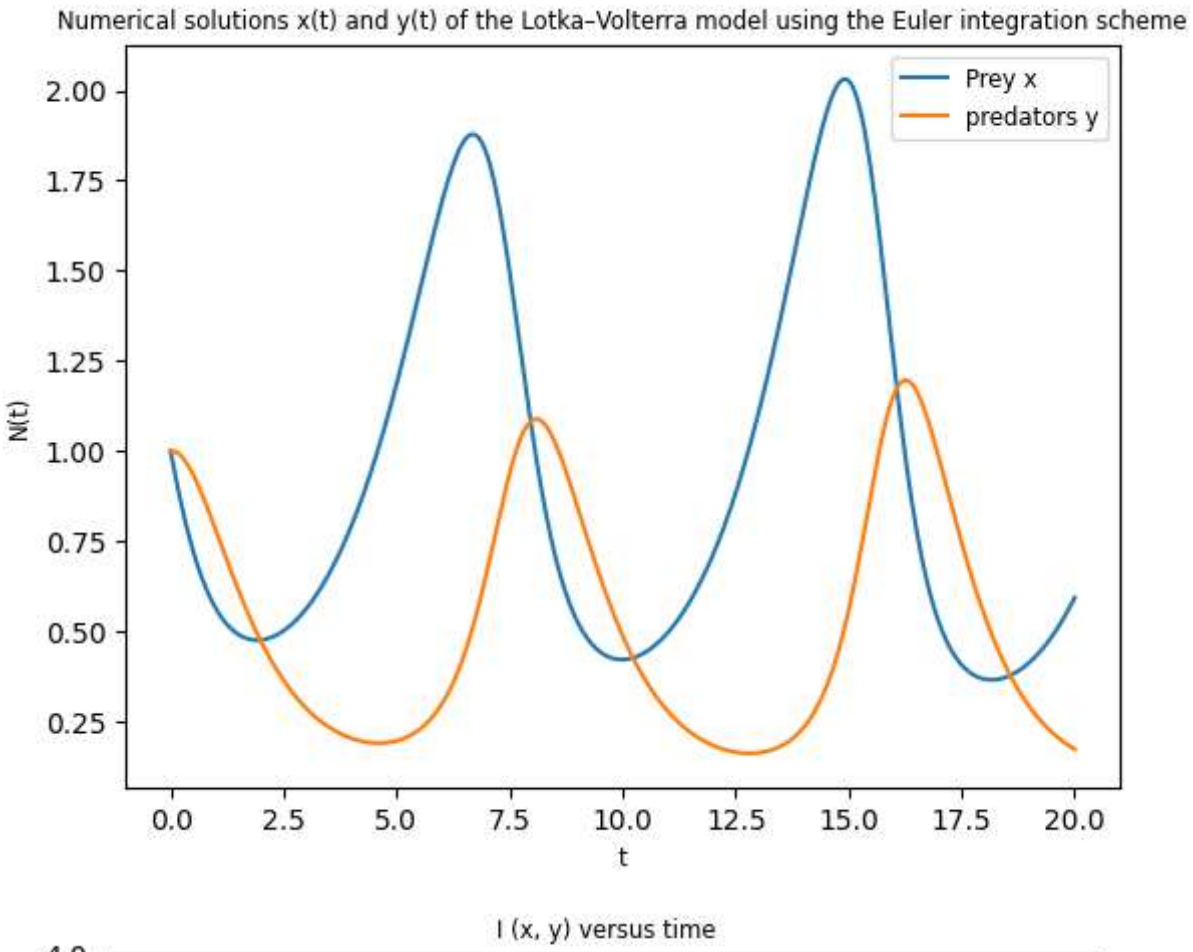
14_3 Explicit Euler integration of the Lotka–Volterra model

In [ ]:
```
import numpy as np
from matplotlib import pyplot as plt
from scipy.integrate import odeint

alpha =2/3
beta = 4/3
gamma = 1
kronicha_delta = 1
x0=1
y0=1
dt=0.05
total_t = int(20/dt)


def calcualte_invariant(x,y):
    return kronicha_delta*x - gamma*np.log(x)+ beta*y - alpha*np.log(y)
print(total_t)

def model(y, t):
    x, y = y
    dxdt = alpha*x - beta*x*y
    dydt = -gamma*y + kronicha_delta*x*y
    return [dxdt, dydt]

x= np.zeros((total_t,1))
y= np.zeros((total_t,1))
x[0]=x0
y[0]=y0
invarient =np.zeros((total_t,1))
invarient[0]= calcualte_invariant(x[0],y[0])
t_list= np.linspace(0,20, total_t)

for i in range(1,total_t):
    x[i]=x[i-1] + (alpha*x[i-1]-beta*x[i-1]*y[i-1])*dt
    y[i]= y[i-1]+ (kronicha_delta*x[i-1]*y[i-1]-gamma*y[i-1])*dt
    invarient[i]= calcualte_invariant(x[i],y[i])

x_initials = np.linspace(1, 2, 3)
y_initials = np.linspace(1, 2, 3)

plt.plot(t_list,x, label='Prey x')
plt.plot(t_list,y, label='predators y')
plt.legend(fontsize ='small')
plt.xlabel('t',fontsize ='small')
plt.ylabel('N(t)',fontsize ='small')
plt.title('Numerical solutions x(t) and y(t) of the Lotka-Volterra model using the Euler integration scheme',fontsize
plt.xticks(fontsize='10')
plt.yticks(fontsize='10')
plt.show()

matrix = np.full((total_t,1), [invarient[0]])

plt.plot(t_list, invarient, label = 'Numerical solution' )
plt.plot(t_list,matrix, linestyle ='dashed', color='black')
plt.xlabel('t',fontsize ='small')
plt.ylabel('I(t)',fontsize ='small')
plt.title('I (x, y) versus time',fontsize ='small')
plt.ylim(0,4)
plt.legend(fontsize ='small')
plt.xticks(fontsize='10')
plt.yticks(fontsize='10')
plt.show()

plt.plot(x,y)
plt.plot(x[0],y[0], marker ='s', label ='intial point')
plt.plot(x[-1],y[-1], marker ='o', label ='end point')

plt.xlabel('Preys x', fontsize ='small')
plt.ylabel('Predactors y', fontsize ='small')
for x0 in x_initials:
    for y0 in y_initials:
        initial_conditions = [x0, y0]
        solution = odeint(model, initial_conditions, t_list)
        plt.plot(solution[:, 0], solution[:, 1], color = 'gray', linewidth=0.5)
plt.title('Trajectory in phase space',fontsize ='small')
plt.legend(fontsize='small')
plt.xticks(fontsize='10')
plt.yticks(fontsize='10')

plt.show()
```

```
400
```

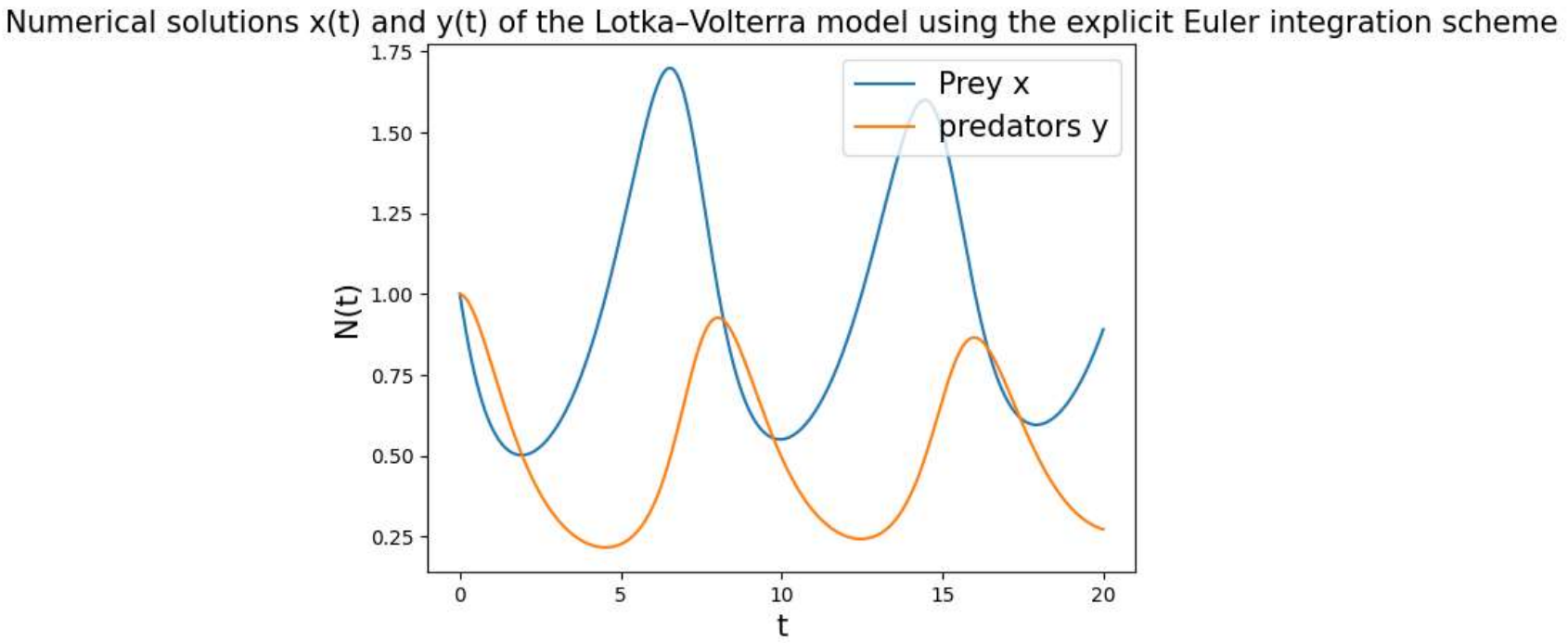

Numerical solutions x(t) and y(t) of the Lotka-Volterra model using the Euler integration scheme
Prey x
predators y
N(t)
t
I (x, y) versus time
Numerical solution
I(t)
t
Trajectory in phase space
intial point
end point
Predactors y
Preys x

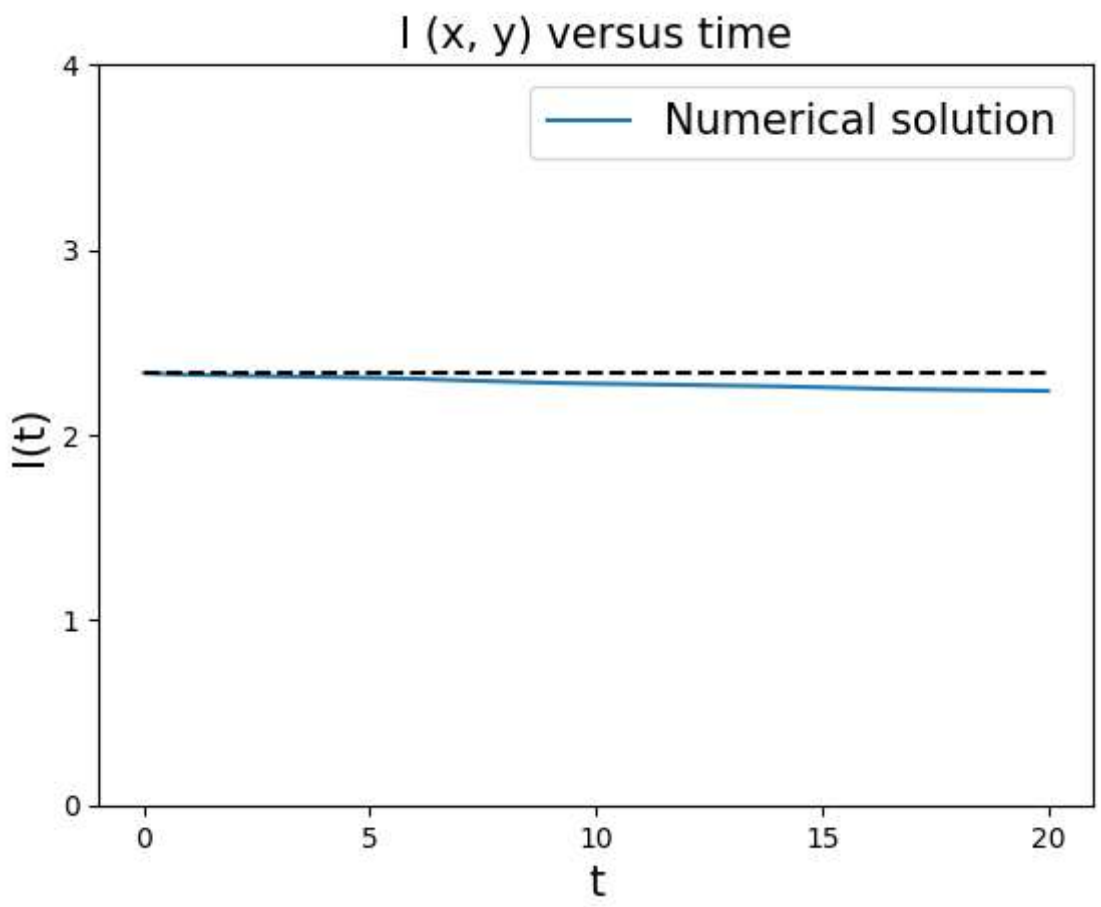
Exercise 14.4. Implicit Euler integration of the Lotka–Volterra model.

