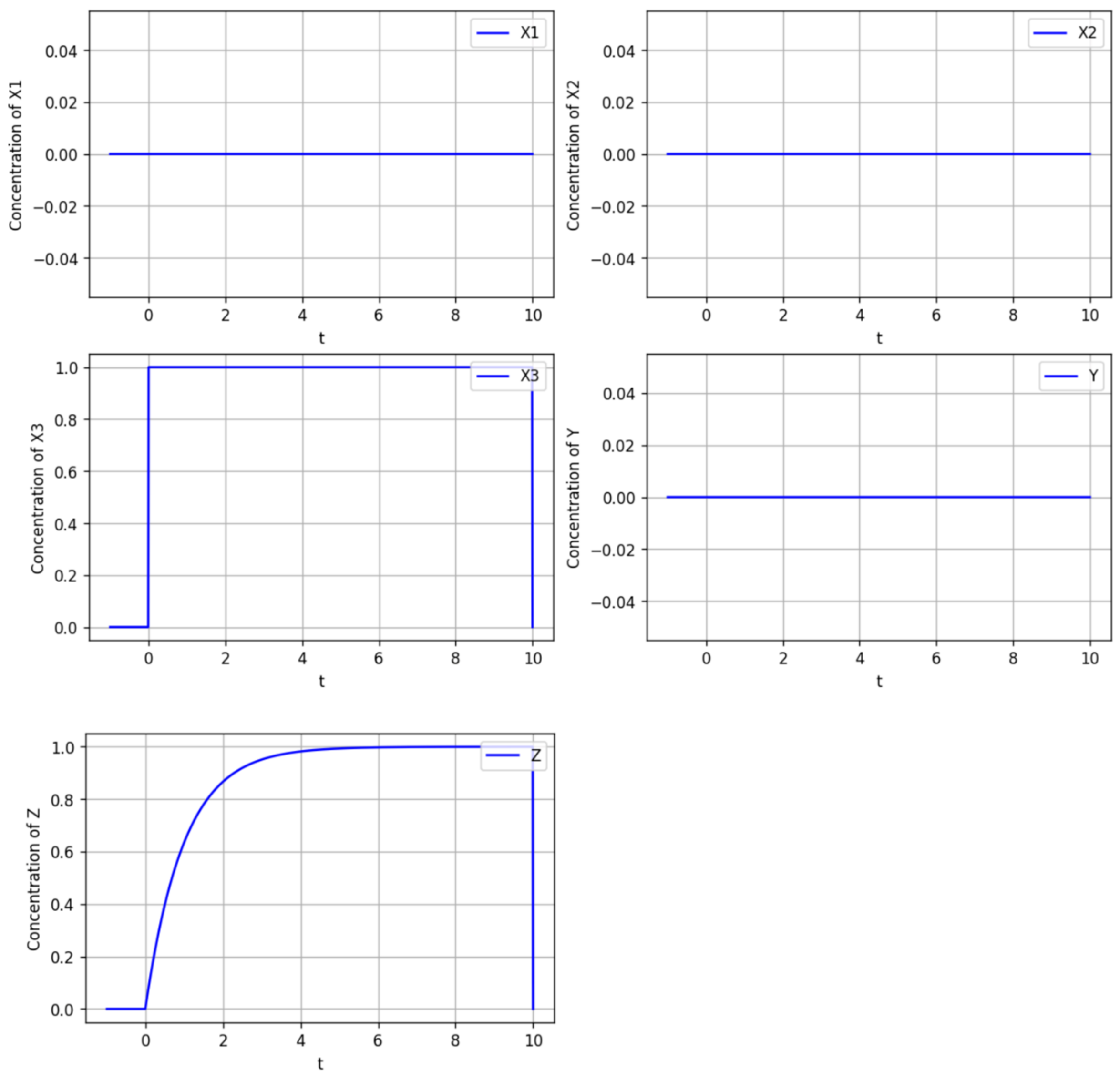
c) The Y and Z concentrations for different combinations of X1, X2, and X3 are as follows. All code is as stated in Appendix I, also available from Google Colab[1].

