1.

[E], [S], [ES], [P] are the concentration of E, S, ES, P

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

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

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

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

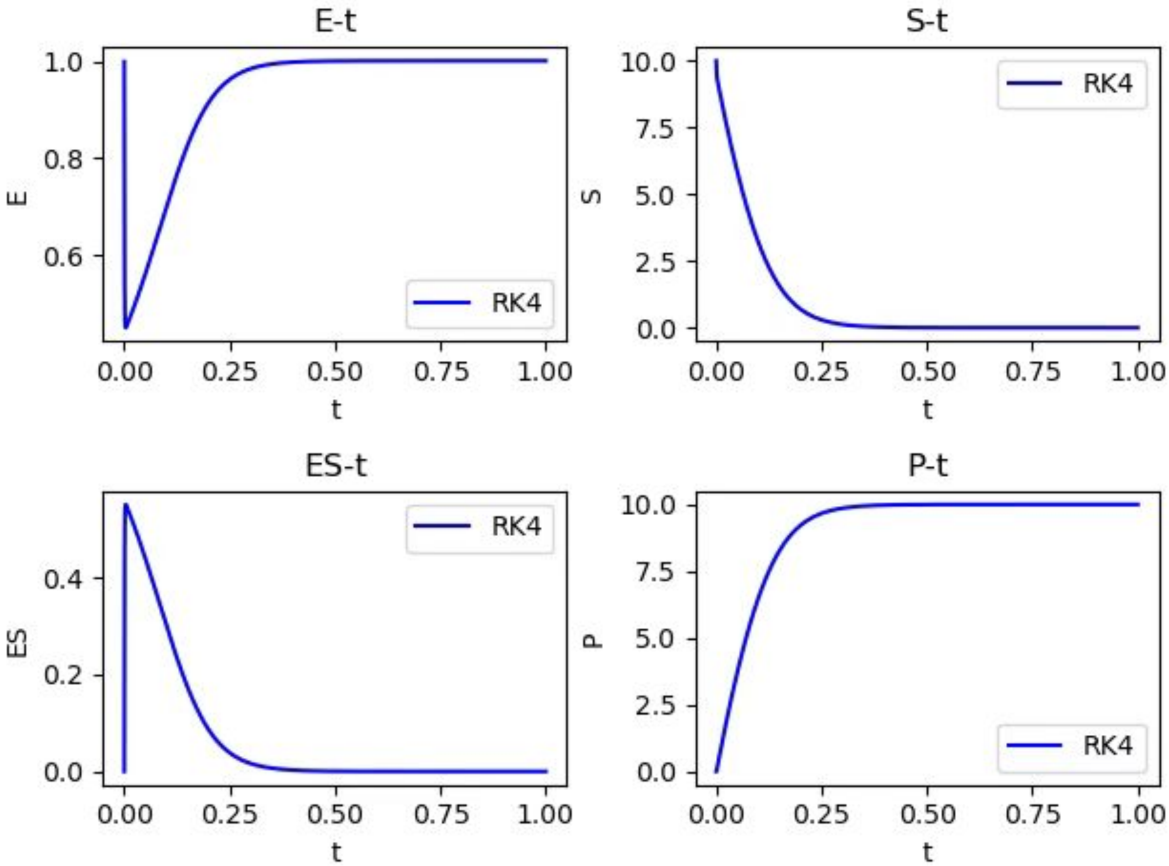
2.

Code:

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

def func_E(E, S, ES, P, k1, k2, k3):
    return -E*S*k1+ES*k2+ES*k3
def func_S(E, S, ES, P, k1, k2, k3):
    return -E*S*k1+ES*k2
def func_ES(E, S, ES, P, k1, k2, k3):
    return E*S*k1-ES*k2-ES*k3
def func_P(E, S, ES, P, k1, k2, k3):
    return ES*k3


### RK4
k1 = 100
k2 = 600
k3 = 150
t_ini = 0
t_end = 1
t_h = 1e-5                    # in minutes

t = np.linspace(t_ini, t_end, int((t_end-t_ini)/t_h+1))
E = t.copy()
S = t.copy()
ES = t.copy()
P = t.copy()
E[0] = 1
S[0] = 10
ES[0] = 0
P[0] = 0
for i in range(t.shape[0]-1):
    h_i = t[i+1] - t[i]

    k1_E = func_E(E[i], S[i], ES[i], P[i], k1, k2, k3)
```

```python
    k1_S = func_S(E[i], S[i], ES[i], P[i], k1, k2, k3)
    k1_ES = func_ES(E[i], S[i], ES[i], P[i], k1, k2, k3)
    k1_P = func_P(E[i], S[i], ES[i], P[i], k1, k2, k3)

    k2_E = func_E(E[i]+h_i/2.0*k1_E, S[i], ES[i], P[i], k1, k2, k3)
    k2_S = func_S(E[i], S[i]+h_i/2.0*k1_S, ES[i], P[i], k1, k2, k3)
    k2_ES = func_ES(E[i], S[i], ES[i]+h_i/2.0*k1_ES, P[i], k1, k2, k3)
    k2_P = func_P(E[i], S[i], ES[i], P[i]+h_i/2.0*k1_P, k1, k2, k3)

    k3_E = func_E(E[i]+h_i/2.0*k2_E, S[i], ES[i], P[i], k1, k2, k3)
    k3_S = func_S(E[i], S[i]+h_i/2.0*k2_S, ES[i], P[i], k1, k2, k3)
    k3_ES = func_ES(E[i], S[i], ES[i]+h_i/2.0*k2_ES, P[i], k1, k2, k3)
    k3_P = func_P(E[i], S[i], ES[i], P[i]+h_i/2.0*k2_P, k1, k2, k3)

    k4_E = func_E(E[i]+h_i*k3_E, S[i], ES[i], P[i], k1, k2, k3)
    k4_S = func_S(E[i], S[i]+h_i*k3_S, ES[i], P[i], k1, k2, k3)
    k4_ES = func_ES(E[i], S[i], ES[i]+h_i*k3_ES, P[i], k1, k2, k3)
    k4_P = func_P(E[i], S[i], ES[i], P[i]+h_i*k3_P, k1, k2, k3)