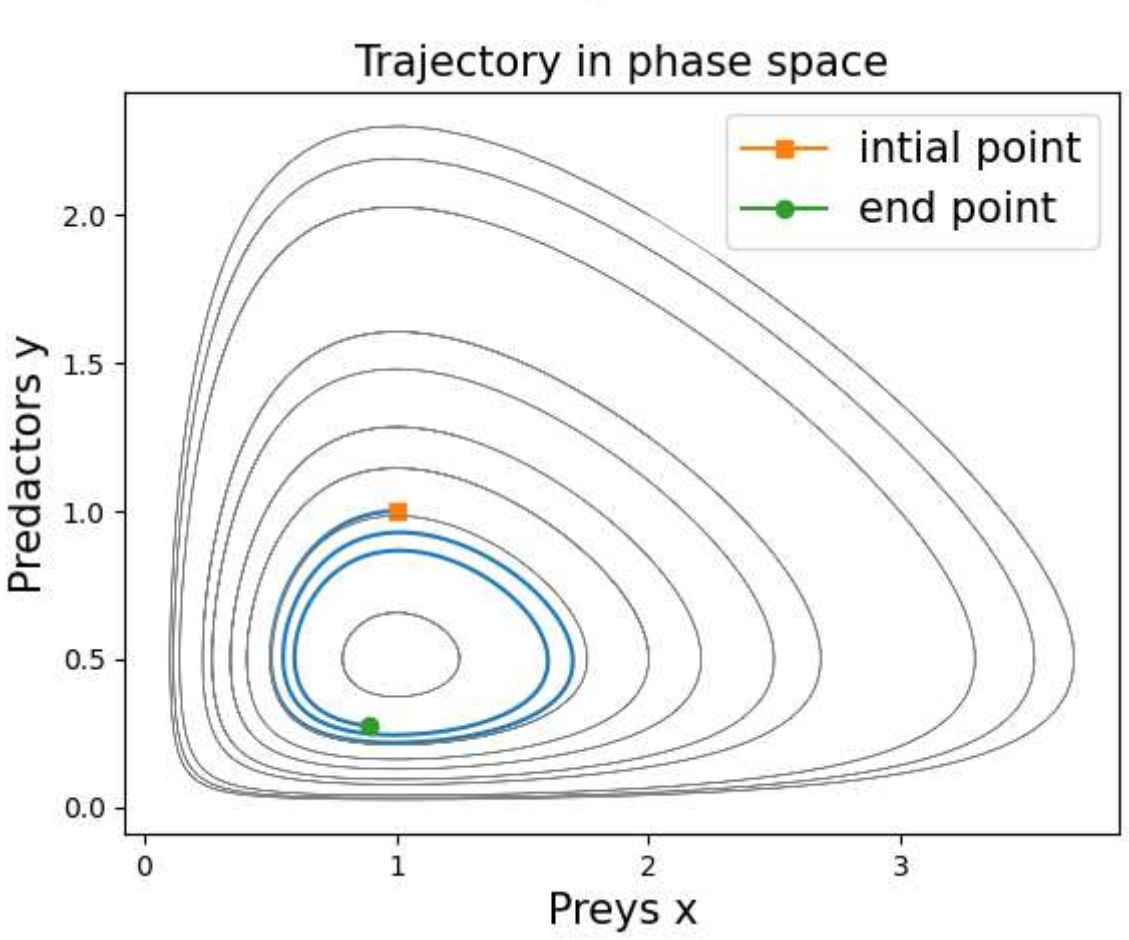
In [ ]:
```
import numpy as np
from matplotlib import pyplot as plt
from scipy.integrate import odeint

alpha =2/3
beta = 4/3
gamma = 1
kronicha_delta = 1
x0=1
y0=1
dt=0.05
total_t = int(20/dt)

def calcualte_invariant(x,y):
    return kronicha_delta*x - gamma*np.log(x)+ beta*y - alpha*np.log(y)
print(total_t)

def model(y, t):
    x, y = y
    dxdt = alpha*x - beta*x*y
    dydt = -gamma*y + kronicha_delta*x*y
    return [dxdt, dydt]


x= np.zeros((total_t,1))
y= np.zeros((total_t,1))
x[0]=x0
y[0]=y0
invarient =np.zeros((total_t,1))
invarient[0]= calcualte_invariant(x[0],y[0])
t_list= np.linspace(0,20, total_t)



for i in range(1,total_t):
    a_x= (1-dt*alpha)*dt*kronicha_delta
    b_x = -((1-dt*alpha)*(1+dt*gamma)+dt*(kronicha_delta*x[i-1]+beta*y[i-1]))
    c_x = (1+dt*gamma)*x[i-1]
    # quadratic_solutionx = [(-b_x+np.sqrt(b_x**2-4*a_x*c_x))/(2*a_x),(-b_x-np.sqrt(b_x**2-4*a_x*c_x))/(2*a_x)]
    quadratic_solutionx = [(-b_x+np.sqrt(b_x**2-4*a_x*c_x))/(2*a_x),(-b_x-np.sqrt(b_x**2-4*a_x*c_x))/(2*a_x)]

    # x_inter = quadratic_solutionx[np.argmin(abs(quadratic_solutionx-x[i-1]))]
    x_inter = quadratic_solutionx[np.argmin(abs(quadratic_solutionx-x[i-1]))]

    a_y = (1+dt*gamma)*dt*beta
    b_y = ((1-dt*alpha)*(1+dt*gamma)-dt*(kronicha_delta*x[i-1]+beta*y[i-1]))
    c_y = -(1-dt*alpha)*y[i-1]
    quadratic_solutiony = [(-b_y+np.sqrt(b_y**2-4*a_y*c_y))/(2*a_y),(-b_y-np.sqrt(b_y**2-4*a_y*c_y))/(2*a_y)]
    y_inter = quadratic_solutiony[np.argmin(abs(quadratic_solutiony-y[i-1]))]
    x[i] = x[i-1]+ (alpha*x_inter-beta*x_inter*y_inter)*dt
    y[i] = y[i-1] + (kronicha_delta*x_inter*y_inter - gamma*y_inter)*dt
    invarient[i]= calcualte_invariant(x[i],y[i])

x_initials = np.linspace(0.5, 2, 3)
y_initials = np.linspace(0.5, 2, 3)

plt.plot(t_list,x, label='Prey x')
plt.plot(t_list,y, label='predators y')
plt.legend(fontsize ='small')
plt.xlabel('t',fontsize ='small')
plt.ylabel('N(t)',fontsize ='small')
plt.title('Numerical solutions x(t) and y(t) of the Lotka-Volterra model using the explicit Euler integration scheme'
plt.xticks(fontsize='10')
plt.yticks(fontsize='10')
plt.show()

matrix = np.full((total_t,1), [invarient[0]])

plt.plot(t_list, invarient, label = 'Numerical solution' )
plt.plot(t_list,matrix, linestyle ='dashed', color='black')
plt.xlabel('t',fontsize ='small')
plt.ylabel('I(t)',fontsize ='small')
plt.title('I (x, y) versus time',fontsize ='small')
plt.ylim(0,4)
plt.legend(fontsize ='small')
plt.xticks(fontsize='10')
plt.yticks(fontsize='10')
plt.show()

plt.plot(x,y)
plt.plot(x[0],y[0], marker ='s', label ='intial point')
plt.plot(x[-1],y[-1], marker ='o', label ='end point')

plt.xlabel('Preys x', fontsize ='small')
plt.ylabel('Predactors y', fontsize ='small')
for x0 in x_initials:
```

```
    for y0 in y_initials:
        initial_conditions = [x0, y0]
        solution = odeint(model, initial_conditions, t_list)
        plt.plot(solution[:, 0], solution[:, 1], color = 'gray', linewidth=0.5)
plt.title('Trajectory in phase space',fontsize ='small')
plt.legend(fontsize='small')
plt.xticks(fontsize='10')
plt.yticks(fontsize='10')

plt.show()
```

```
400
```







14.4

$$x_{n+1} = x_n + (\alpha x_{n+1} - \beta x_{n+1} y_{n+1})\Delta t$$

$$x_{n+1} - \alpha x_{n+1}\Delta t + \beta x_{n+1} y_{n+1}\Delta t = x_n$$

$$x_{n+1}(1 - \alpha\Delta t + \beta y_{n+1}\Delta t) = x_n$$

$$x_{n+1} = \frac{x_n}{1 - \alpha\Delta t + \beta y_{n+1}\Delta t}$$

$$y_{n+1} = y_n + (\delta x_{n+1} y_{n+1} - \gamma y_{n+1})\Delta t$$

$$y_{n+1} = y_n + \delta x_{n+1} y_{n+1}\Delta t - \gamma y_{n+1}\Delta t$$

$$y_{n+1} - \delta x_{n+1} y_{n+1}\Delta t + \gamma y_{n+1}\Delta t = y_n$$

$$y_{n+1}(1 - \delta x_{n+1}\Delta t + \gamma\Delta t) = y_n$$

$$y_{n+1} = \frac{y_n}{(1 - \delta x_{n+1}\Delta t + \gamma\Delta t)}$$

Sub x_{n+1} in y_{n+1}

$$y_{n+1} = \frac{y_n}{\left(1 - \delta\left(\frac{x_n}{1-\alpha\Delta t+\beta y_{n+1}\Delta t}\right)\Delta t + \gamma\Delta t\right)}$$

$$1 - \alpha\Delta t + \beta y_{n+1}\Delta t - \delta x_n\Delta t + \gamma\Delta t\left[1 - \alpha\Delta t + \beta y_{n+1}\Delta t\right]$$

$$1 - \alpha\Delta t + \beta y_{n+1}\Delta t - \delta x_n\Delta t + \gamma\Delta t - \alpha\Delta t^2\gamma + \gamma\beta y_{n+1}\Delta t^2$$

$$\Rightarrow y_{n+1} - \alpha\Delta t\, y_{n+1} + \beta(y_{n+1})^2\Delta t - \delta x_n y_{n+1}\Delta t + \gamma\Delta t\, y_{n+1} - \alpha\Delta t^2\gamma\, y_{n+1} + \gamma\beta(y_{n+1})^2\Delta t^2$$

$$= y_n\left[1 - \alpha\Delta t + \beta y_{n+1}\Delta t\right]$$

$$= y_n - \alpha\Delta t\, y_n + \beta y_{n+1} y_n\Delta t$$

$$(y_{n+1})^2\left[\beta\Delta t + \gamma\beta\Delta t^2\right] + y_{n+1}\left[1 - \alpha\Delta t - \delta x_n\Delta t + \gamma\Delta t - \alpha\Delta t^2\gamma - \beta y_n\Delta t\right]$$

$$- y_n - \alpha\Delta t\, y_n = 0$$

Sol $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

$$a_y = \left[\beta\Delta t + \gamma\beta\Delta t^2\right]$$

$$b_y = \left[1 - \alpha\Delta t - \delta x_n\Delta t + \gamma\Delta t - \alpha\Delta t^2\gamma - \beta y_n\Delta t\right]$$

$$C_y = -y_n - \alpha\Delta t\, y_n$$

To find a_x b_x c_x Sub y_{n+1} in x_{n+1}

$$x_{n+1} = \frac{x_n}{1 - \alpha\Delta t + \beta y_{n+1}\Delta t} \qquad y_{n+1} = \frac{y_n}{(1 - \delta x_{n+1}\Delta t + \gamma\Delta t)}$$

$$x_{n+1} = \frac{x_n}{1 - \alpha\Delta t + \beta \dfrac{y_n}{1 - \Delta t\,\delta x_{n+1} + \gamma\Delta t}\,\Delta t}$$

$$\frac{1 - \Delta t\,\delta x_{n+1} + \gamma\Delta t - \alpha\Delta t + \beta y_n \Delta t - \alpha\Delta t^2\delta x_{n+1} - \alpha\Delta t^2\gamma}{1 - \Delta t\,\delta x_{n+1} + \gamma\Delta t}$$

$$x_{n+1} - \Delta t\,\delta (x_{n+1})^2 + \gamma\Delta t\, x_{n+1} - \alpha\Delta t\, x_{n+1} + \beta y_n x_{n+1}\Delta t - \alpha\Delta t\, x_{n+1} + \alpha\Delta t^2\delta (x_{n+1})^2 - \alpha\Delta t^2\gamma\, x_{n+1} = x_n - \Delta t\,\delta x_{n+1} x_n + \gamma\Delta t\, x_n$$

$$(x_{n+1})^2\left[\alpha\Delta t^2\delta - \Delta t\,\delta\right] + \left[1 + \gamma\Delta t - \alpha\Delta t + \beta y_n\Delta t - \alpha\Delta t - \alpha\Delta t^2\gamma + \Delta t\,\delta x\right] - x_n - \gamma\Delta t x_n = 0$$

Multiply by '-1'

$$a = \Delta t\,\delta - \alpha\Delta t^2\delta$$

$$b = -\left[1 - \gamma\Delta t - 2\alpha\Delta t + \beta y_n\Delta t - \Delta t^2\alpha\gamma + \Delta t\,\delta x_n\right]$$

