8.1

$\mathtt{r}()$ = the rate of changes
[ ] = concentration

$$
\begin{aligned}
&r(E) = (k_2 + k_3)[ES] - k_1[E][S] \\
&r(S) = k_2[ES] - k_1[E][S] \\
&r(ES) = k_1[E][S] - (k_2 + k_3)[ES] \\
&r(P) = k_3[ES]
\end{aligned} \tag{1}
$$

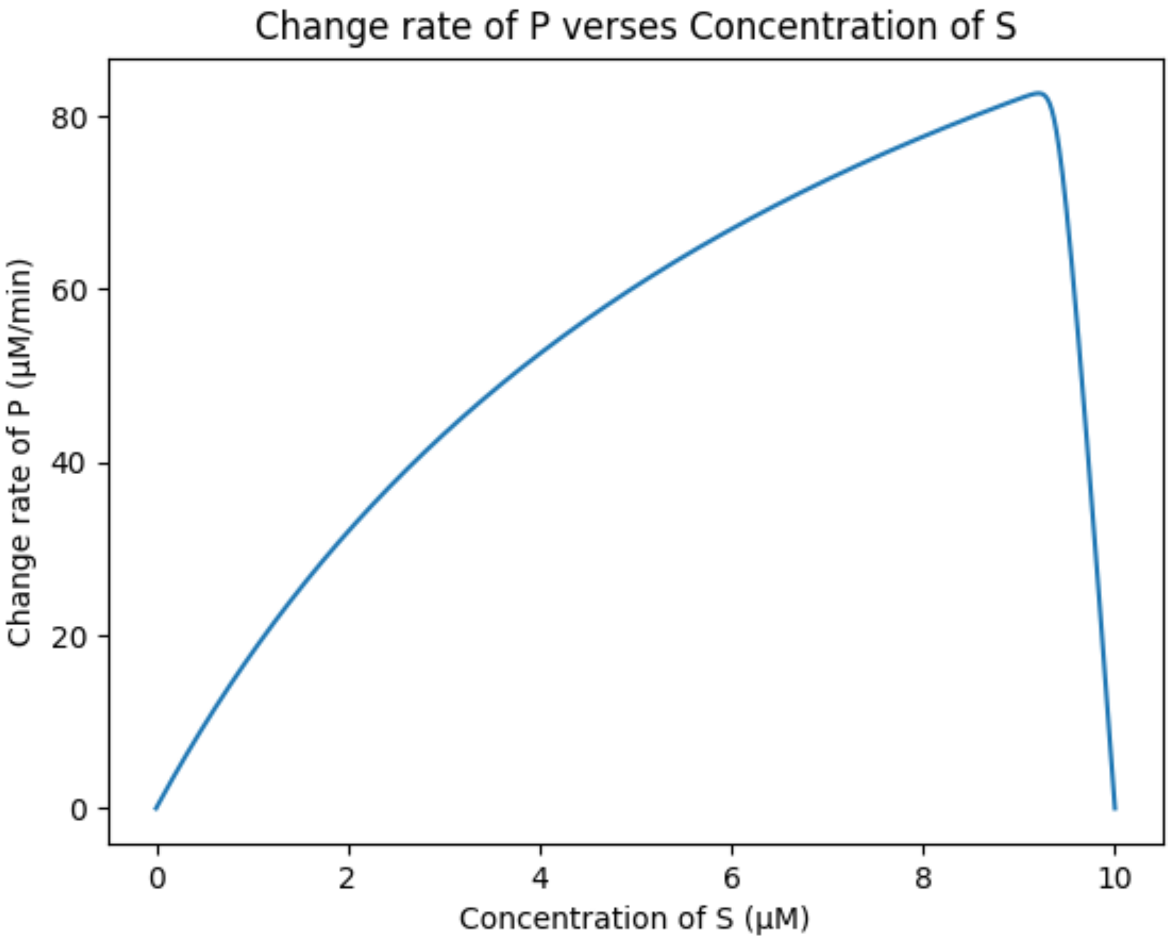
8.2

The final result is:

E: 0.9999999441180305 μM
S: 4.090395657009357e-07 μM
ES: 5.5881969762389094e-08 μM
P: 9.999999535078526 μM

```
After 10000 iterations, set 0.0001 as time step, the results are shown as:
E: 0.9999999441180305 μM
S: 4.090395657009357e-07 μM
ES: 5.5881969762389094e-08 μM
P: 9.999999535078526 μM
```

8.3



When the concentration of S is smaller than 9μM, the velocity V increases approximately linearly. When the concentration of S is around 9μM, however, the velocity V saturates to a maximum value Vm. From this graph, Vm is approximately equal to 80μM/min.