For X1 = 0, X2 = 0, X3 = 0

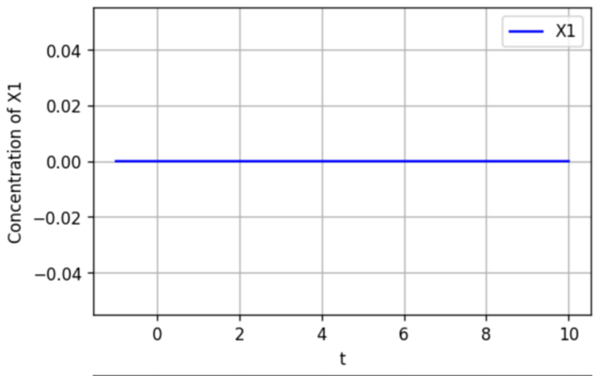


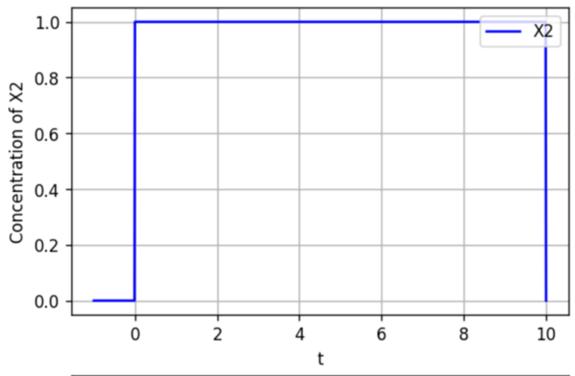
[1] https://colab.research.google.com/drive/1ndEY84J8C2fIv3h8fxVmih7K-ONJstol?usp=sharing

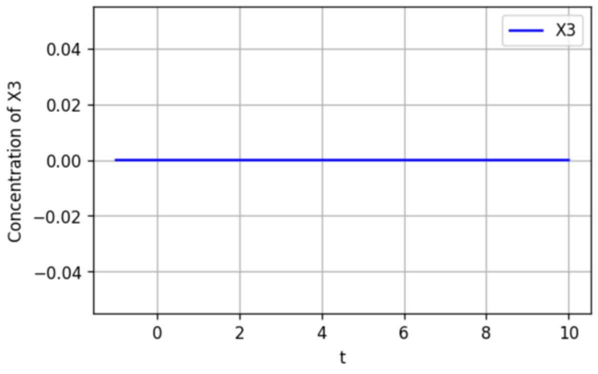
For X1 = 0, X2 = 0, X3 = 1 ;

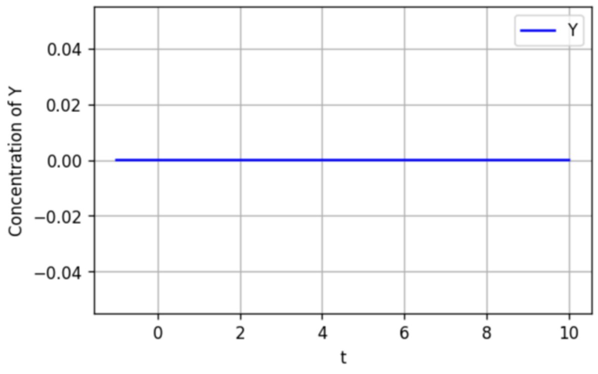


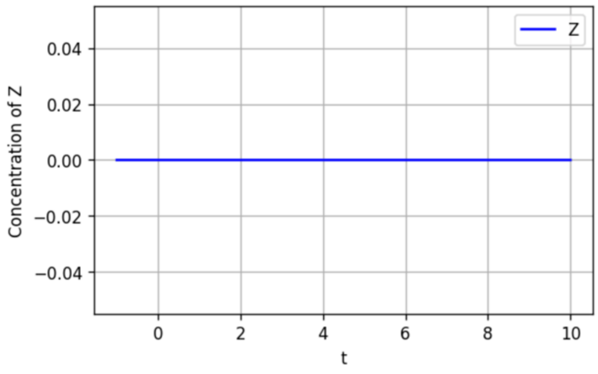
For X1 = 0, X2 = 1, X3 = 0 ;