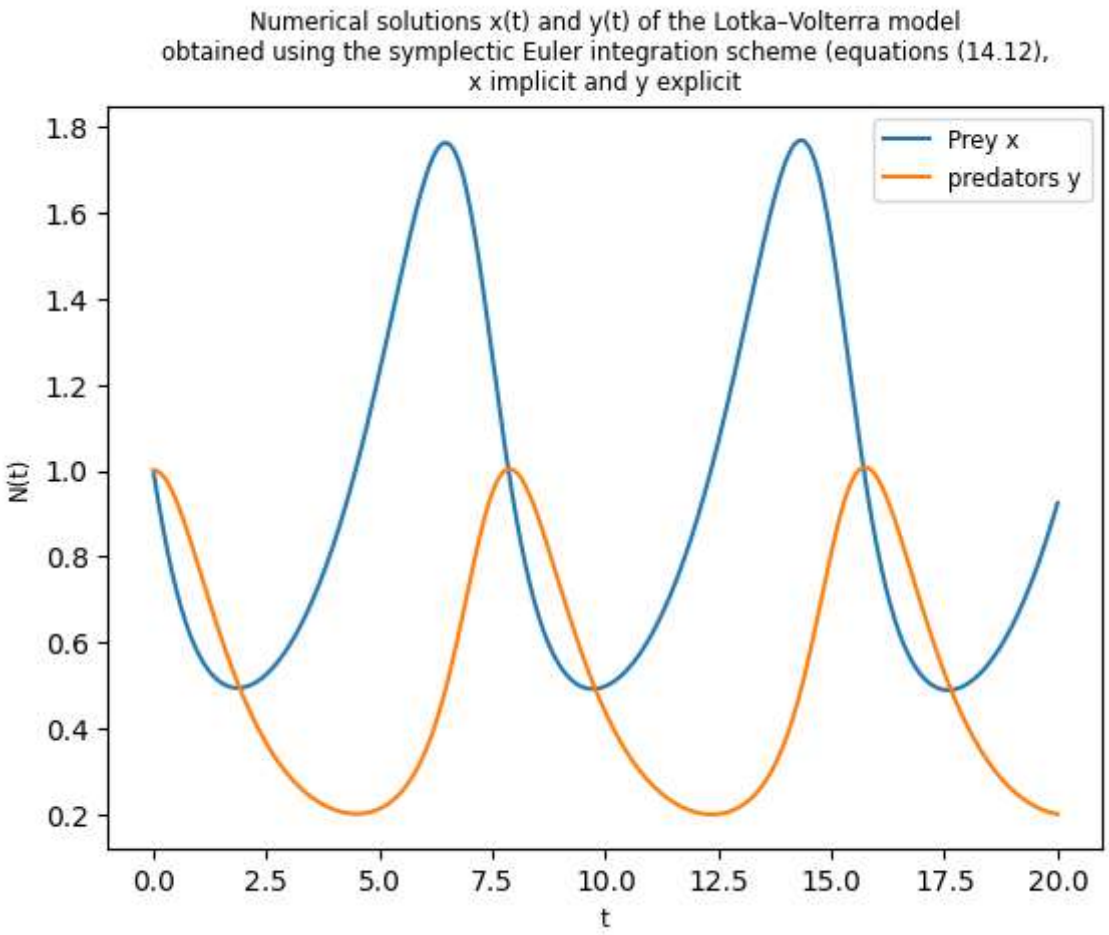
$$c = x_n + \gamma\Delta t\, x_n$$

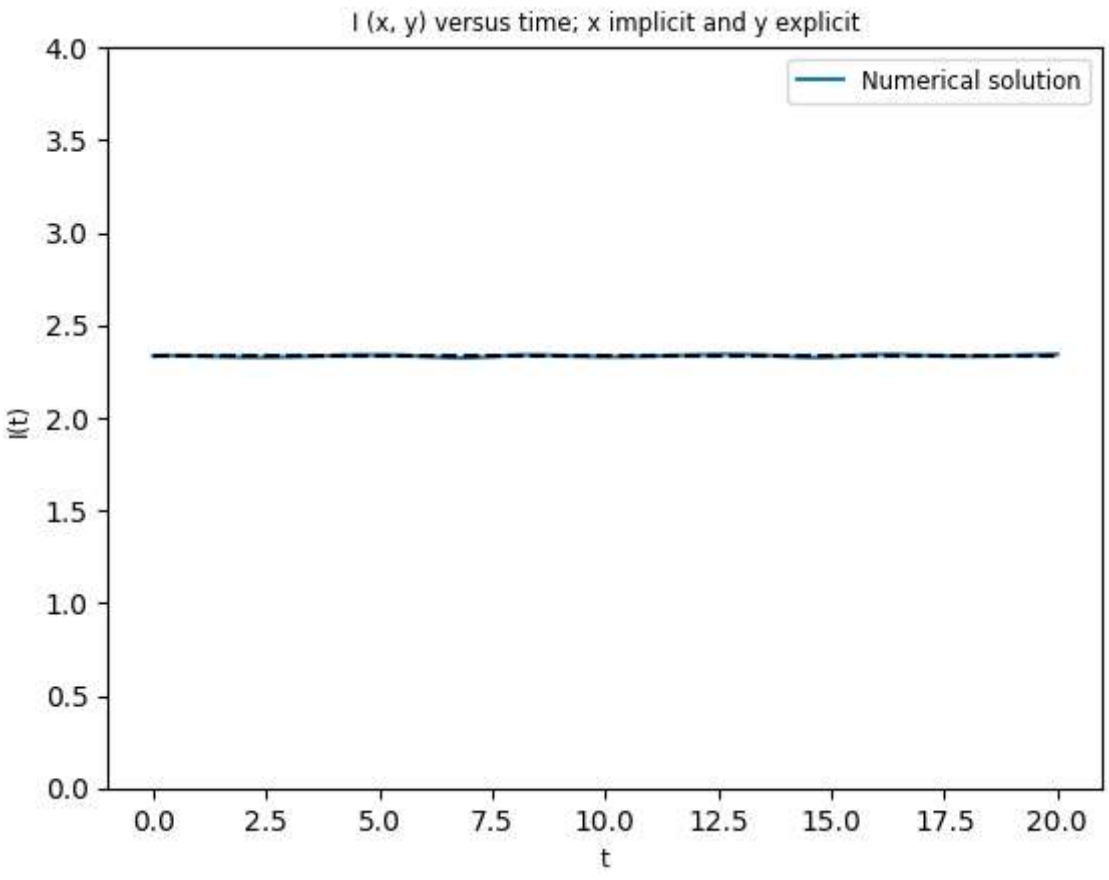
Exercise 14.5. Symplectic Euler integration of the Lotka–Volterra model.
with x explicit and y implicit

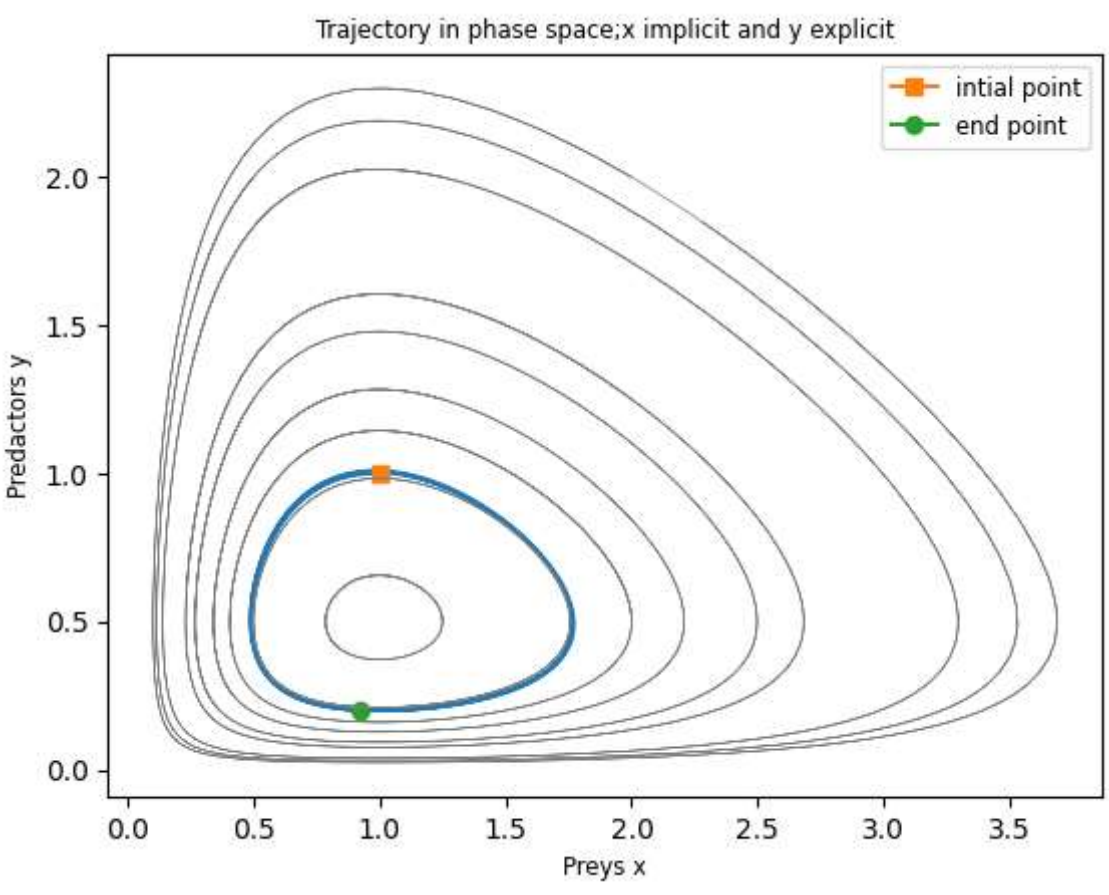
In [ ]:
```
import numpy as np
from matplotlib import pyplot as plt
from scipy.integrate import odeint

alpha =2/3
beta = 4/3
gamma = 1
kronicha_delta = 1
x0=1
y0=1
dt=0.05
total_t = int(20/dt)

def calcualte_invariant(x,y):
    return kronicha_delta*x - gamma*np.log(x)+ beta*y - alpha*np.log(y)
print(total_t)

def model(y, t):
    x, y = y
    dxdt = alpha*x - beta*x*y
    dydt = -gamma*y + kronicha_delta*x*y
    return [dxdt, dydt]


x= np.zeros((total_t,1))
y= np.zeros((total_t,1))
x[0]=x0
y[0]=y0
invarient =np.zeros((total_t,1))
invarient[0]= calcualte_invariant(x[0],y[0])
t_list= np.linspace(0,20, total_t)

for i in range(1,total_t):
    y_inter = y[i-1]/(1-(dt*(kronicha_delta*x[i-1]-gamma)))

    x[i] = x[i-1] + (alpha*x[i-1] - beta*x[i-1]*y_inter)*dt
    y[i] = y[i-1] + (kronicha_delta*x[i-1]*y_inter- gamma*y[i-1])*dt
    invarient[i]= calcualte_invariant(x[i],y[i])

x_initials = np.linspace(0.5, 2, 3)
y_initials = np.linspace(0.5, 2, 3)

plt.plot(t_list,x, label='Prey x')
plt.plot(t_list,y, label='predators y')
plt.legend(fontsize ='small')
plt.xlabel('t',fontsize ='small')
plt.ylabel('N(t)',fontsize ='small')
# plt.title('Numerical solutions x(t) and y(t) of the Lotka-Volterra model' \'obtained using the symplectic Euler int
plt.title('Numerical solutions x(t) and y(t) of the Lotka-Volterra model\n'
          'obtained using the symplectic Euler integration scheme (equations (14.12),\n'
          'x implicit and y explicit', fontsize='small')
plt.xticks(fontsize='10')
plt.yticks(fontsize='10')
plt.show()

matrix = np.full((total_t,1), [invarient[0]])

plt.plot(t_list, invarient, label = 'Numerical solution' )
plt.plot(t_list,matrix, linestyle ='dashed', color='black')
plt.xlabel('t',fontsize ='small')
plt.ylabel('I(t)',fontsize ='small')
plt.title('I (x, y) versus time; x implicit and y explicit',fontsize ='small')
plt.ylim(0,4)
plt.legend(fontsize ='small')
plt.xticks(fontsize='10')
plt.yticks(fontsize='10')
plt.show()

plt.plot(x,y)
plt.plot(x[0],y[0], marker ='s', label ='intial point')
plt.plot(x[-1],y[-1], marker ='o', label ='end point')

plt.xlabel('Preys x', fontsize ='small')
plt.ylabel('Predactors y', fontsize ='small')
for x0 in x_initials:
    for y0 in y_initials:
        initial_conditions = [x0, y0]
        solution = odeint(model, initial_conditions, t_list)
        plt.plot(solution[:, 0], solution[:, 1], color = 'gray', linewidth=0.5)
plt.title('Trajectory in phase space;x implicit and y explicit',fontsize ='small')
plt.legend(fontsize='small')
plt.xticks(fontsize='10')
plt.yticks(fontsize='10')
```

```
plt.show()
```

```
400
```



