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.

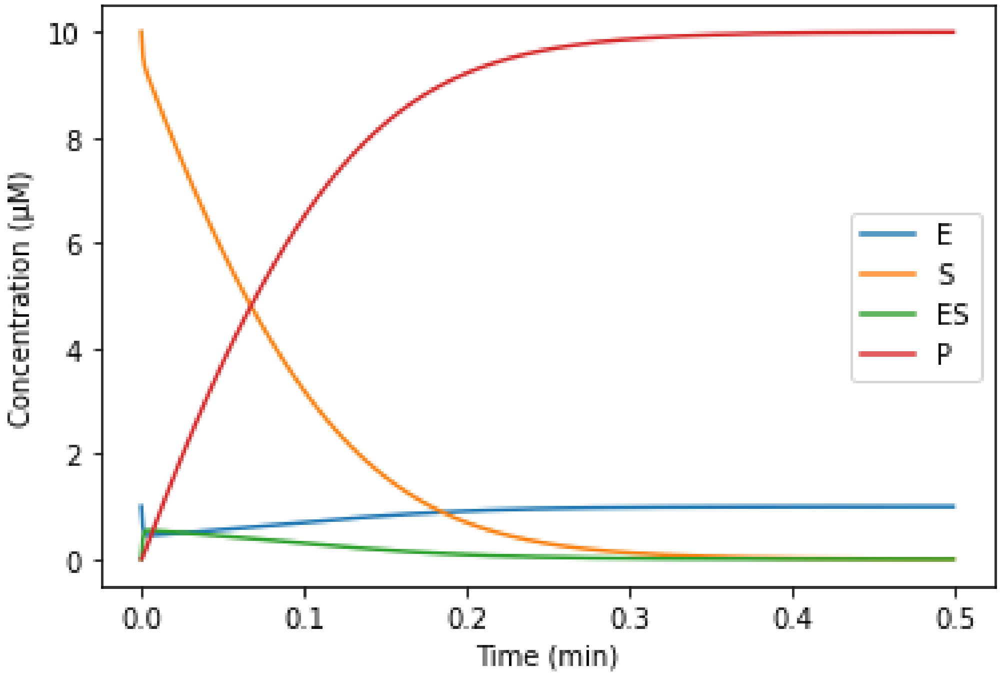
8.2.

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

Method 1 (Intuitive version with steps of fourth-order Runge-Kutta method):