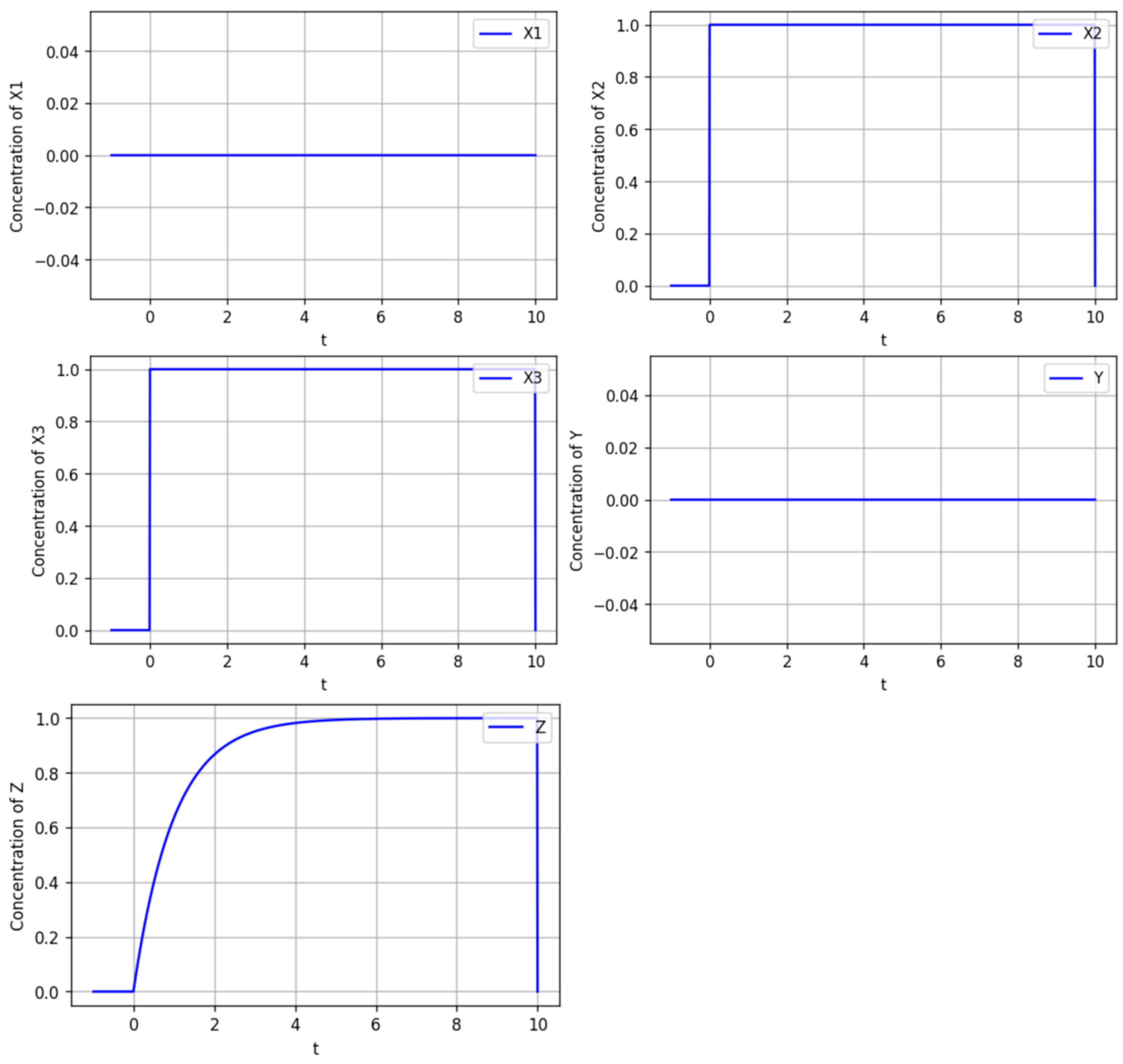




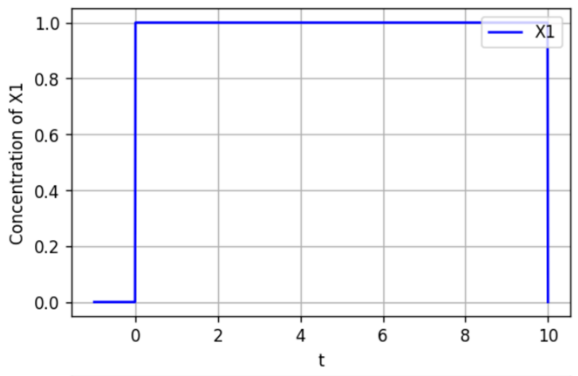


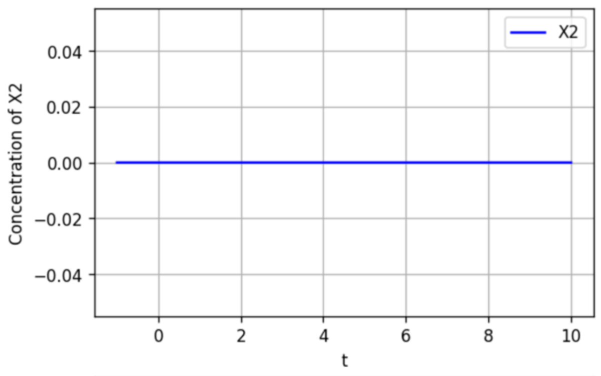


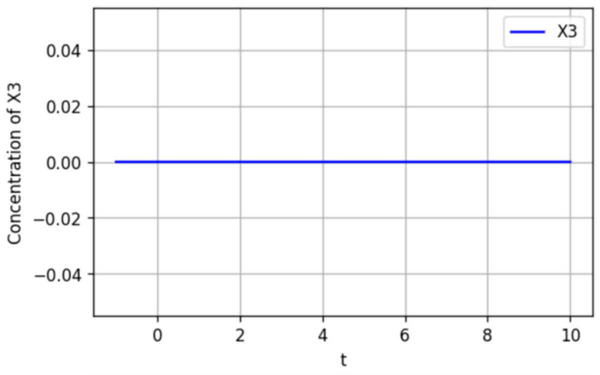