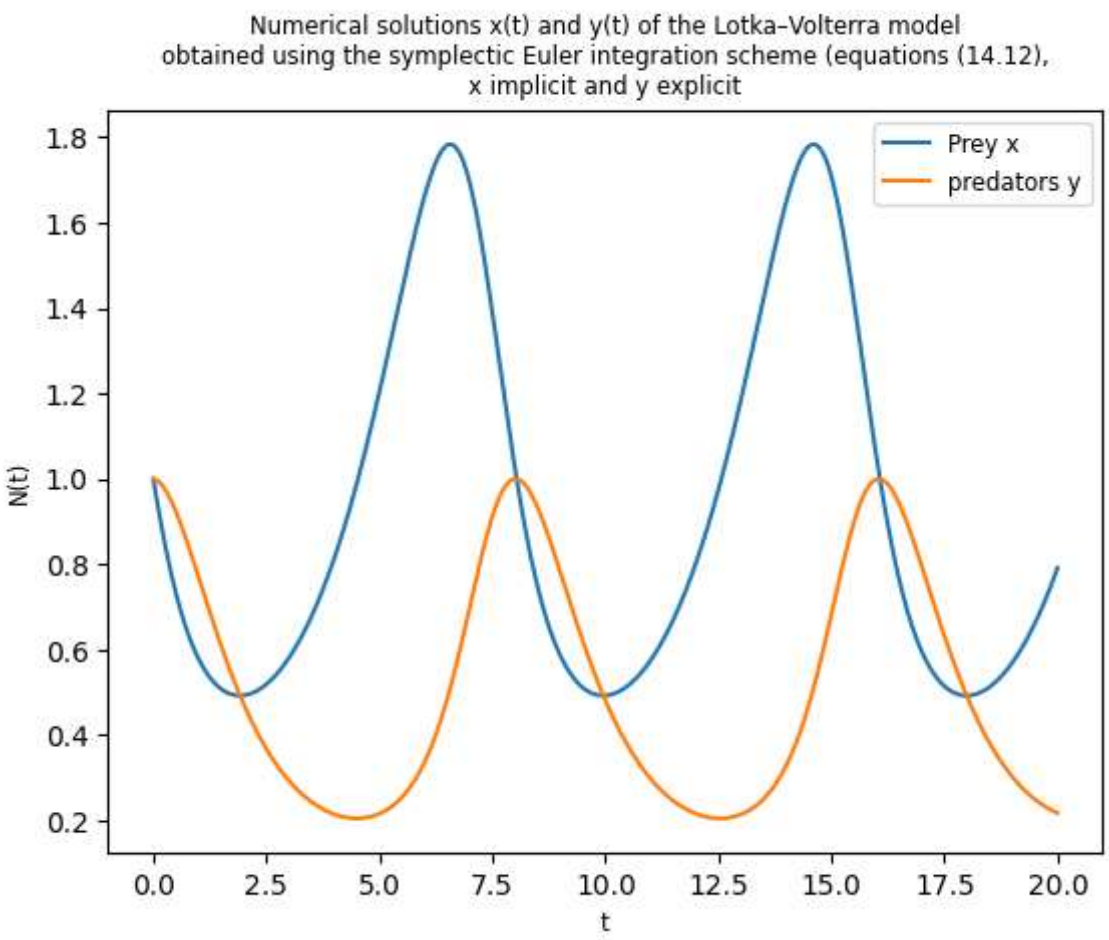


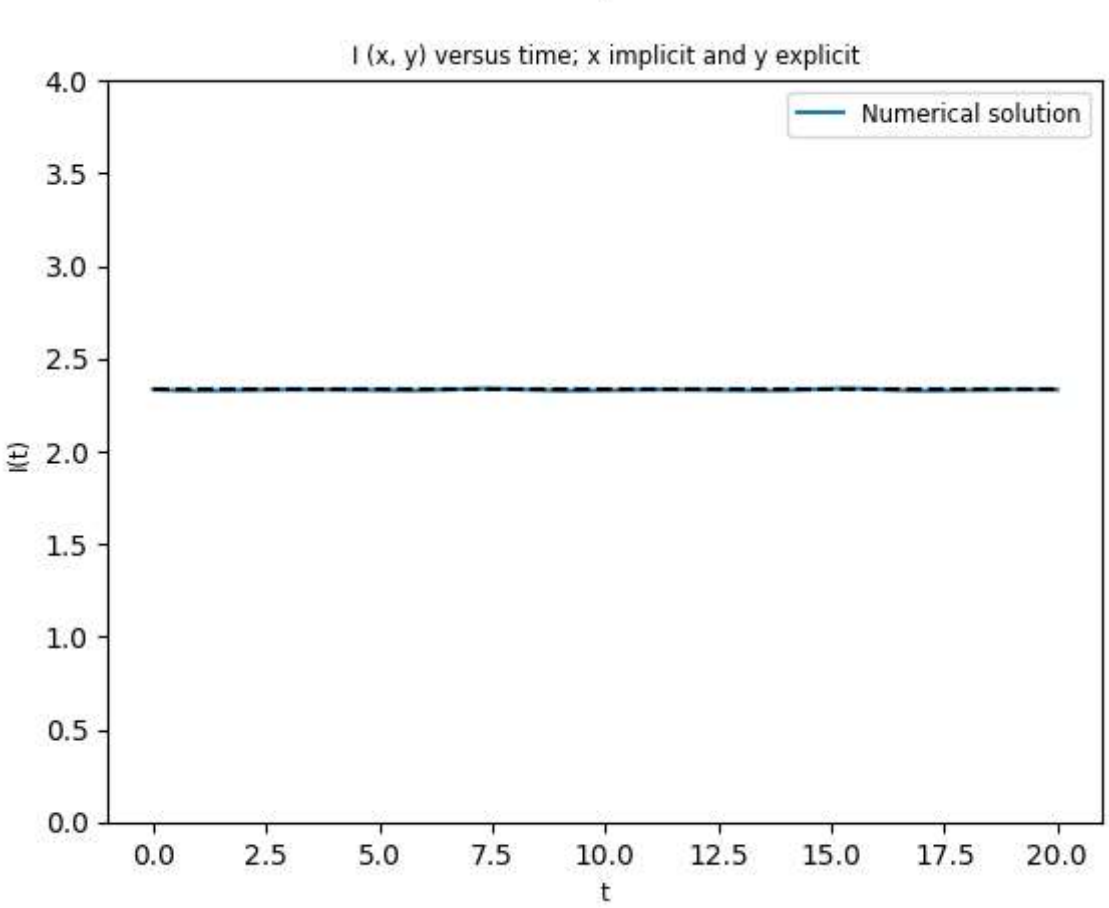


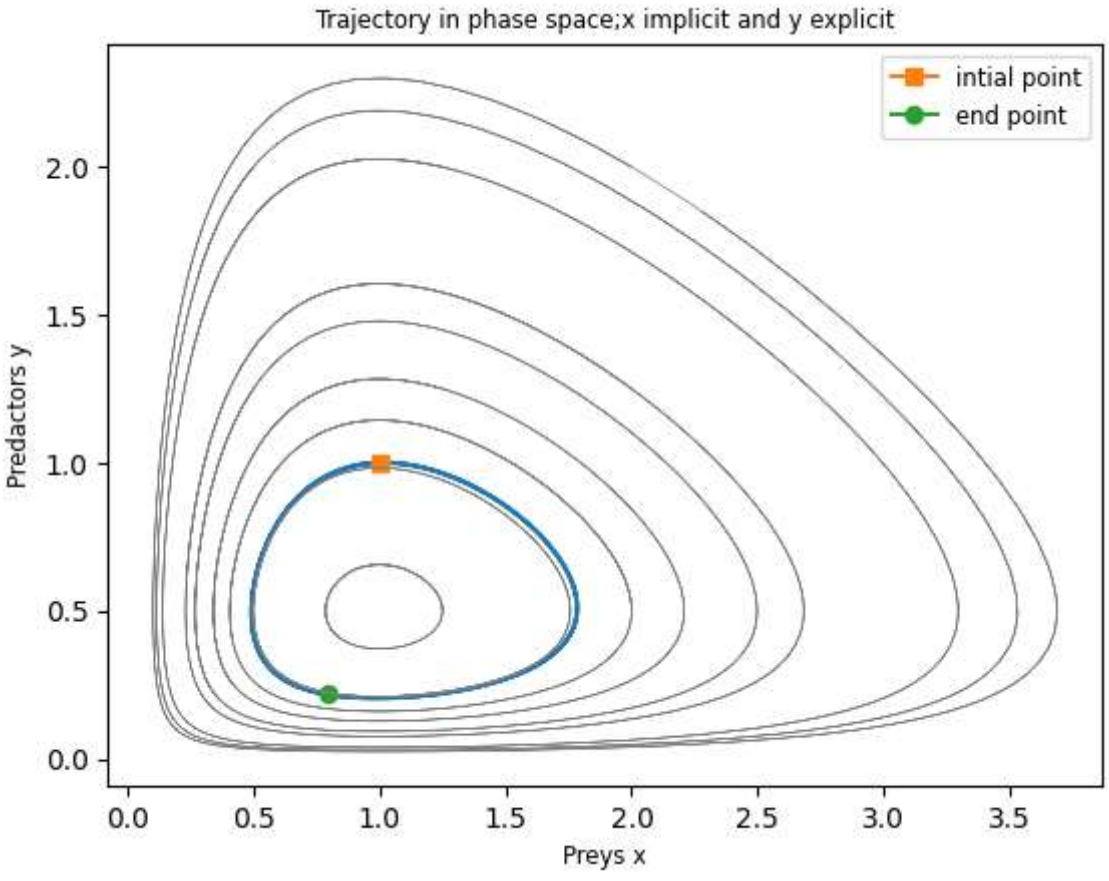
In [ ]:
```
alpha =2/3
beta = 4/3
gamma = 1
```

```
kronicha_delta = 1
x0=1
y0=1
dt=0.05
total_t = int(20/dt)

def calcualte_invariant(x,y):
    return kronicha_delta*x - gamma*np.log(x)+ beta*y - alpha*np.log(y)
print(total_t)

def model(y, t):
    x, y = y
    dxdt = alpha*x - beta*x*y
    dydt = -gamma*y + kronicha_delta*x*y
    return [dxdt, dydt]


x= np.zeros((total_t,1))
y= np.zeros((total_t,1))
x[0]=x0
y[0]=y0
invarient =np.zeros((total_t,1))
invarient[0]= calcualte_invariant(x[0],y[0])
t_list= np.linspace(0,20, total_t)

for i in range(1,total_t):
    x_inter = x[i-1]/(1-dt*(alpha- beta*y[i-1]))

    x[i]= x[i-1] + (alpha*x_inter - beta* x_inter*y[i-1])*dt
    y[i] = y[i-1] + (kronicha_delta*x_inter*y[i-1]-gamma*y[i-1])*dt

    invarient[i]= calcualte_invariant(x[i],y[i])

x_initials = np.linspace(0.5, 2, 3)
y_initials = np.linspace(0.5, 2, 3)

plt.plot(t_list,x, label='Prey x')
plt.plot(t_list,y, label='predators y')
plt.legend(fontsize ='small')
plt.xlabel('t',fontsize ='small')
plt.ylabel('N(t)',fontsize ='small')
# plt.title('Numerical solutions x(t) and y(t) of the Lotka-Volterra model' \'obtained using the symplectic Euler int
plt.title('Numerical solutions x(t) and y(t) of the Lotka-Volterra model\n'
          'obtained using the symplectic Euler integration scheme (equations (14.12),\n'
          'x implicit and y explicit', fontsize='small')
plt.xticks(fontsize='10')
plt.yticks(fontsize='10')
plt.show()

matrix = np.full((total_t,1), [invarient[0]])

plt.plot(t_list, invarient, label = 'Numerical solution' )
plt.plot(t_list,matrix, linestyle ='dashed', color='black')
plt.xlabel('t',fontsize ='small')
plt.ylabel('I(t)',fontsize ='small')
plt.title('I (x, y) versus time; x implicit and y explicit',fontsize ='small')
plt.ylim(0,4)
plt.legend(fontsize ='small')
plt.xticks(fontsize='10')
plt.yticks(fontsize='10')
plt.show()


plt.plot(x,y)
plt.plot(x[0],y[0], marker ='s', label ='intial point')
plt.plot(x[-1],y[-1], marker ='o', label ='end point')

plt.xlabel('Preys x', fontsize ='small')
plt.ylabel('Predactors y', fontsize ='small')
for x0 in x_initials:
    for y0 in y_initials:
        initial_conditions = [x0, y0]
        solution = odeint(model, initial_conditions, t_list)
        plt.plot(solution[:, 0], solution[:, 1], color = 'gray', linewidth=0.5)
plt.title('Trajectory in phase space;x implicit and y explicit',fontsize ='small')
plt.legend(fontsize='small')
plt.xticks(fontsize='10')
plt.yticks(fontsize='10')

plt.show()
```

```
400
```


Numerical solutions x(t) and y(t) of the Lotka–Volterra model
obtained using the symplectic Euler integration scheme (equations (14.12),
x implicit and y explicit
Prey x
predators y
N(t)
t


I (x, y) versus time; x implicit and y explicit
Numerical solution
I(t)
t


Trajectory in phase space;x implicit and y explicit
intial point
end point
Predactors y
Preys x

14.5) x explicit y implicit

$$\bar{x}_{n+1} = x_n + (\alpha x_n - \beta x_n y_{n+1})\,dt$$

$$\tilde{y}_{n+1} = y_n + (\delta x_n y_{n+1} - \gamma y_n)\,dt$$

$$\tilde{y}_{n+1} - \delta x_n y_{n+1}\,dt = y_n - \gamma y_n\,dt$$

$$\tilde{y}_{n+1} = \frac{y_n}{1 - \delta x_n\,dt - \gamma\,dt}$$

Sub $\tilde{y}_{n+1}$ in $\tilde{x}_{n+1}$

$$x_{n+1} = x_n + (\alpha x_n - \beta x_n y_{n+1})\,dt$$

$$x_{n+1} = x_n \left[1 + \left(\alpha - \frac{\beta y_n}{1 - dt(\delta x_n - \gamma)}\right) dt\right]$$

Eq: 14.12 x implicit y explicit

$$x_{n+1} = x_n + (\alpha x_{n+1} - \beta x_{n+1} y_n)\,dt$$

$$x_{n+1} = x_n + \alpha x_{n+1}\,dt - \beta x_{n+1} y_n\,dt$$

$$x_{n+1} - \alpha x_{n+1}\,dt + \beta x_{n+1} y_n\,dt = x_n$$

$$x_{n+1}\left[1 - \alpha\,dt + \beta y_n\,dt\right] = x_n$$

$$x_{n+1} = \frac{x_n}{1 - \alpha\,dt - \beta y_n\,dt}$$

Now sub x_{n+1} in y_{n+1}

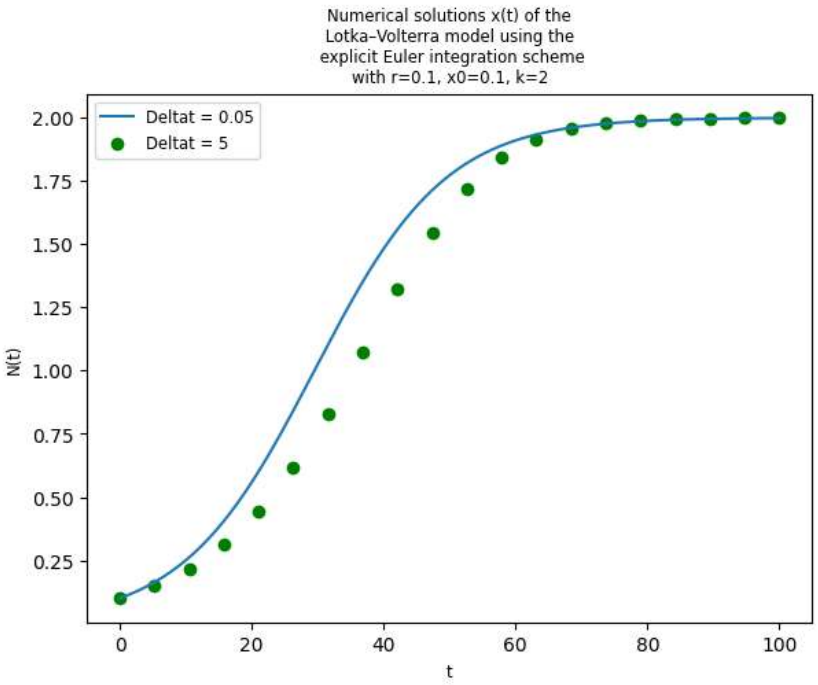
$$y_{n+1} = y_n + (\delta x_{n+1} y_n - \gamma y_n)\,dt$$

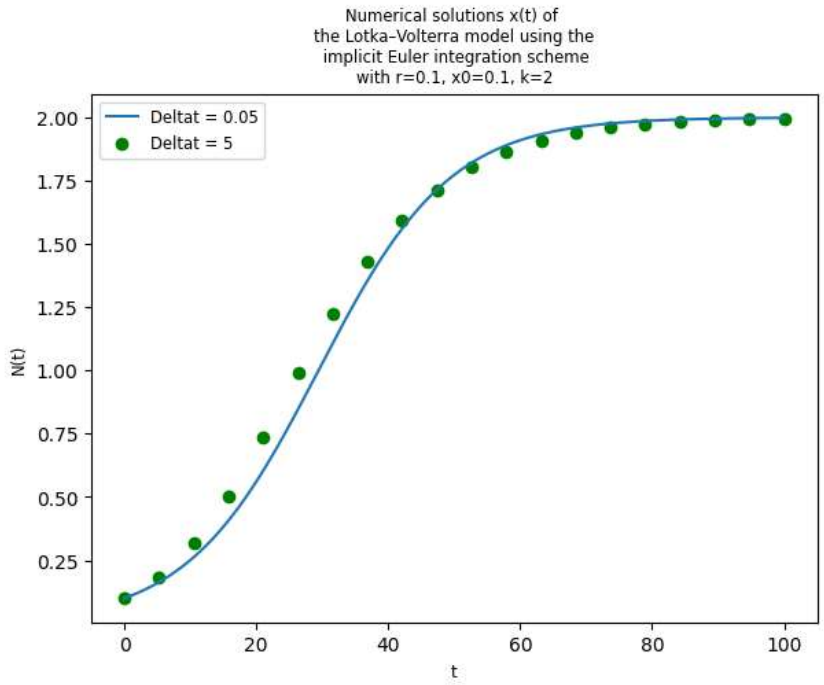
$$y_{n+1} = y_n + \left[\delta \frac{x_n y_n\,dt}{1 - \alpha\,dt - \beta y_n\,dt} - \gamma y_n\right]$$