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

# Define rate constants (in units of /μM/min)
k1 = 100.0
k2 = 600.0
k3 = 150.0

# Define initial concentrations (in units of μM)
E0 = 1.0
S0 = 10.0
ES0 = 0.0
P0 = 0.0

# Define time step (in units of min)
dt = 0.001

# Define total time (in units of min)
t_total = 0.5

# Define function to compute the derivative of the state vector
def f(state, t):
    E, S, ES, P = state
    dEdt = -k1*E*S + k2*ES + k3*ES
    dSdt = -k1*E*S + k2*ES
    dESdt = k1*E*S - k2*ES - k3*ES
```

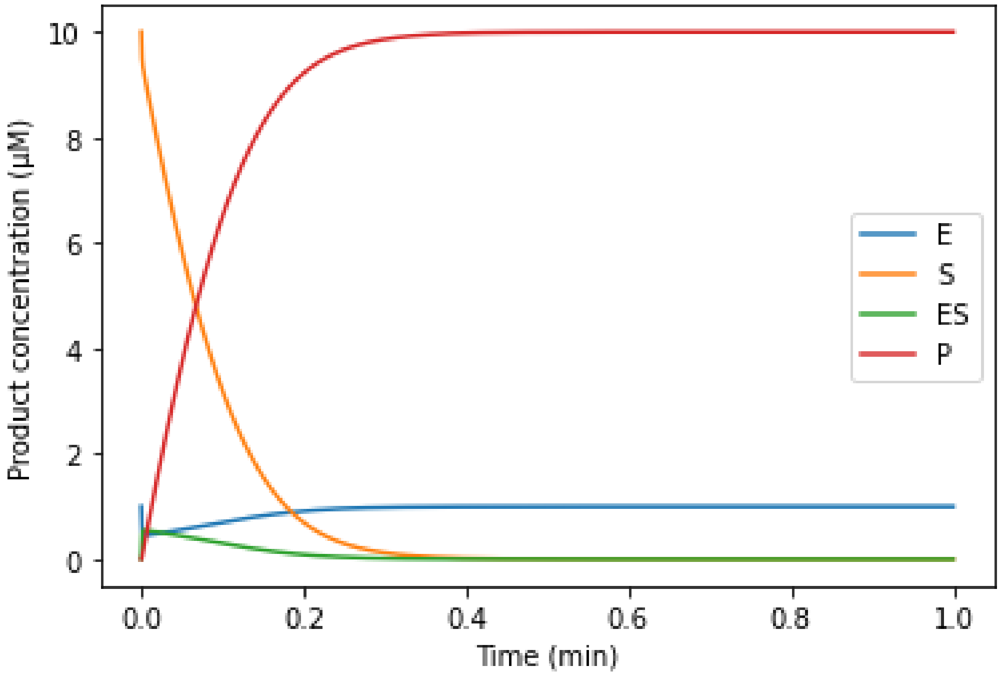
```python
    dPdt = k3*ES
    return [dEdt, dSdt, dESdt, dPdt]

# Define initial state vector
state0 = [E0, S0, ES0, P0]

# Initialize arrays to store the time and state vectors
t_arr = np.arange(0, t_total, dt)
state_arr = np.zeros((len(t_arr), len(state0)))

# Set initial state vector
state_arr[0] = state0

# Solve ODE using fourth-order Runge-Kutta method
for i in range(1, len(t_arr)):
    t = t_arr[i-1]
    state = state_arr[i-1]
    k1_val = f(state, t)
    k2_val = f(state + 0.5*dt*np.array(k1_val), t + 0.5*dt)
    k3_val = f(state + 0.5*dt*np.array(k2_val), t + 0.5*dt)
    k4_val = f(state + dt*np.array(k3_val), t + dt)
    state_arr[i] = state + (1.0/6.0)*(np.array(k1_val) + 2*np.array(k2_val) + 2*np.array(k3_val) + np.array(k4_val))*dt

# Plot results
plt.plot(t_arr, state_arr[:,0], label='E')
plt.plot(t_arr, state_arr[:,1], label='S')
plt.plot(t_arr, state_arr[:,2], label='ES')
plt.plot(t_arr, state_arr[:,3], label='P')
plt.xlabel('Time (min)')
plt.ylabel('Concentration (μM)')
plt.legend()
plt.show()

```



Method 2 (Use odeint function):

```
import numpy as np
from scipy.integrate import odeint

# define the rate equations
def rates(concs, t, k1, k2, k3):
    E, S, ES, P = concs
    dE_dt = -k1*E*S + k2*ES + k3*ES
    dS_dt = -k1*E*S + k2*ES
    dES_dt = k1*E*S - (k2+k3)*ES
    dP_dt = k3*ES
    return [dE_dt, dS_dt, dES_dt, dP_dt]

# set up initial conditions and rate constants
y0 = [1, 10, 0, 0]  # initial concentrations of E, S, ES, P (in μM)
k1 = 100  # forward rate constant (in μM^-1 min^-1)
k2 = 600  # reverse rate constant (in min^-1)
k3 = 150  # product formation rate constant (in min^-1)