For X1 = 0, X2 = 1, X3 = 1 ;

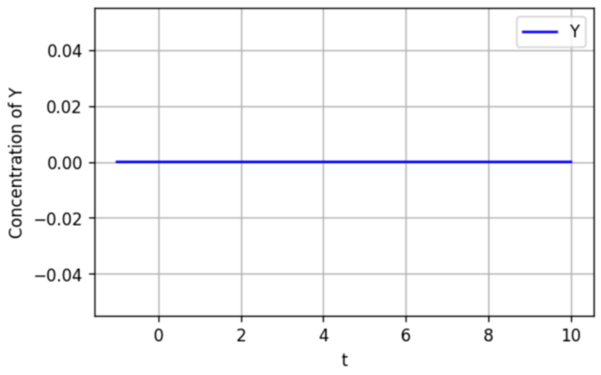


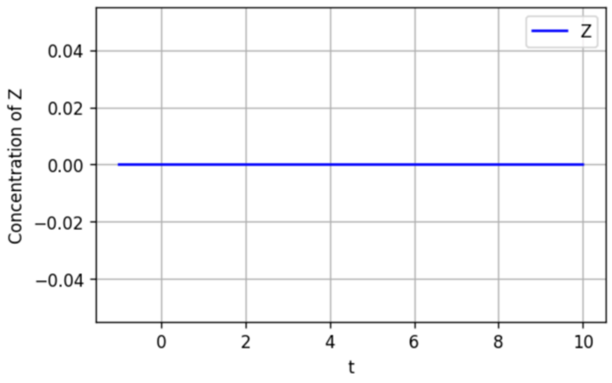
For X1 = 1, X2 = 0, X3 = 0 ;