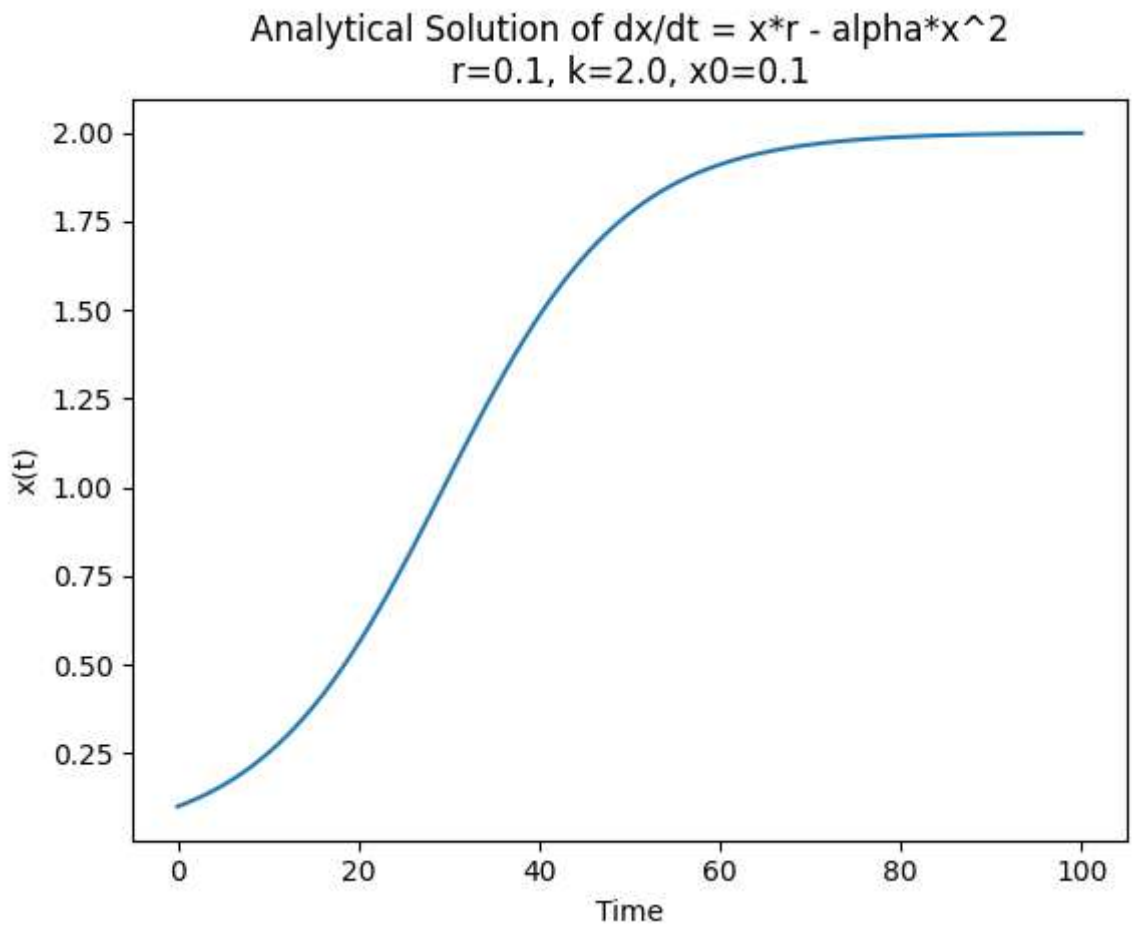
$$y_{n+1} = y_n \left[1 + dt \left[\frac{\delta x_n}{1 - dt(\alpha - \beta y_n)} - \gamma \right] \right]$$

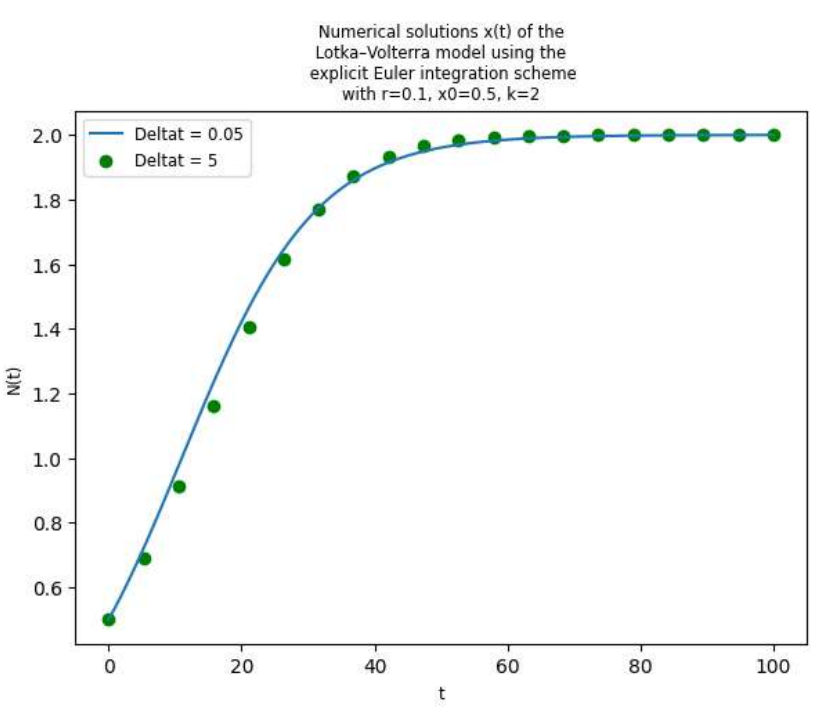
Exercise 14.7. Logistic growth model.

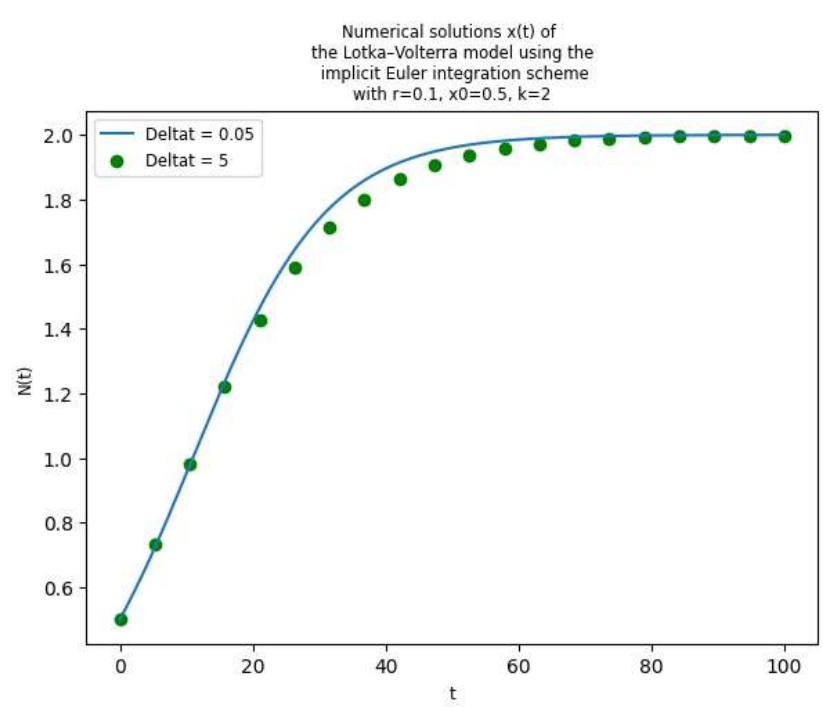
In [ ]:
```
import numpy as np
from matplotlib import pyplot as plt
from scipy.integrate import odeint

r_list=[0.1,0.3]
k_list =[2,3]
x0_list=[0.1,0.5,1]
dt_list= [0.05,5]
time = 100

def analytical_formula(t_list,K,x_0):
    sol=[]
    for t in t_list:
        denominoator = 1 + (((K / x_0) - 1) * np.exp(-r * (t - t_list[0])))
        sol.append(K/denominoator)
    return sol


def model(x, t, r, alpha):
    dxdt = x * r - alpha * x**2
    return dxdt

for r in r_list:
    for k in k_list:
        for x0 in x0_list:
            plt.figure(figsize=(15, 5))
            for dt in dt_list:
                total_t = int(time/dt)
                x_euler= np.zeros((total_t,1))
                x_euler[0]=x0

                x2 = np.zeros((total_t,1))
                x2[0]=x0

                t_list= np.linspace(0,time, total_t)

                for i in range(1,total_t):
                    x_euler[i]=x_euler[i-1] + (r*x_euler[i-1]*(1-(x_euler[i-1]/k)))*dt
                    # invarient[i]= calcualte_invariant(x[i],y[i])

                    x2[i] = k*(-(1-r*dt)+ np.sqrt((1-r*dt)**2+ 4*r*dt*x2[i-1]/k))/(2*r*dt)


                x_initials = np.linspace(1, 2, 3)
                y_initials = np.linspace(1, 2, 3)

                plt.subplot(1,2,1)
                if dt ==0.05:
                    plt.plot(t_list,x_euler, label='Deltat = 0.05')
                else:
                    plt.scatter(t_list,x_euler, label='Deltat = 5', color='green')
                plt.legend(fontsize ='small')
                plt.xlabel('t',fontsize ='small')
                plt.ylabel('N(t)',fontsize ='small')
                plt.title('Numerical solutions x(t) of the\n Lotka-Volterra model using the \n explicit Euler integra
                    'with r={}, x0={}, k={}'.format(r, x0, k), fontsize='small')
                plt.xticks(fontsize='10')
                plt.yticks(fontsize='10')
                plt.subplot(1,2,2)
                if dt ==0.05:
                    plt.plot(t_list,x2, label='Deltat = 0.05')
                else:
                    plt.scatter(t_list,x2, label='Deltat = 5', color='green')
                plt.legend(fontsize ='small')
                plt.xlabel('t',fontsize ='small')
                plt.ylabel('N(t)',fontsize ='small')
                plt.title('Numerical solutions x(t) of \nthe Lotka-Volterra model using the\n implicit Euler integrat
                    'with r={}, x0={}, k={}'.format(r, x0, k), fontsize='small')

                plt.xticks(fontsize='10')
                plt.yticks(fontsize='10')

            plt.show()
            t = np.linspace(0, 100, 100)
            alpha = r/k
            solution = analytical_formula(t,k,x0)
            # solution = odeint(model, x0, t, args=(r, alpha))
            plt.plot(t, solution)
            plt.xlabel('Time')
            plt.ylabel('x(t)')
            plt.title(f'Analytical Solution of dx/dt = x*r - alpha*x^2\nr={r}, k={r/alpha}, x0={x0}')
            plt.show()
```

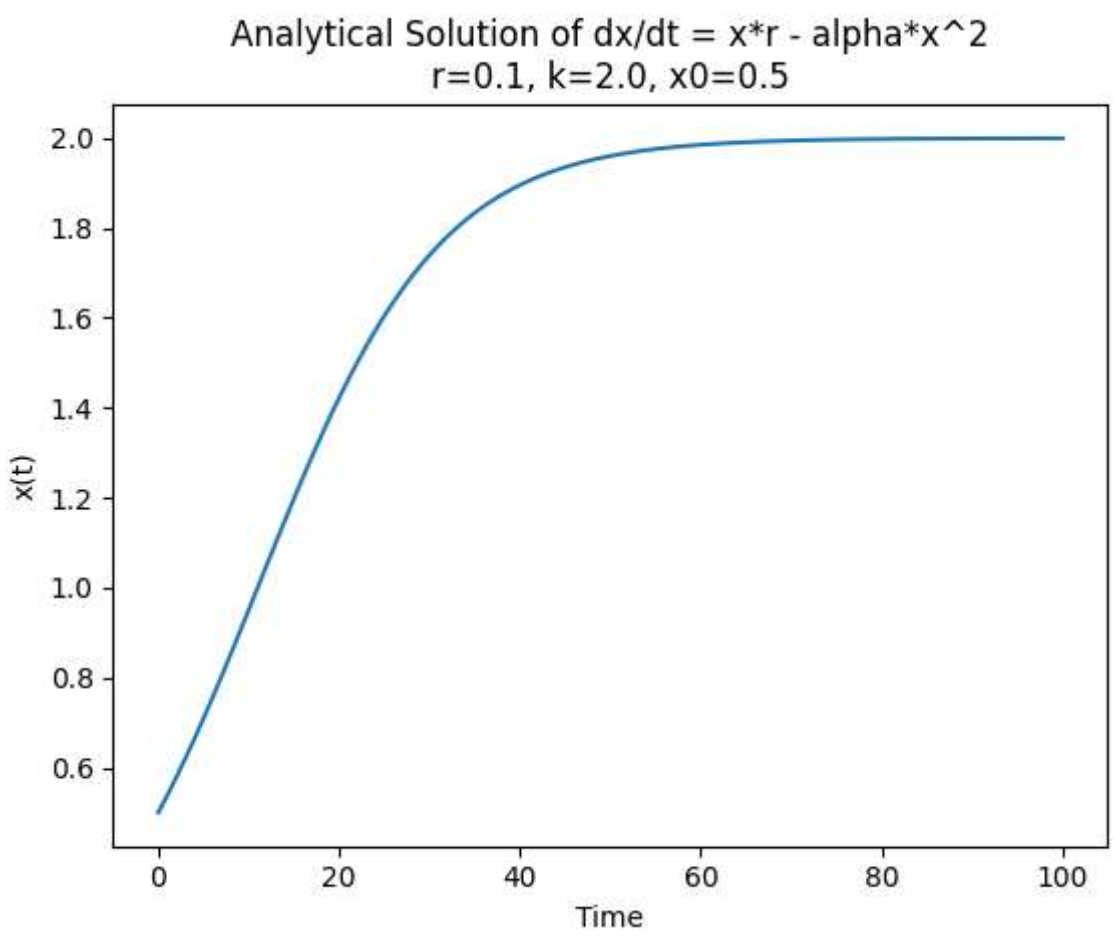

Numerical solutions x(t) of the
Lotka–Volterra model using the
explicit Euler integration scheme
with r=0.1, x0=0.1, k=2
Deltat = 0.05
Deltat = 5
N(t)
t

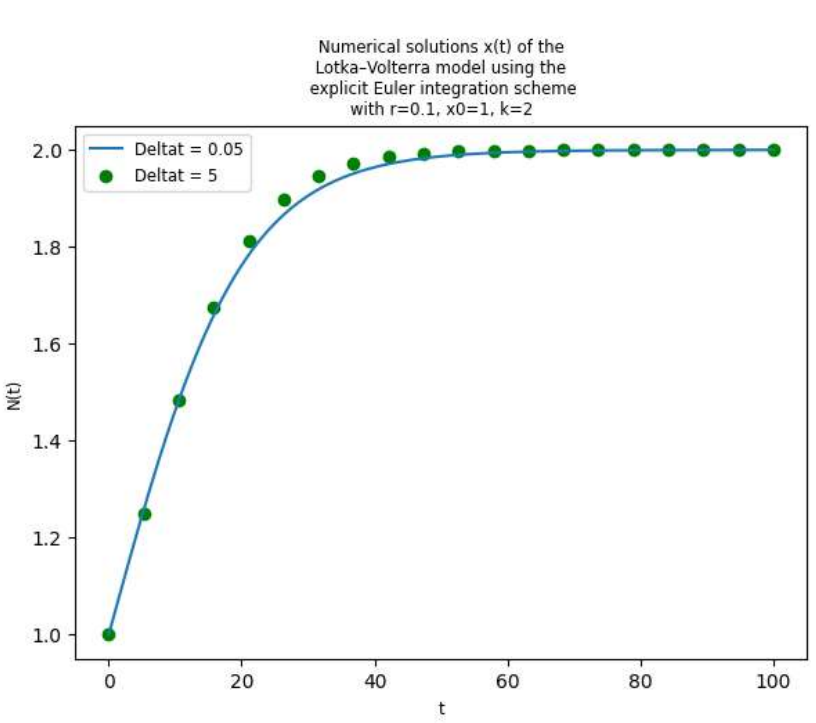

Numerical solutions x(t) of
the Lotka–Volterra model using the
implicit Euler integration scheme
with r=0.1, x0=0.1, k=2
Deltat = 0.05
Deltat = 5
N(t)
t