Appendix

```
import numpy as np
import matplotlib.pyplot as plt

#define constant rate
k1 = 100
k2 = 600
k3 = 150

#define four equations
def Ve(E, S, ES):
    df = (k2 + k3) * ES - k1 * E * S
    return df
def Vs(E, S, ES):
    df = k2 * ES - k1 * E * S
    return df
def Ves(E, S, ES):
    df = k1 * E * S - (k2 + k3) * ES
```

```
    return df
def Vp(E, S, ES):
    df = k3 * ES
    return df

def RK4(y1, y2, y3, y4, h, n):
    """
    :param y1: Initial value of y1 :param y2: Initial value of y2 :param y3: Initial value of y3 :param h: time step
    :return: New iterative solution
    """
    E, S, ES, P, vp = [], [], [], [], []
    for i in range(n):
        E.append(y1)
        S.append(y2)
        ES.append(y3)
        P.append(y4)
        K_1 = Ve(E[i], S[i], ES[i])
        L_1 = Vs(E[i], S[i], ES[i])
        M_1 = Ves(E[i], S[i], ES[i])
        N_1 = Vp(E[i], S[i], ES[i])
        K_2 = Ve(E[i] + h / 2 * K_1, S[i] + h / 2 * L_1, ES[i] + h / 2 * M_1)
        L_2 = Vs(E[i] + h / 2 * K_1, S[i] + h / 2 * L_1, ES[i] + h / 2 * M_1)
        M_2 = Ves(E[i] + h / 2 * K_1, S[i] + h / 2 * L_1, ES[i] + h / 2 * M_1)
        N_2 = Vp(E[i] + h / 2 * K_1, S[i] + h / 2 * L_1, ES[i] + h / 2 * M_1)
        K_3 = Ve(E[i] + h / 2 * K_2, S[i] + h / 2 * L_2, ES[i] + h / 2 * M_2)
        L_3 = Vs(E[i] + h / 2 * K_2, S[i] + h / 2 * L_2, ES[i] + h / 2 * M_2)
        M_3 = Ves(E[i] + h / 2 * K_2, S[i] + h / 2 * L_2, ES[i] + h / 2 * M_2)
        N_3 = Vp(E[i] + h / 2 * K_2, S[i] + h / 2 * L_2, ES[i] + h / 2 * M_2)
        K_4 = Ve(E[i] + h * K_3, S[i] + h * L_3, ES[i] + h * M_3)
        L_4 = Vs(E[i] + h * K_3, S[i] + h * L_3, ES[i] + h * M_3)
        M_4 = Ves(E[i] + h * K_3, S[i] + h * L_3, ES[i] + h * M_3)
        N_4 = Vp(E[i] + h * K_3, S[i] + h * L_3, ES[i] + h * M_3)
        y1 = y1 + (K_1 + 2 * K_2 + 2 * K_3 + K_4) * h / 6
        y2 = y2 + (L_1 + 2 * L_2 + 2 * L_3 + L_4) * h / 6
        y3 = y3 + (M_1 + 2 * M_2 + 2 * M_3 + M_4) * h / 6
        y4 = y4 + (N_1 + 2 * N_2 + 2 * N_3 + N_4) * h / 6
        vp.append(Vp(E[i],S[i],ES[i]))
    return E, S, ES, P, vp

def main():
    h = 0.0001
    n = 10000
    E, S, ES, P, vp= RK4(1, 10, 0, 0, h, n)
    print ("After", n, "iterations, set",h,"as time step, the results are shown as:")
    print ("E:",E[n-1],"µM")
    print ("S:",S[n-1],"µM")
    print ("ES:",ES[n-1],"µM")
```

```
    print ("P:", P[n-1],"μM")
    #plot question 8.3
    plt.title("Change rate of P verses Concentration of S")
    plt.xlabel("Concentration of S (μM)")
    plt.ylabel("Change rate of P (μM/min)")
    plt.plot(S, vp)
    plt.show()

if __name__ == '__main__': main()
```