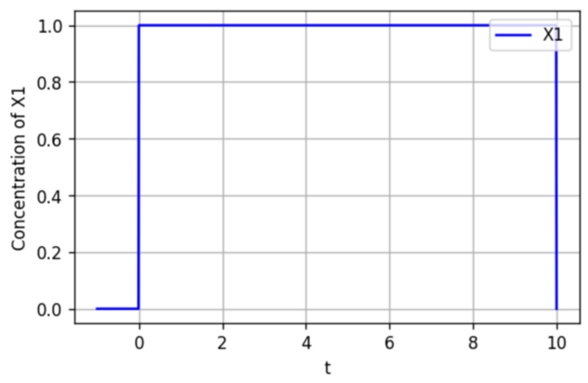


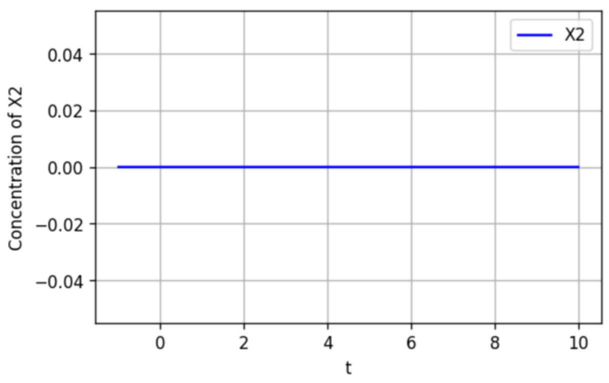


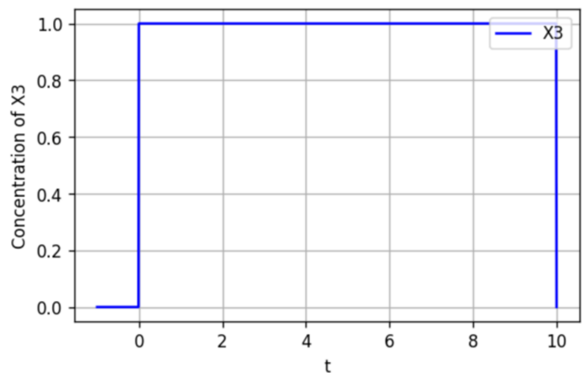


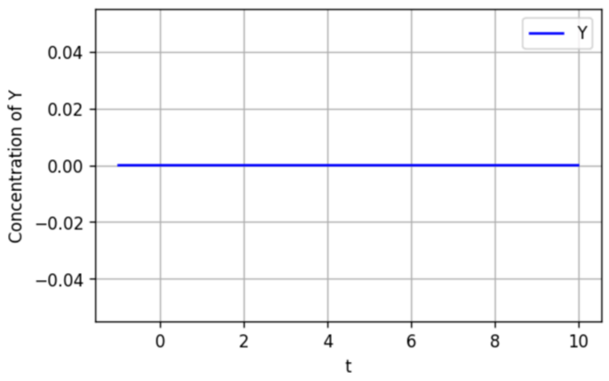


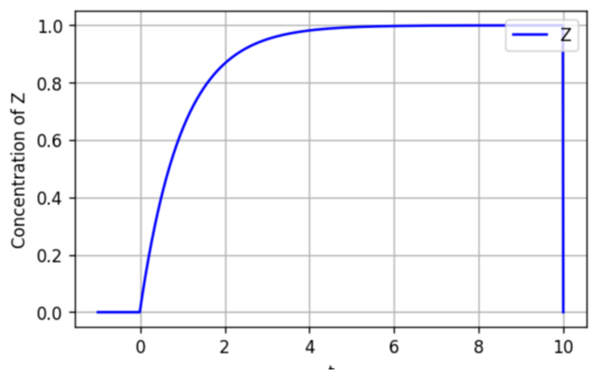
For X1 = 1, X2 = 0, X3 = 1 ;