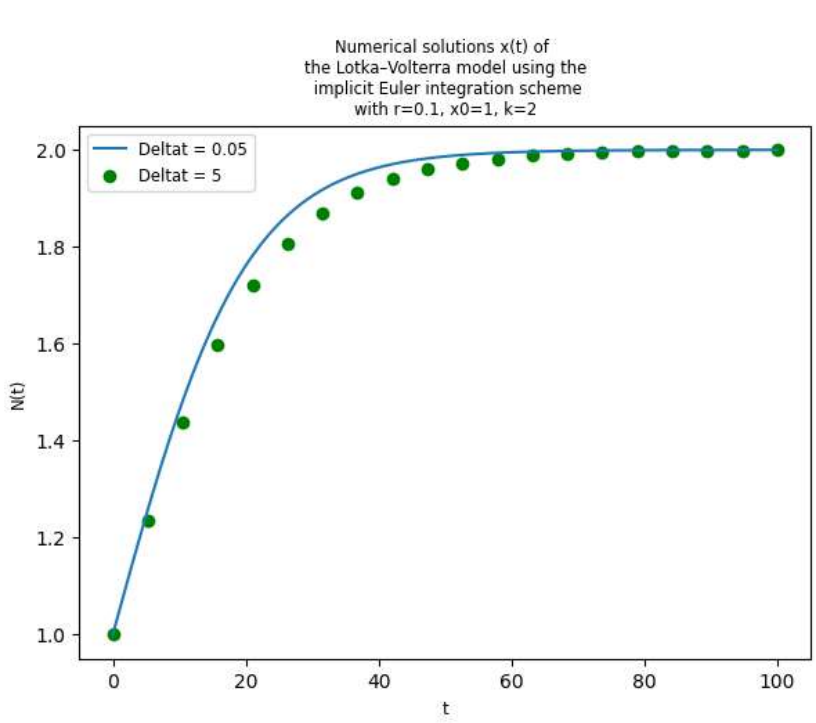

Analytical Solution of dx/dt = x*r - alpha*x^2
r=0.1, k=2.0, x0=0.1
x(t)
Time

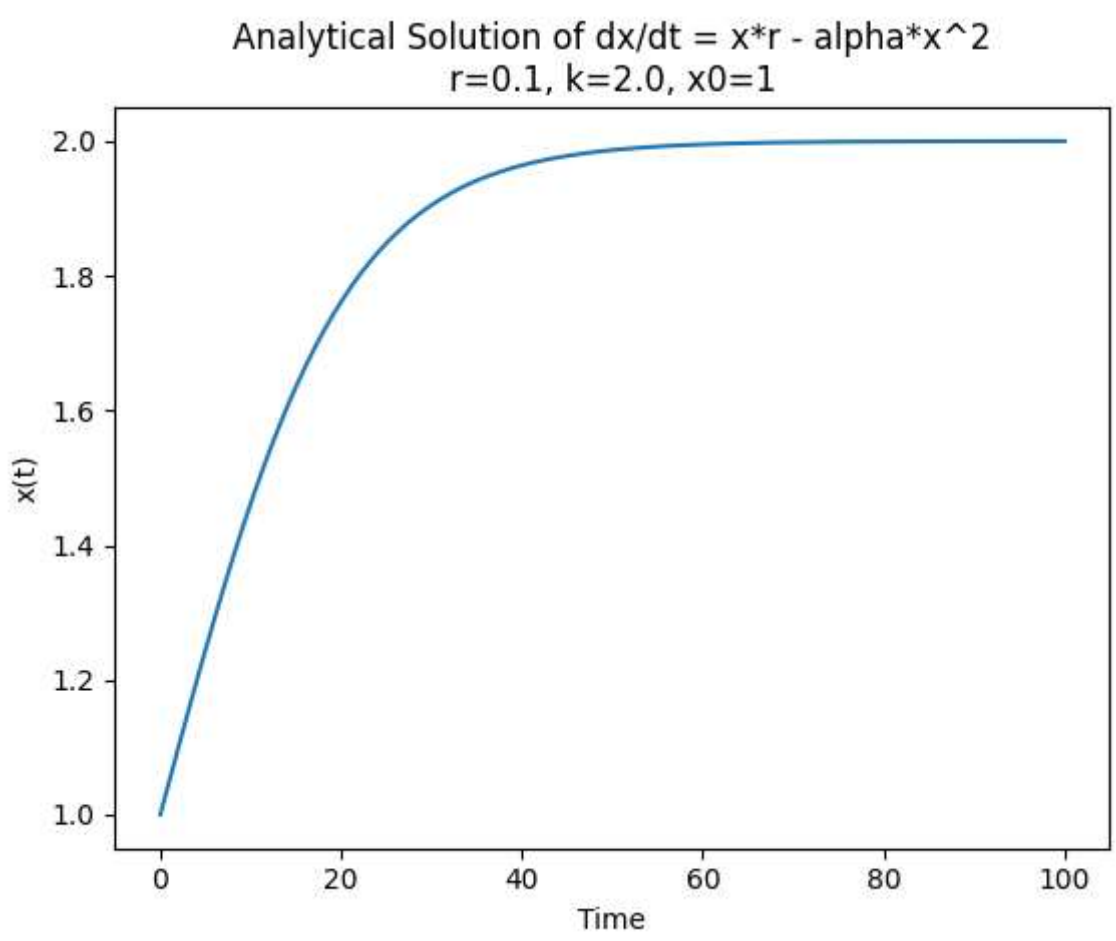

Numerical solutions x(t) of the
Lotka–Volterra model using the
explicit Euler integration scheme
with r=0.1, x0=0.5, k=2
Deltat = 0.05
Deltat = 5
N(t)
t

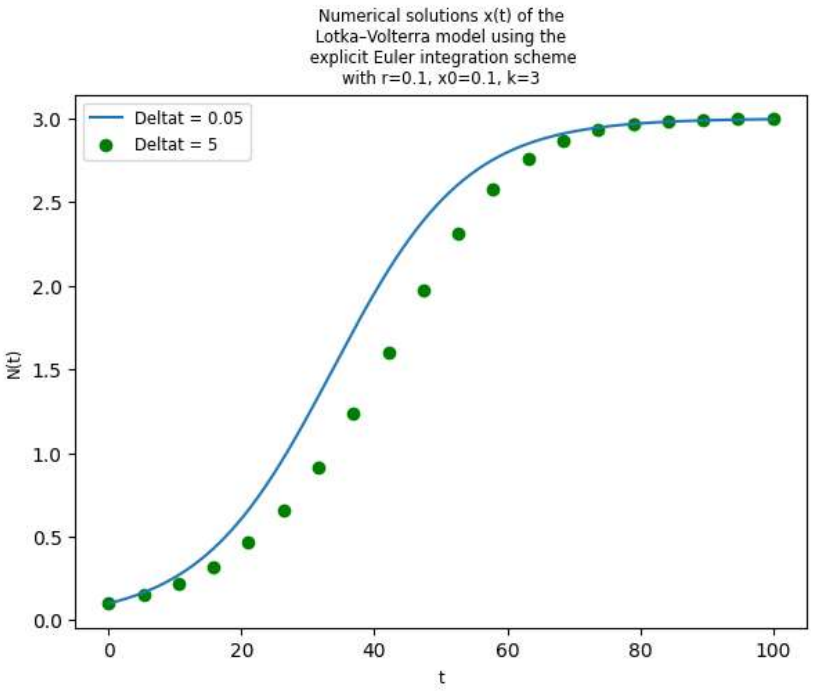

Numerical solutions x(t) of
the Lotka–Volterra model using the
implicit Euler integration scheme
with r=0.1, x0=0.5, k=2
Deltat = 0.05
Deltat = 5
N(t)
t

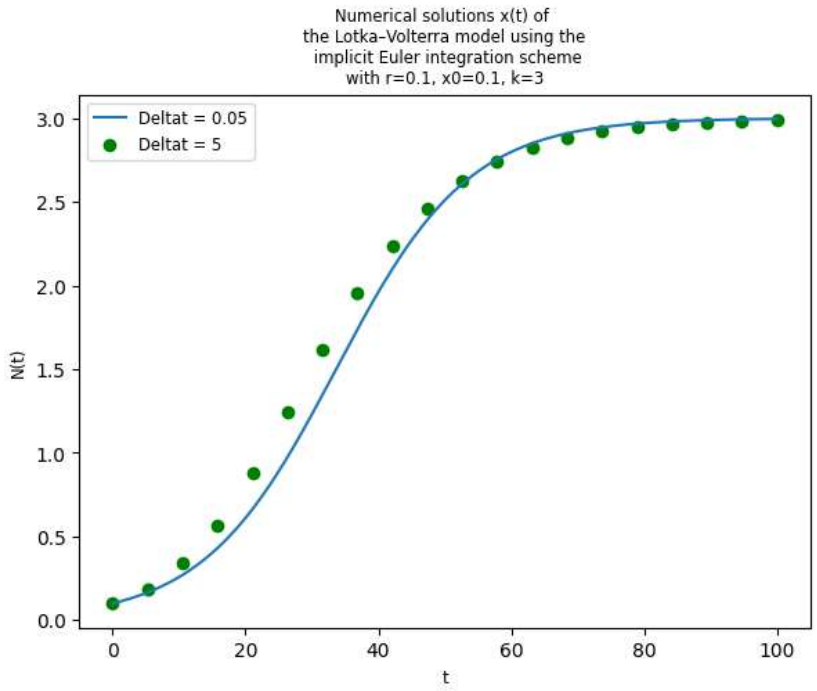

Analytical Solution of dx/dt = x*r - alpha*x^2
r=0.1, k=2.0, x0=0.5
x(t)
Time
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0
20
40
60
80
100

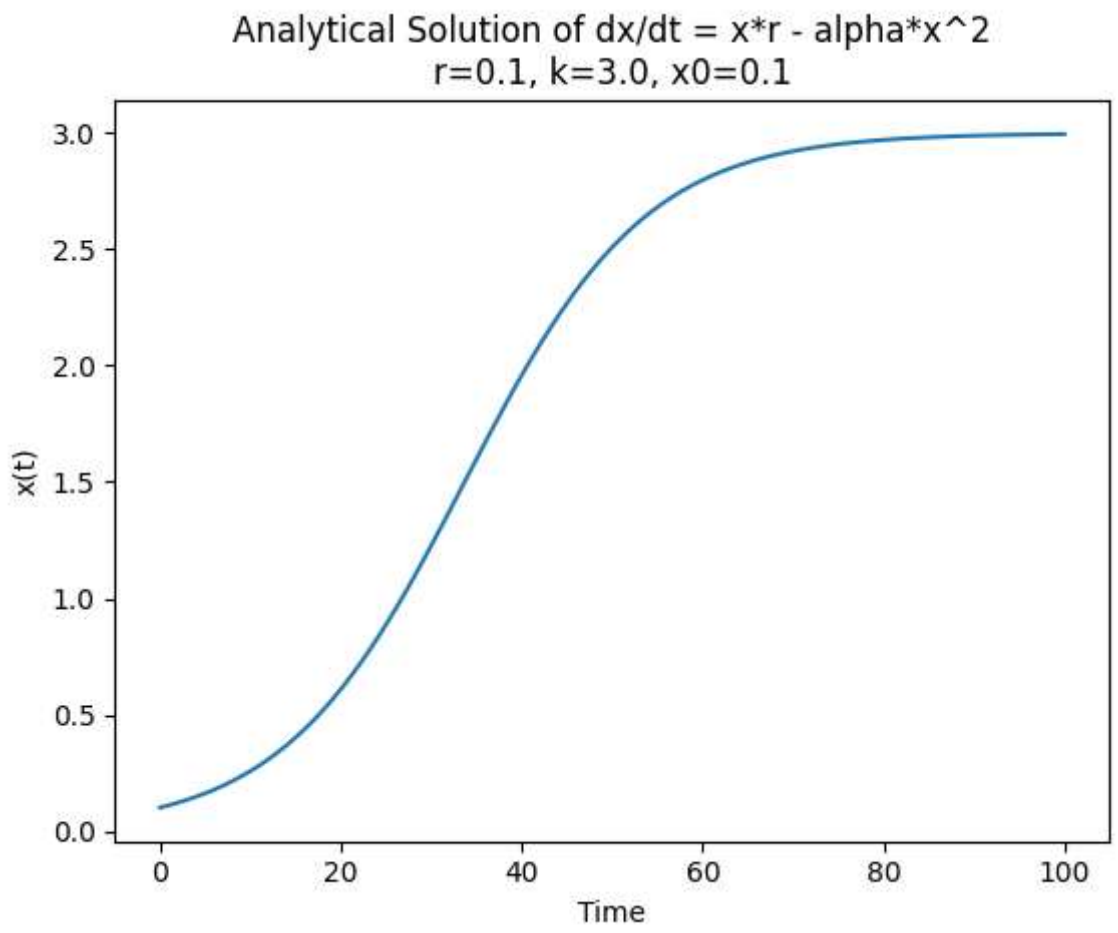

Numerical solutions x(t) of the
Lotka–Volterra model using the
explicit Euler integration scheme
with r=0.1, x0=1, k=2
Deltat = 0.05
Deltat = 5
N(t)
t
2.0
1.8
1.6
1.4
1.2
1.0
0
20
40
60
80
100

