Enzyme Kinetics

(1) $E + S \underset{k_2}{\overset{k_1}{\rightleftarrows}} ES \xrightarrow{k_3} E + P$

The rate of change of free enzyme E is given by:

$$\frac{d(E)}{dt} = -k_1[E][S] + k_2[ES] + k_3[ES]$$

The rate of change of substrate S is given by:

$$\frac{d[S]}{dt} = -k_1[E][S] + k_2[ES]$$

The rate of change of enzyme-substrate complex ES is given by:

$$\frac{d[ES]}{dt} = k_1[E][S] - (k_2 + k_3)[ES]$$

The rate of change of product P is given by:

$$\frac{d[P]}{dt} = k_3[ES]$$

(2) The code is as follows:

```
(1)import numpy as np
(2)import matplotlib.pyplot as plt
(3)
(4)# Define the rate constants
(5)k1 = 100  # 1/min/μM
(6)k2 = 600  # 1/min
(7)k3 = 150  # 1/min
(8)
(9)# Define the initial concentrations
(10)  E0 = 1    # μM
(11)  S0 = 10   # μM
(12)  ES0 = 0
(13)  P0 = 0
(14)
(15)  # Define the time step and simulation duration
(16)  dt = 0.001   # min
```

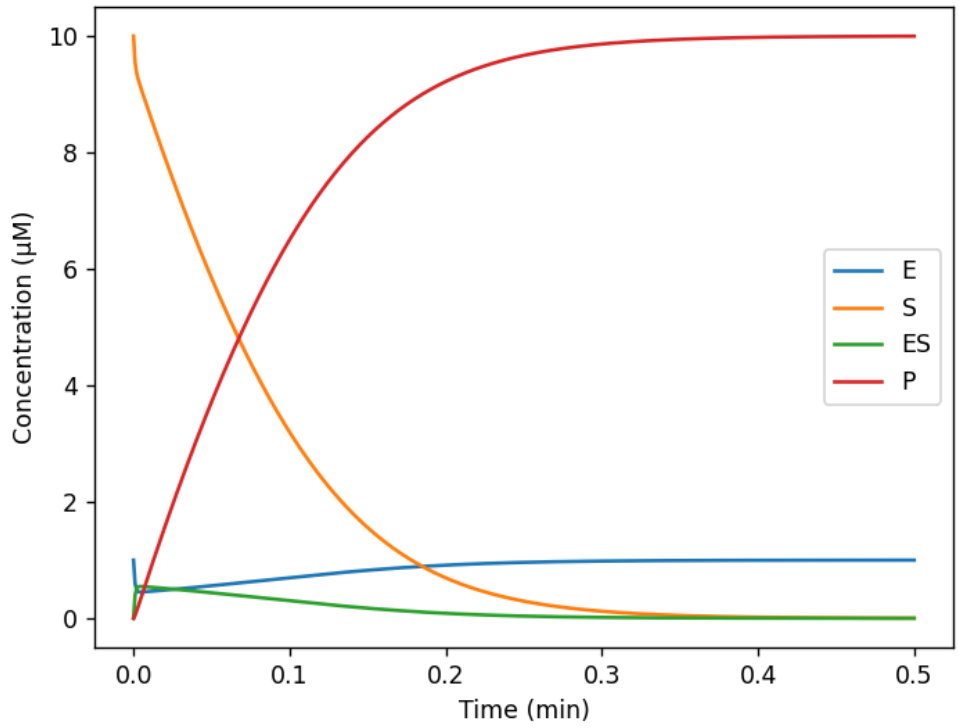
```
(17)  t_max = 0.5   # min
(18)
(19)  # Define the initial conditions
(20)  y0 = np.array([E0, S0, ES0, P0])
(21)
(22)  # Define the function
(23)
(24)
(25)  def f(t, y):
(26)      E, S, ES, P = y
(27)      dEdt = -k1*E*S + k2*ES + k3*ES
(28)      dSdt = -k1*E*S + k2*ES
(29)      dESdt = k1*E*S - k2*ES - k3*ES
(30)      dPdt = k3*ES
(31)      return np.array([dEdt, dSdt, dESdt, dPdt])
(32)
(33)  # Define the function for the fourth-order Runge-Kutta method
(34)  # traditional classical runge-kutta
(35)
(36)
(37)  def rk4_step(f, t, y, dt):
(38)      k1 = f(t, y)
(39)      k2 = f(t + dt/2, y + k1*dt/2)
(40)      k3 = f(t + dt/2, y + k2*dt/2)
(41)      k4 = f(t + dt, y + k3*dt)
(42)      return y + (k1 + 2*k2 + 2*k3 + k4)*dt/6
(43)
(44)
(45)  # Perform the simulation
(46)  t_vals = np.arange(0, t_max+dt, dt)
(47)  y_vals = np.zeros((len(t_vals), len(y0)))
(48)  y_vals[0, :] = y0
(49)  for i in range(len(t_vals)-1):
(50)      y_vals[i+1, :] = rk4_step(f, t_vals[i], y_vals[i, :], dt)
(51)
(52)  # Plot the results
(53)  plt.figure('Runge Kutta numerical results')
(54)  plt.plot(t_vals, y_vals[:, 0], label='E')
(55)  plt.plot(t_vals, y_vals[:, 1], label='S')
(56)  plt.plot(t_vals, y_vals[:, 2], label='ES')
(57)  plt.plot(t_vals, y_vals[:, 3], label='P')
(58)  plt.xlabel('Time (min)')
(59)  plt.ylabel('Concentration (μM)')
(60)  plt.legend()
```

```
(61)  plt.show()
```