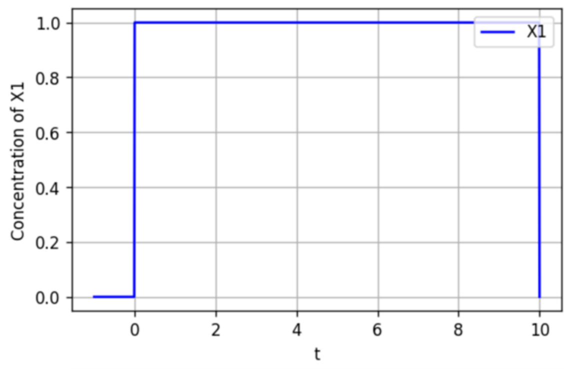


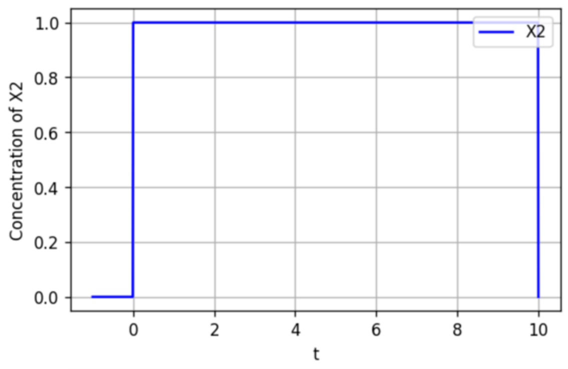


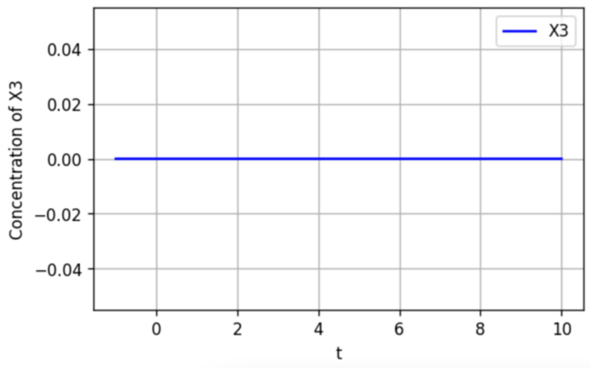


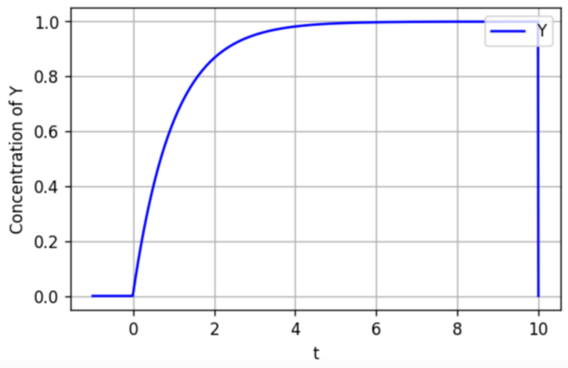


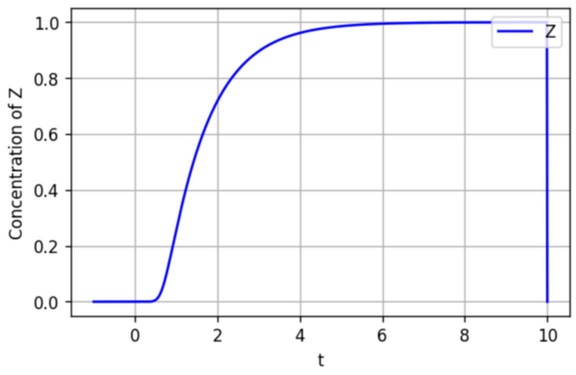
For X1 = 1, X2 = 1, X3 = 0 ;