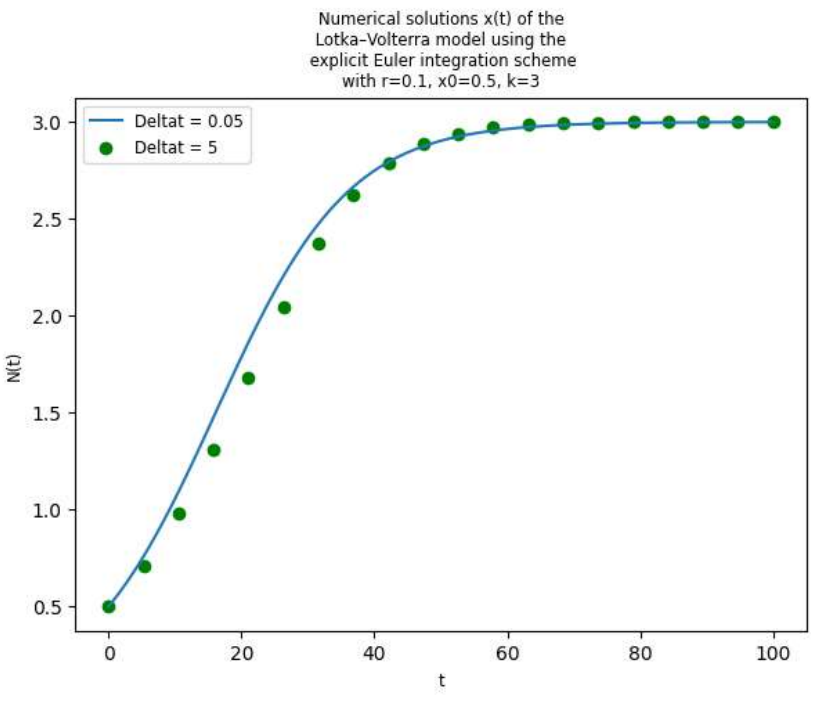

Numerical solutions x(t) of
the Lotka–Volterra model using the
implicit Euler integration scheme
with r=0.1, x0=1, k=2
Deltat = 0.05
Deltat = 5
N(t)
t
2.0
1.8
1.6
1.4
1.2
1.0
0
20
40
60
80
100

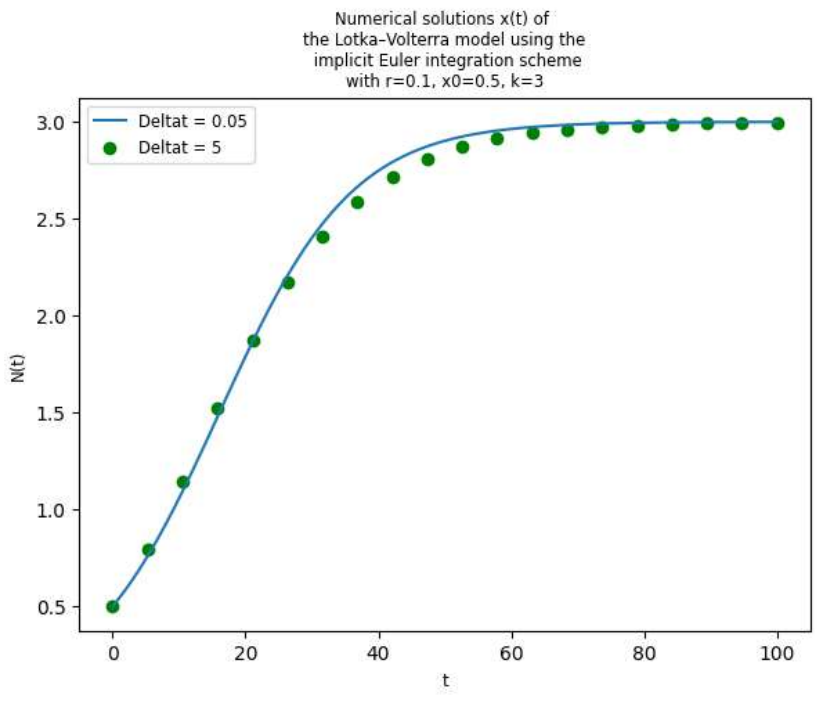

Analytical Solution of dx/dt = x*r - alpha*x^2
r=0.1, k=2.0, x0=1
x(t)
Time
2.0
1.8
1.6
1.4
1.2
1.0
0
20
40
60
80
100

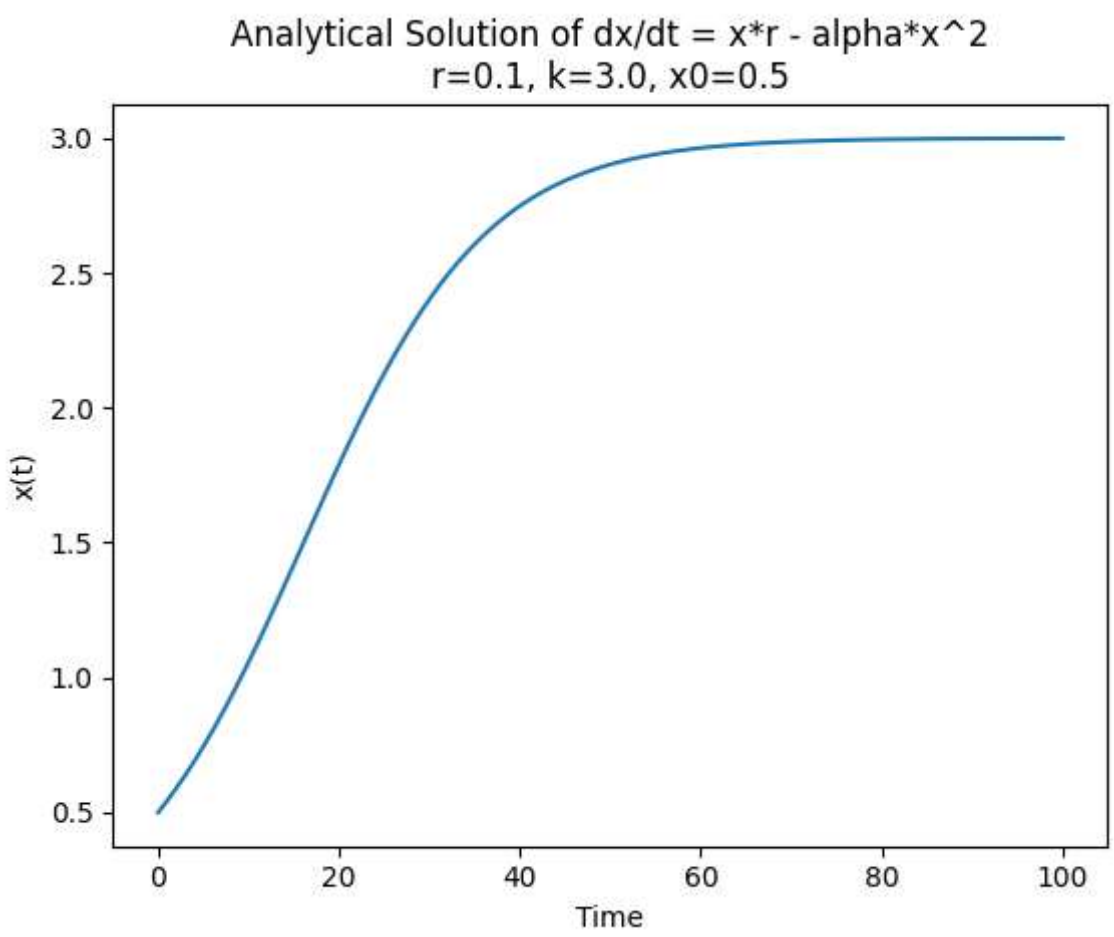

Numerical solutions x(t) of the
Lotka–Volterra model using the
explicit Euler integration scheme
with r=0.1, x0=0.1, k=3
Deltat = 0.05
Deltat = 5
N(t)
t

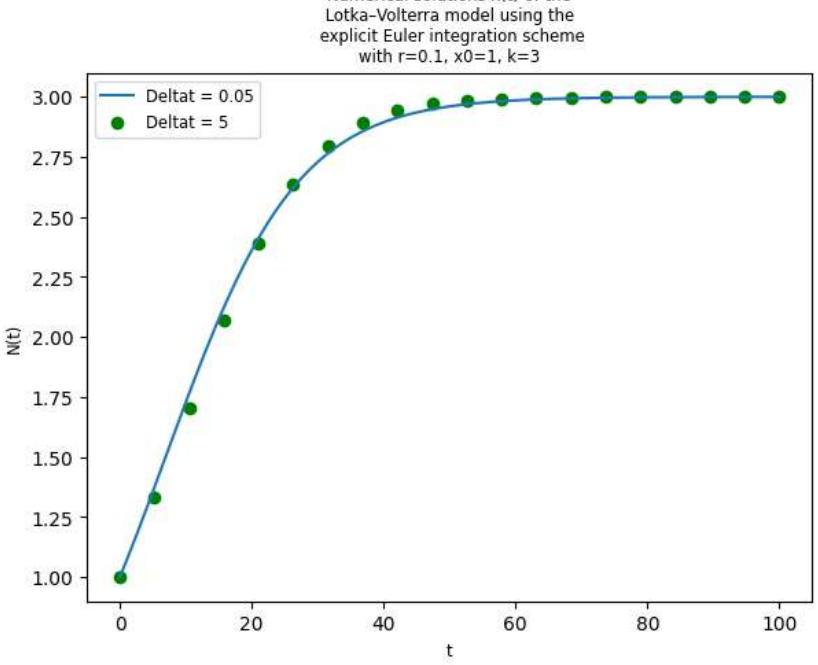

Numerical solutions x(t) of
the Lotka–Volterra model using the
implicit Euler integration scheme
with r=0.1, x0=0.1, k=3
Deltat = 0.05
Deltat = 5
N(t)
t

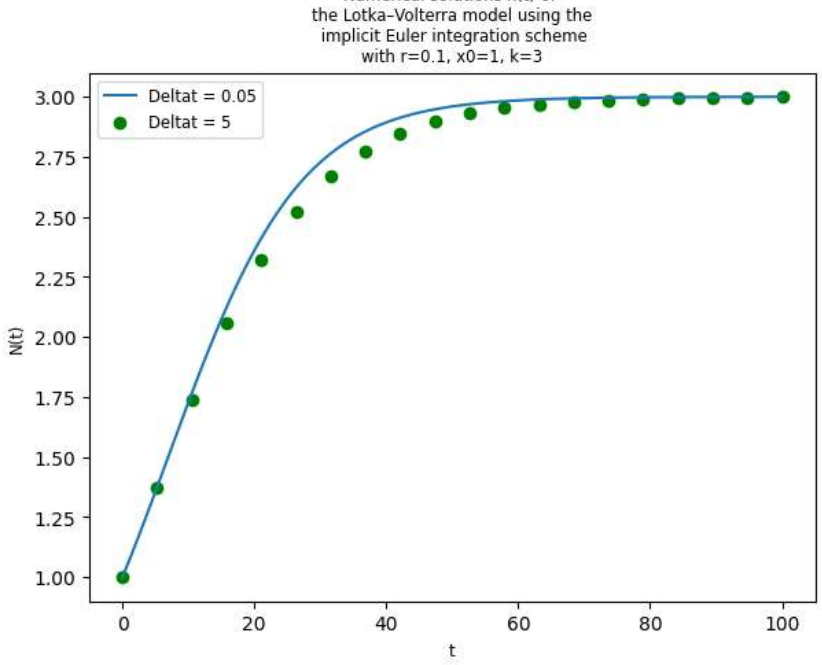

Analytical Solution of dx/dt = x*r - alpha*x^2
r=0.1, k=3.0, x0=0.1
x(t)
Time

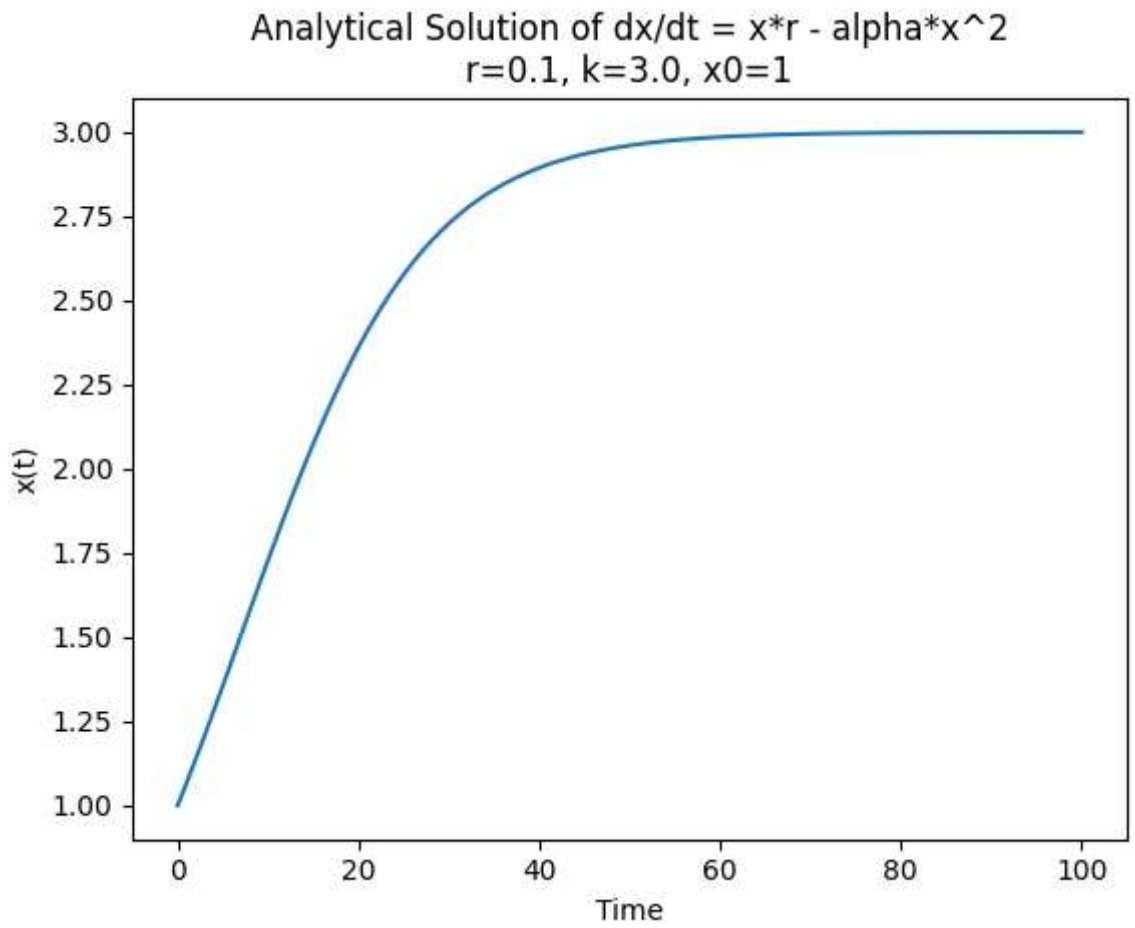

Numerical solutions x(t) of the
Lotka–Volterra model using the
explicit Euler integration scheme
with r=0.1, x0=0.5, k=3
Deltat = 0.05
Deltat = 5
N(t)
t

