Enzyme Kinetics

XIA Ri Xin,

x_rxin@163.com

Application Report for Msc Biomedical Data Science

1. Problem 1 Solution:

Let C_E, C_S, C_{ES} , C_P as the concentration of E, S, ES and P. Based on the loss of mass function, the rate of chages of four species can be written as:

$$C_E' = (k_2+k_3) * C_{ES} - k_1 * C_S * C_E$$
$$C_P' = k_3 * C_{ES}$$
$$C_{ES}' = k_1 * C_E * C_S - (k_2+k_3) * C_{ES}$$
$$C_S' = k_2 * C_{ES} - k_1 * C_S * C_E$$

2. Problem 2 Solution:

the law of mass action

```
import matplotlib.pyplot as plt
# Using the law of mass action
# Get the rate of changes of E,S,ES and P
def dP(E,S,ES):
    return 150*ES #k3=150
def dES(E,S,ES):
    return 100*E*S-750*ES #k1=100; k2+k3=750
def dE(E,S,ES):
    return 750*ES-100*E*S
def dS(E,S,ES):
    return 600*ES-100*E*S #k2=600
```

Fourth Order Runge-Kutta method

```
# the initial concentration
E=[1]
S=[10]
P=[0]
ES=[0]
V=[0] #Velocity

h=0.001
N=1001
ans=[-100,-1]
t=[]
for i in range(1, N):
    temp = h*i
    t.append(temp)
```

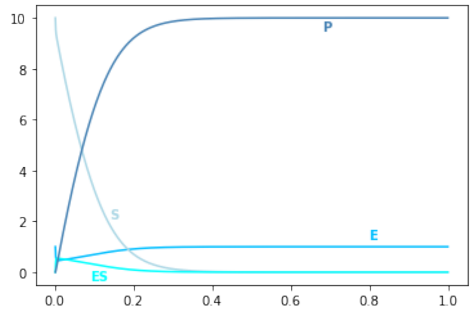
```
def main():
    for i in range(N-2):
        A1 = dE(E[-1], S[-1], ES[-1])
        B1 = dS(E[-1], S[-1], ES[-1])
        C1 = dES(E[-1],S[-1],ES[-1])
        D1 = dP(E[-1], S[-1], ES[-1])

        A2 = dE(E[-1]+h*A1/2, S[-1]+h*B1/2, ES[-1]+h*C1/2)
        B2 = dS(E[-1]+h*A1/2, S[-1]+h*B1/2, ES[-1]+h*C1/2)
        C2 = dES(E[-1]+h*A1/2,S[-1]+h*B1/2, ES[-1]+h*C1/2)
        D2 = dP(E[-1]+h*A1/2, S[-1]+h*B1/2, ES[-1]+h*C1/2)

        A3 = dE(E[-1]+h*A2/2, S[-1]+h*B2/2, ES[-1]+h*C2/2)
        B3 = dS(E[-1]+h*A2/2, S[-1]+h*B2/2, ES[-1]+h*C2/2)
        C3 = dES(E[-1]+h*A2/2,S[-1]+h*B2/2, ES[-1]+h*C2/2)
        D3 = dP(E[-1]+h*A2/2, S[-1]+h*B2/2, ES[-1]+h*C2/2)

        A4 = dE(E[-1]+h*A3, S[-1]+h*B3,ES[-1]+h*C3)
        B4 = dS(E[-1]+h*A3, S[-1]+h*B3,ES[-1]+h*C3)
        C4 = dES(E[-1]+h*A3,S[-1]+h*B3,ES[-1]+h*C3)
        D4 = dP(E[-1]+h*A3, S[-1]+h*B3,ES[-1]+h*C3)

        E.append(E[-1]+h*(A1+2*A2+2*A3+A4)/6)
        S.append(S[-1]+h*(B1+2*B2+2*B3+B4)/6)
        ES.append(ES[-1]+h*(C1+2*C2+2*C3+C4)/6)
        P.append(P[-1]+h*(D1+2*D2+2*D3+D4)/6)
        V.append(dP(E[-1],S[-1],ES[-1]))
        if(V[-1]>ans[0]):
            ans[0]=V[-1]
            ans[1]=S[-1]
main()
print(E[-1],S[-1],ES[-1],P[-1])
```

```
0.9999999432120072 4.1567138490306397e-07 5.678799238022218e-08 9.999999527540618
```

Time-Concentration Plot

```
plt.title("")
figure1 = plt.plot(t, E, color='#00BFFF')
figure2 = plt.plot(t, S, color='#ADD8E6')
figure3 = plt.plot(t, ES, color='#00FFFF')
figure3 = plt.plot(t, P, color='#4682B4')
plt.annotate(text='E',xy=(t[800],E[800]),xytext=(0.8,1.3),weight='bold',color='#00BFFF')
plt.annotate(text='S',xy=(t[150],E[150]),xytext=(0.14,2.1),weight='bold',color='#ADD8E6')
plt.annotate(text='ES',xy=(t[90],E[90]),xytext=(0.09,-0.3),weight='bold',color='#00FFFF')
```

```
plt.annotate(text='P',xy=(t[500],E[500]),xytext=(0.68,9.5),weight='bold',color='#4682B4')
```



3. Problem 3 Solution:

Plot V as a function of the concentration of S

```
print(ans[1], ans[0])
plt.figure()
plt.title("V as a function of the concentration of S")
plt.xlabel('Concentration of S')
plt.ylabel('Velocity')
plt.plot(S,V,linestyle=':',color='b')
plt.annotate(text ='Vm',xy=(ans[1],ans[0]),xytext=(8,60),weight='bold',color='b',arrowprops=dict(arrowstyle='-|>',color='k'))
#plt.scatter(S,t)
plt.show()
```

```
9.083032406486833 82.21916075356128
```