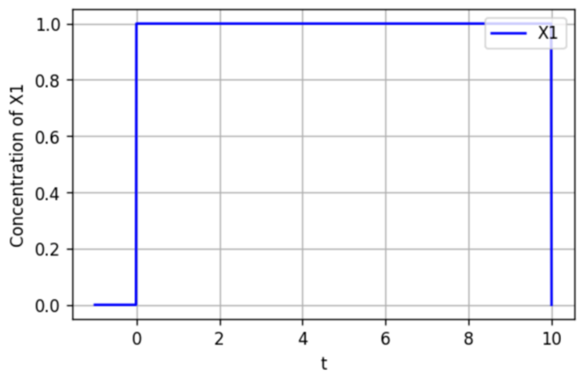


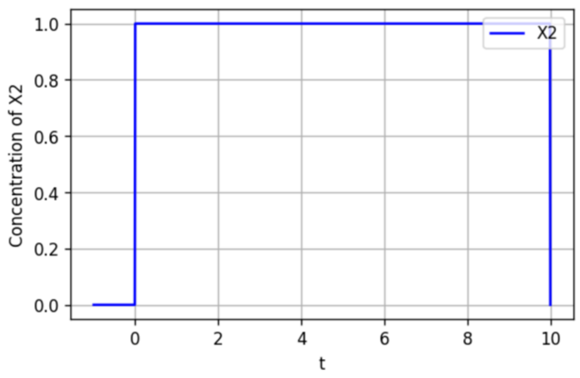


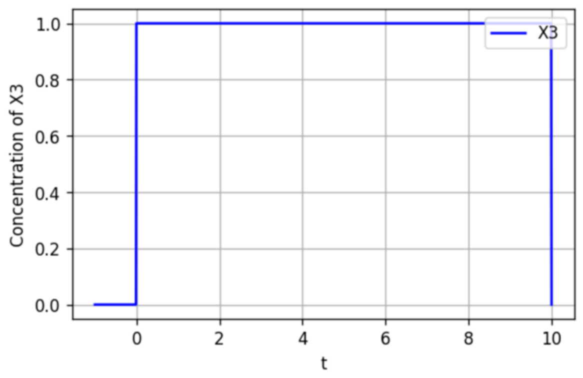


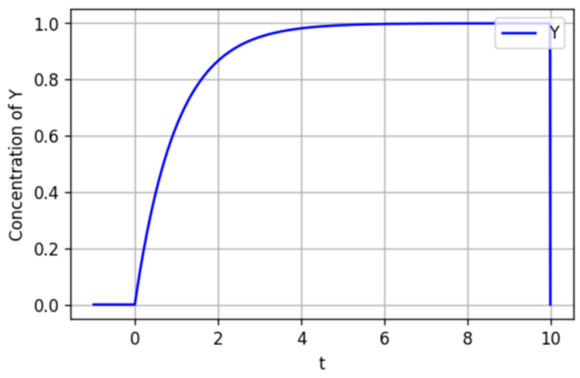


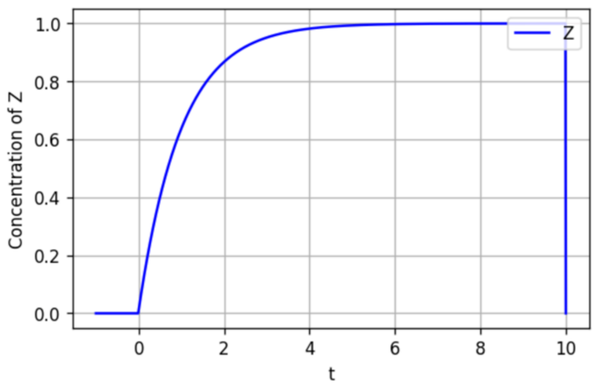
For X1 = 1, X2 = 1, X3 = 1 ;