    E[i+1] = E[i] + h_i/6.0*(k1_E+2.0*k2_E+2.0*k3_E+k4_E)
    S[i+1] = S[i] + h_i/6.0*(k1_S+2.0*k2_S+2.0*k3_S+k4_S)
    ES[i+1] = ES[i] + h_i/6.0*(k1_ES+2.0*k2_ES+2.0*k3_ES+k4_ES)
    P[i+1] = P[i] + h_i/6.0*(k1_P+2.0*k2_P+2.0*k3_P+k4_P)


### Plot
plt.subplot(2, 2, 1)
plt.plot(t, E, 'b', label='RK4')
plt.legend()
plt.xlabel('t')
plt.ylabel('E')
plt.title('E-t')

plt.subplot(2, 2, 2)
plt.plot(t, S, 'b', label='RK4')
plt.legend()
plt.xlabel('t')
plt.ylabel('S')
plt.title('S-t')

plt.subplot(2, 2, 3)
plt.plot(t, ES, 'b', label='RK4')
plt.legend()
plt.xlabel('t')
plt.ylabel('ES')
plt.title('ES-t')

plt.subplot(2, 2, 4)
plt.plot(t, P, 'b', label='RK4')
plt.legend()
plt.xlabel('t')
plt.ylabel('P')
plt.title('P-t')

plt.show()
plt.close()
```

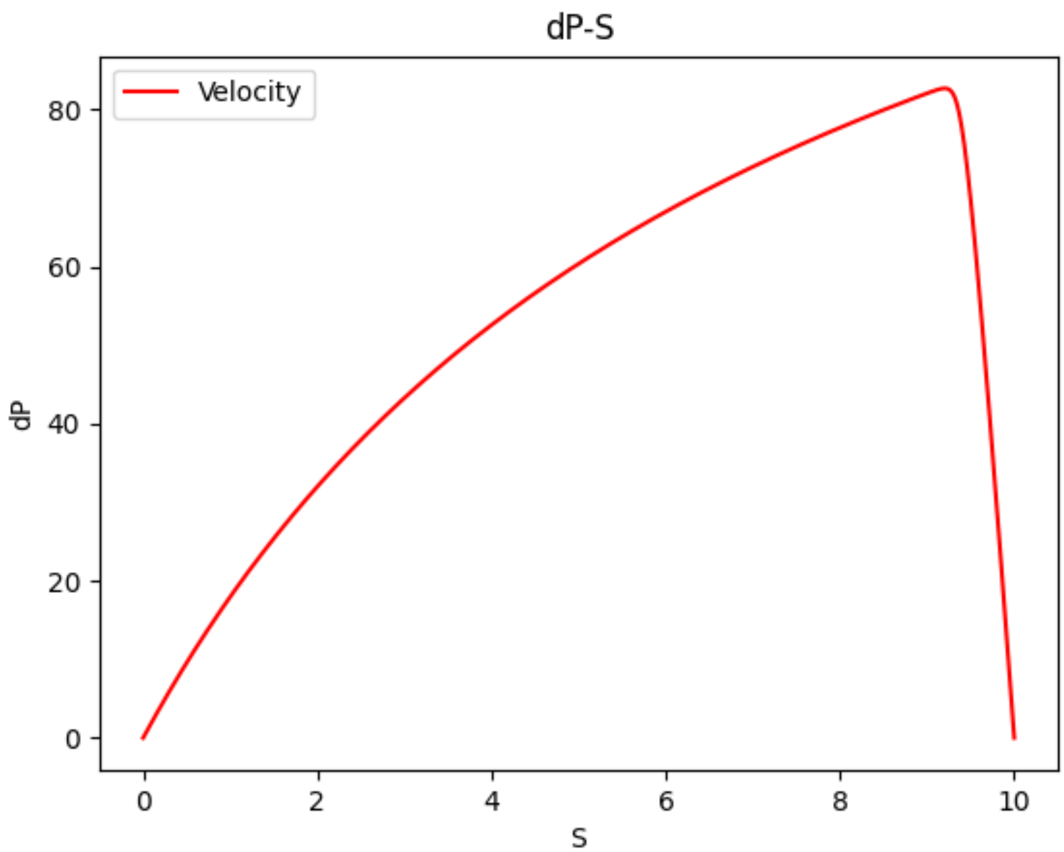
Result:


E-t
S-t
ES-t
P-t
E
S
ES
P
t
RK4
1.0
0.8
0.6
10.0
7.5
5.0
2.5
0.0
0.4
0.2
0.00
0.25
0.50
0.75
1.00

3.

```
### Plot
V=ES*k3
plt.title("dP-S")
plt.plot(S, V, color='red', linestyle='-', marker='', label="Velocity")
plt.legend(loc='upper left')
plt.xlabel('S')
plt.ylabel('dP')
plt.show()
print (max(V)) # per minutes
```

Result:



From the plot, we could find that Vm= 82.69595761906773.