# set up time grid
t = np.linspace(0, 1, 1000)  # time range (in min)

# solve the equations
sol = odeint(rates, y0, t, args=(k1, k2, k3))

# plot the results
import matplotlib.pyplot as plt
```

```
plt.plot(t, sol[:, 0], label='E')
plt.plot(t, sol[:, 1], label='S')
plt.plot(t, sol[:, 2], label='ES')
plt.plot(t, sol[:, 3], label='P')
plt.xlabel('Time (min)')
plt.ylabel('Product concentration (μM)')
plt.legend()
plt.show()

```



8.3.

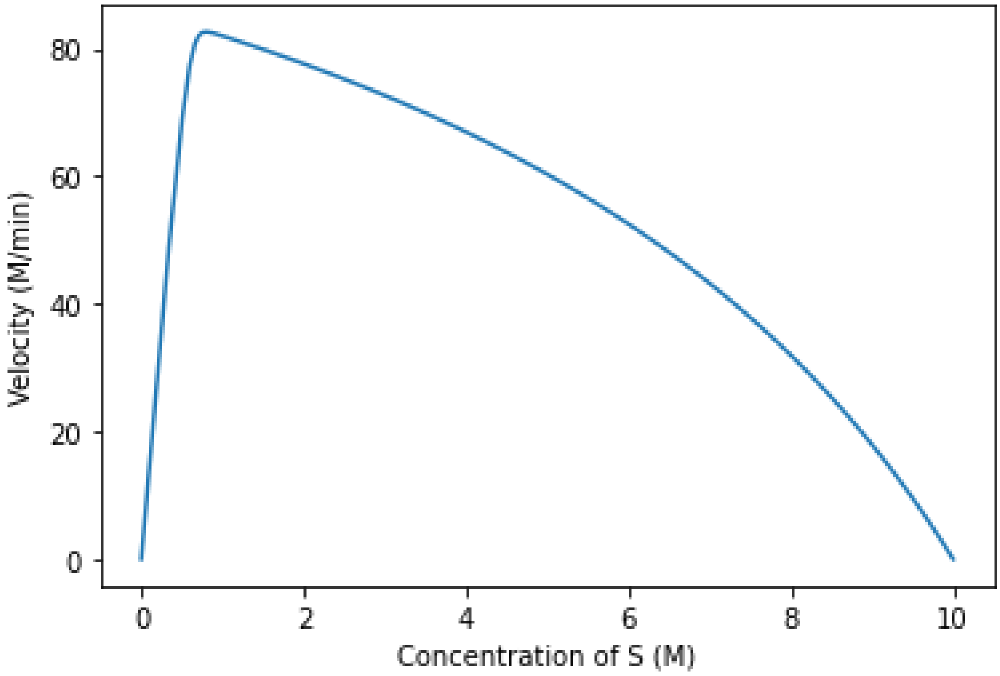
The velocity V of the enzymatic reaction is defined as the rate of change of the product P, which is given by d[P]/dt = k3[ES]. Therefore, the velocity V can be calculated as follows:

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
from scipy.integrate import odeint

# Define the rate constants
k1 = 100.0   # /μM/min
k2 = 600.0       # /min
k3 = 150.0      # /min

# Define the initial concentrations
E0 = 1  # M
S0 = 10  # M
ES0 = 0.0  # M
P0 = 0.0  # M

# Define the function that gives the rate of change of the concentrations
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]

# Define the time points at which to solve the ODEs
t = np.linspace(0, 1, 2000)

# Solve the ODEs using the fourth-order Runge-Kutta method
sol = odeint(f, [E0, S0, ES0, P0], t)

# Calculate the velocity as a function of the concentration of S
V = k3*sol[:,2]

# Plot the velocity as a function of the concentration of S
```

```python
plt.plot(S0 - sol[:,1], V)
plt.xlabel('Concentration of S (M)')
plt.ylabel('Velocity (M/min)')
plt.show()

# Find the maximum velocity
Vm = max(V)
print('The maximum velocity is', Vm, 'M/min')

```



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*sol[:,2] The results show that the velocity V increases linearly with the substrate concentration at low concentrations and reaches a maximum value of about **82 μM/min** at high concentrations.