Appendix I

```
# to calculate the concentration of genes
def conc_act(t0, t_end, dt,betas, alphas,nodes, kappa, n_2gene, t_ext,
t_act,dependency,dep):
    # number of interval according to time variables
    N = int((t_end - t0) / dt + 1)
    # a node matrice, initially zero
    node_matrice = np.zeros((N, len(nodes)))
    # a zero array for initial concentration
    c_nodes = [0] * len(nodes)
    # initialization the array that indicates the change of nodes over time
    dN_dt = [0] * len(nodes)
    # time array
    t_ax = np.arange(t0, t_end, dt)
    print(len(t_ax))
    # calculates the new concentration according to time
    new_n=[0] * (len(nodes))
    # calculates the new concentration according to time
    for index, t in enumerate(t_ax):
      # d to access dependency genes of the gene
      d=0
      for i in range(len(nodes)):
        # genes that has no dependency
        if (dependency[i]==0):
          # change of transcription factor over time
          new_n[i] = betas[i] * ((t >= t_act[i]) & (t <=
t_ext[i])).astype(float)
          # dN_dt[i] = betas[i] * ((t >= t_act[i]) & (t <=
t_ext[i])).astype(float) - alphas[i] * c_nodes[i]

        # for and operation
        elif (dependency[i]==1):
          dN_dt[i]=betas[i] * (c_nodes[dep[d][0]]**n_2gene)/(kappa**n_2gene
+
c_nodes[dep[d][0]]**n_2gene)*((c_nodes[dep[d][1]]**n_2gene)/(kappa**n_2gene
+ c_nodes[dep[d][1]]**n_2gene))  - alphas[i] * c_nodes[i]
          d+=1
          new_n[i] = c_nodes[i] + dN_dt[i] * dt

        # for or operation
        elif (dependency[i]==2):
          dN_dt[i]=betas[i] *
max(((c_nodes[dep[d][0]]**n_2gene)/(kappa**n_2gene +
c_nodes[dep[d][0]]**n_2gene)),(((c_nodes[dep[d][1]]**n_2gene)/(kappa**n_2ge
ne + c_nodes[dep[d][1]]**n_2gene)))  - alphas[i] * c_nodes[i]
          d+=1
          new_n[i] = c_nodes[i] + dN_dt[i] * dt
        # new concentration value for activator
        #new_n[i] = c_nodes[i] + dN_dt[i] * dt
        # assign the new value to the output matrice
        node_matrice[:,i][index] = new_n[i]
        # to calculate cumulatively
        c_nodes[i] = new_n[i]
    #gives neew concentration values as outputs
    return node_matrice


# necessary libraries
import numpy as np
import matplotlib.pyplot as plt
import itertools
import matplotlib as mpl

kappa=0.5       # kappa value
n_2gene=10      # n order

# to define all combinations
l = [0.0, 1.0]
l2=list(itertools.product(l, repeat=3))
print(l2)


nodes = ['X1', 'X2', 'X3', 'Y', 'Z']                              # e.g. Y-
>Z
comb=7                                                             # type of
combination
dependency=[0,0,0,1,2]                                             # 0 for
initial gene, 1 for and operation, 2 for or operation
dep=[[0,1],[2,3]]                                                  # to
determine which genes a gene is linked to
betas = [l2[comb][0], l2[comb][1], l2[comb][2], 1.0, 1.0]          # values
of beta in node order
alphas = [1.0, 1.0, 1.0, 1.0, 1.0 ]                                # Values
of alpha in node order

t0 = -1                     # time before activation
dt = 0.01                   # time interval
t_end=10                    # end time
t_ext=[10,10,10,10,10]      # for extra time interventions
t_act=[0,0,0,0,0]           # activation time

# to calculate the concentrations
outs=
conc_act(t0,t_end,dt,betas,alphas,nodes,kappa,n_2gene,t_ext,t_act,dependenc
y,dep)
```

```python
#for high resolution graphs
mpl.rcParams['figure.dpi'] = 250

# number of interval according to time variables
N = int((t_end - t0) / dt +1) #
# figure size according to number of figures
fig = plt.figure(figsize=(10,len(outs[1])*3))

# loop for creating figures dinamically
for j in range(len(outs[1])):
    # create figures according to number of nodes
    ax = fig.add_subplot(len(outs[1]),1,j+1)
    # plot(t,nodes)
    ax.plot(np.linspace(t0, t_end, N),outs[:,j],'b',label=nodes[j])
    #legend position
    ax.legend(loc="upper right")
    # add x label
    ax.set_xlabel('t')
    # add y label
    ax.set_ylabel('Concentration of {}'.format(nodes[j]))
    # grid on
    ax.grid()
```