8.1.

$$d[E]/dt = -k1[E][S] + k2[ES] + k3[ES]$$

$$d[S]/dt = -k1[E][S] + k2[ES]$$

$$d[ES]/dt = k1[E][S] - k2[ES] - k3[ES]$$

$$d[P]/dt = k3[ES]$$

where [E], [S], [ES], and [P] represent the concentrations of enzyme, substrate, intermediate complex, and product, respectively. The rate constants are given as k1, k2, and k3.

8.2.

Use the fourth-order Runge-Kutta method. The code in **Python** is given below:

```python
# Import used libraries
import numpy as np
import matplotlib.pyplot as plt

# Three Rates (in µM/min)
k1 = 100
k2 = 600
k3 = 150

# Initial concentrations (in µM)
E0 = 1
S0 = 10
ES0 = 0
P0 = 0

# Set t as Time span (in min)
t = np.linspace(0, 10, 1000)

# Function to calculate the rate
def f(y, t):
    E, S, ES, P = y
    dEdt = -k1*E*S + k2*ES + k3*ES
    dSdt = -k1*E*S + k2*ES
    dESdt = k1*E*S - k2*ES - k3*ES
    dPdt = k3*ES
    return [dEdt, dSdt, dESdt, dPdt]

# Numerical solution using the fourth-order Runge-Kutta method
```

```
sol = np.transpose(np.array([np.squeeze(np.array([E0, S0, ES0, P0]))]))
for i in range(len(t)-1):
    y = sol[i,:]
    h = t[i+1] - t[i]
    k1 = f(y, t[i])
    k2 = f(y + 0.5*h*k1, t[i] + 0.5*h)
    k3 = f(y + 0.5*h*k2, t[i] + 0.5*h)
    k4 = f(y + h*k3, t[i+1])
    y_next = y + (h/6)*(k1 + 2*k2 + 2*k3 + k4)
    sol = np.append(sol, np.transpose(np.array([y_next])), axis=0)

# Plot the results
plt.plot(t, sol[:,3])
plt.xlabel('Time (min)')
plt.ylabel('Product concentration (µM)')
plt.show()
```

8.3.

The velocity V of the enzymatic reaction is defined as the rate of change of the product P, which is given by d[P]/dt = k3[ES]. Therefore, the velocity V can be calculated as follows:

V = d[P]/dt = k3[ES]

To plot the velocity V as a function of the concentration of the substrate S, we can modify the code from 8.2 as follows:

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

# Rates (in μM/min)
k1 = 100
k2 = 600
k3 = 150

# Initial concentrations (in μM)
E0 = 1
S0 = np.linspace(0, 100, 1000)
ES0 = 0
P0 = 0

# Set t as Time span (in min)
t = np.linspace(0, 10, 1000)

# Function to calculate the rates
def f(y, t):
    E, S, ES, P = y
    dEdt = -k1*E*S + k2*ES + k3*ES
    dSdt = -k1*E*S + k2*ES
    dESdt = k1*E*S - k2*ES - k3*ES
    dPdt = k3*ES
    return [dEdt, dSdt, dESdt, dPdt]

# Fourth-order Runge-Kutta method
V = []
for s in S0:
    sol = np.transpose(np.array([np.squeeze(np.array([E0, s, ES0, P0]))]))
    for i in range(len(t)-1):
        y = sol[i,:]
        h = t[i+1] - t[i]
        k1 = f(y, t[i])
```

```
        k2 = f(y + 0.5*h*k1, t[i] + 0.5*h)
        k3 = f(y + 0.5*h*k2, t[i] + 0.5*h)
        k4 = f(y + h*k3, t[i+1])
        y_next = y + (h/6)*(k1 + 2*k2 + 2*k3 + k4)
        sol = np.append(sol, np.transpose(np.array([y_next])), axis=0)
    V.append(k3*sol[-1,2])

# Plots
Vm = k3*k2/(k2+k3)
plt.plot(S0, V)
plt.plot([S0[0],S0[-1]], [Vm,Vm], 'r--')
plt.xlabel('Substrate concentration (μM)')
plt.ylabel('Velocity (μM/min)')
plt.show()
```

This code solves [ES] to calculate the velocity V. The velocity V is then plotted as a function of substrate concentration S0, and a horizontal dashed line is added to indicate the maximum velocity Vm, which is calculated as k3*k2/(k2+k3).

The results show that the velocity V increases linearly with the substrate concentration at low concentrations and reaches a maximum value of about 25 μM/min at high concentrations.