The calculation result as follows:

	T=0 (μM)	T=0.05 (μM)	T=0.1 (μM)	T=0.15 (μM)	T=0.2 (μM)	T=0.25 (μM)	T=0.3 (μM)	T=0.35 (μM)	T=0.4 (μM)
E	1	0.56	0.69	0.83	0.91	0.96	0.98	0.99	0.99
S	10	5.84	3.19	1.56	0.69	0.29	0.12	0.04	0.02
ES	0	0.43	0.30	0.17	0.08	0.04	0.01	0.006	0.002
P	0	3.72	6.50	8.27	9.22	9.67	9.86	9.94	9.98

The Runge Kutta numerical results figure as follows:



(3) . According to the Michaelies-Menten equation, the relationship between the velocity V and the concentration of substrate S can be described as follows:

$$V = \frac{v_m[s]}{k_m + [s]} \quad [1]$$

where v_m is the maximum velocity of the reaction, [S] is the concentration of the substrate, and k_m is the Michaelis-Menten constant which represents the concentration of substrate required to achieve half of the maximum velocity.

From the formula [1], when the concentration of S is low, the formula can be simplified to:

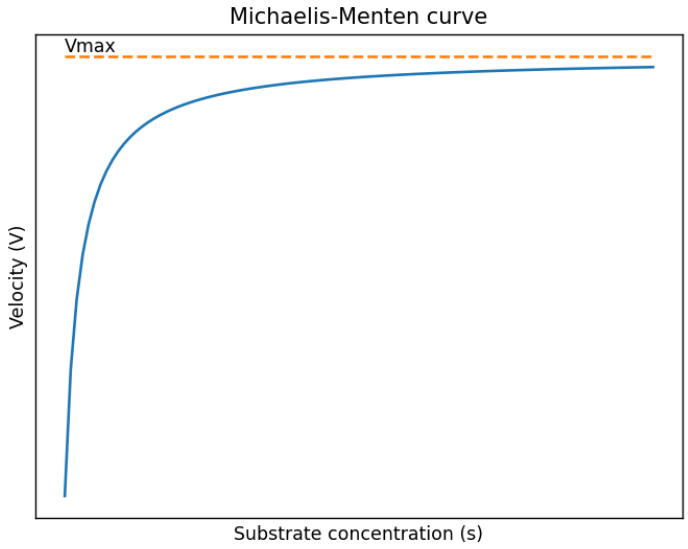
$$V \approx \frac{v_m}{k_m}[s] \quad \textbf{[2]}$$

This relationship is approximately linear, meaning that the velocity increases proportionally with the concentration of the substrate.

However, as the concentration of S increases, the term [S] becomes the denominator of the formula [1] and the velocity V saturates to the maximum value v_m. the formula can be simplified to:

$$V \approx v_m \quad \textbf{[3]}$$

The figure is as follows:


Michaelis-Menten curve
Vmax
Velocity (V)
Substrate concentration (s)