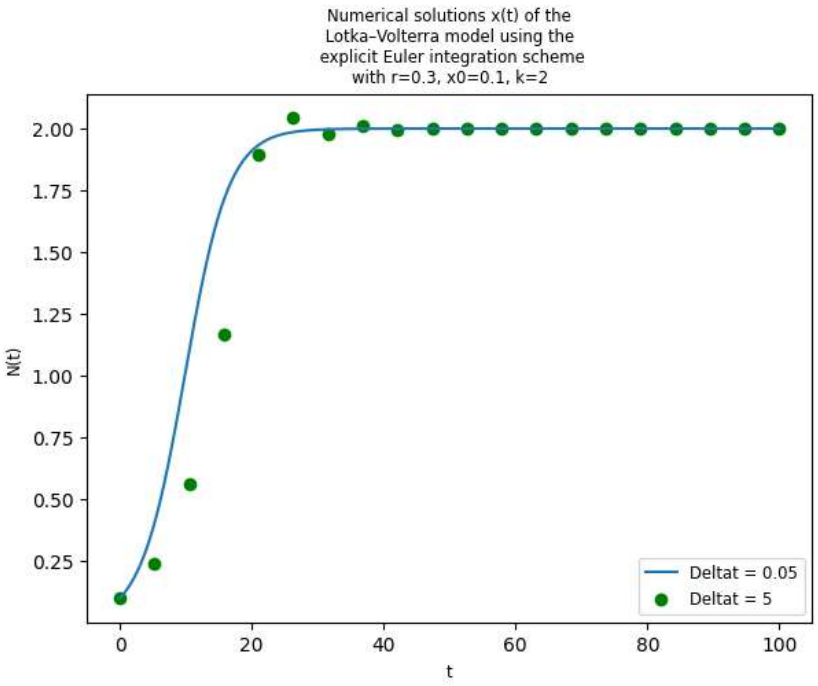

Numerical solutions x(t) of
the Lotka–Volterra model using the
implicit Euler integration scheme
with r=0.1, x0=0.5, k=3
Deltat = 0.05
Deltat = 5
N(t)
t

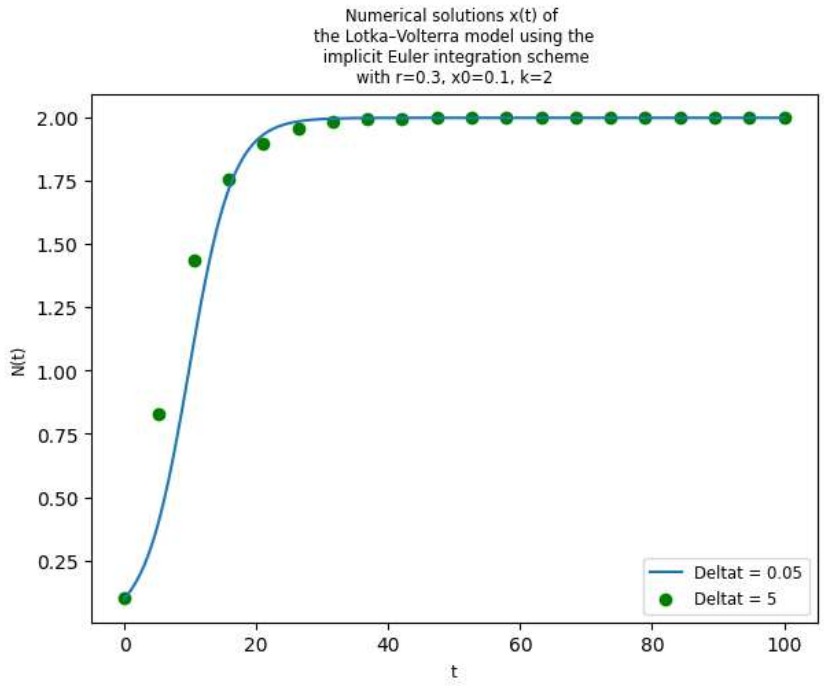

Analytical Solution of dx/dt = x*r - alpha*x^2
r=0.1, k=3.0, x0=0.5
x(t)
Time
3.0
2.5
2.0
1.5
1.0
0.5
0
20
40
60
80
100

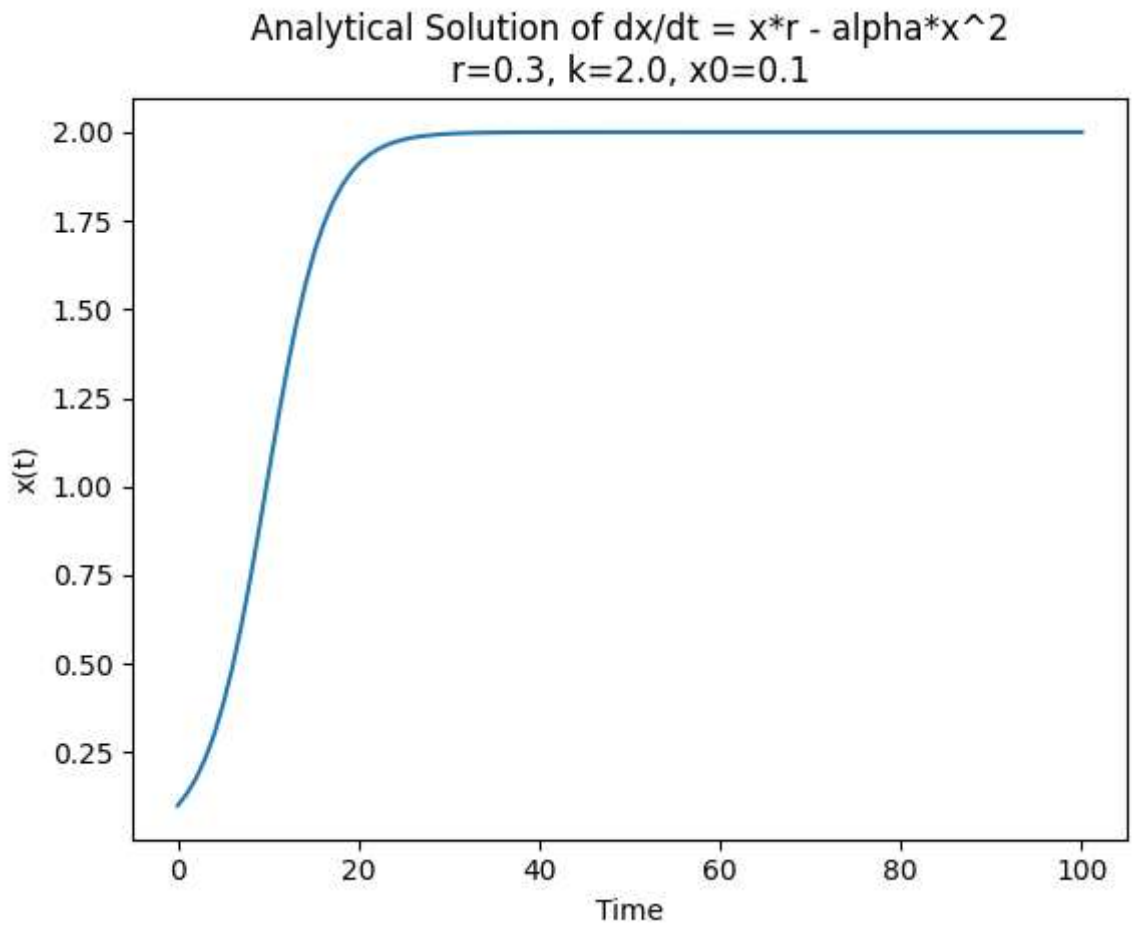

Numerical solutions x(t) of the
Lotka–Volterra model using the
explicit Euler integration scheme
with r=0.1, x0=1, k=3
Deltat = 0.05
Deltat = 5
N(t)
t

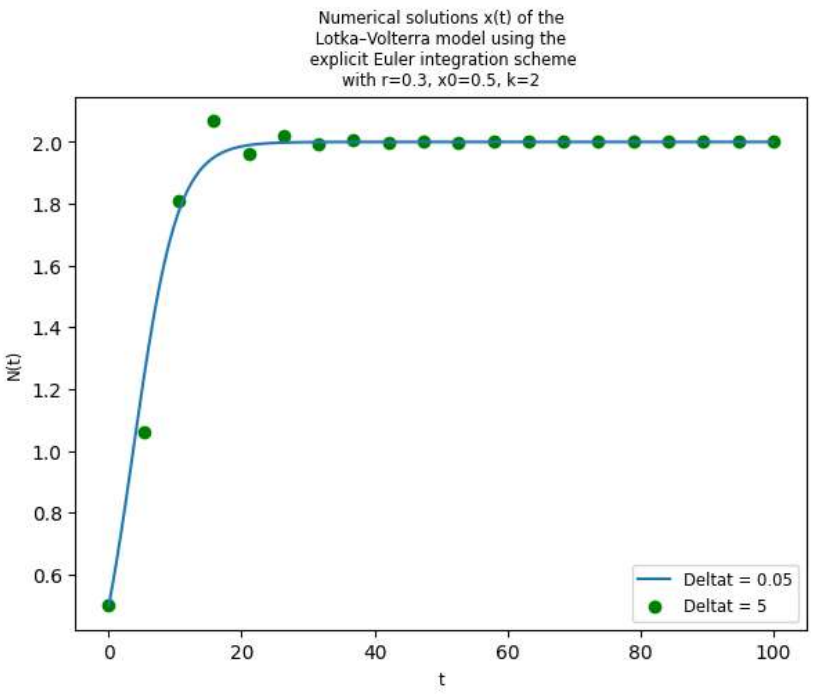

Numerical solutions x(t) of
the Lotka–Volterra model using the
implicit Euler integration scheme
with r=0.1, x0=1, k=3
Deltat = 0.05
Deltat = 5
N(t)
t

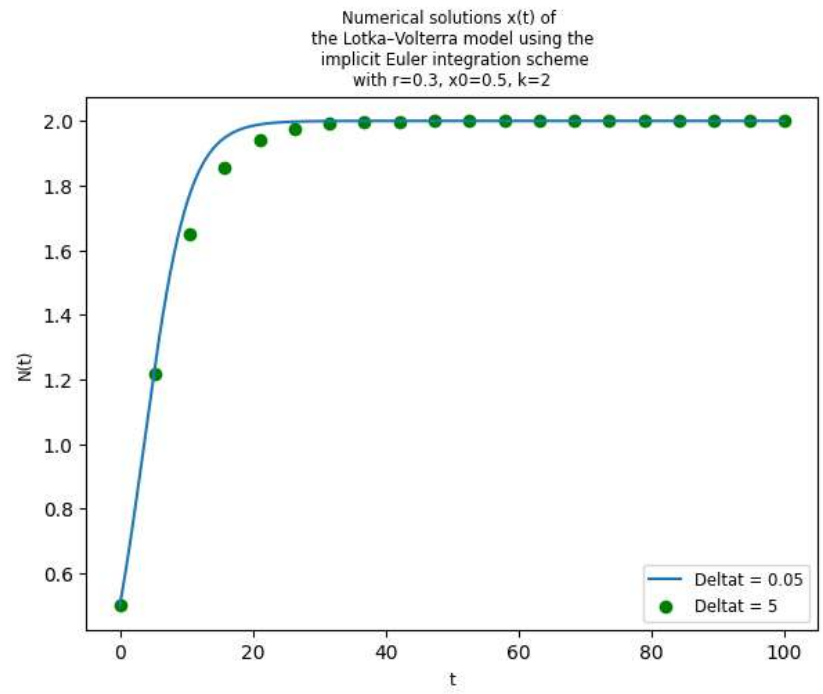

Analytical Solution of dx/dt = x*r - alpha*x^2
r=0.1, k=3.0, x0=1
x(t)
Time
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0
20
40
60
80
100

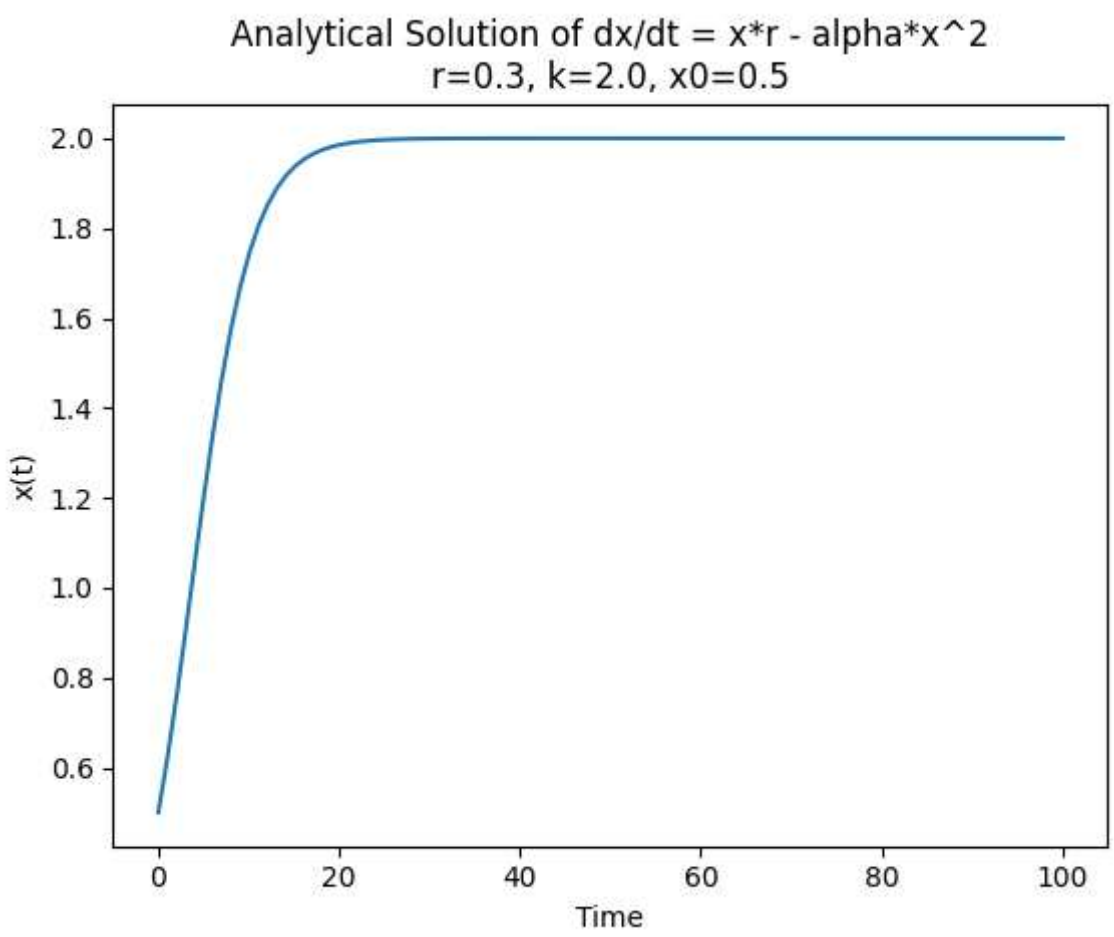

Numerical solutions x(t) of the
Lotka–Volterra model using the
explicit Euler integration scheme
with r=0.3, x0=0.1, k=2
N(t)
t
Deltat = 0.05
Deltat = 5

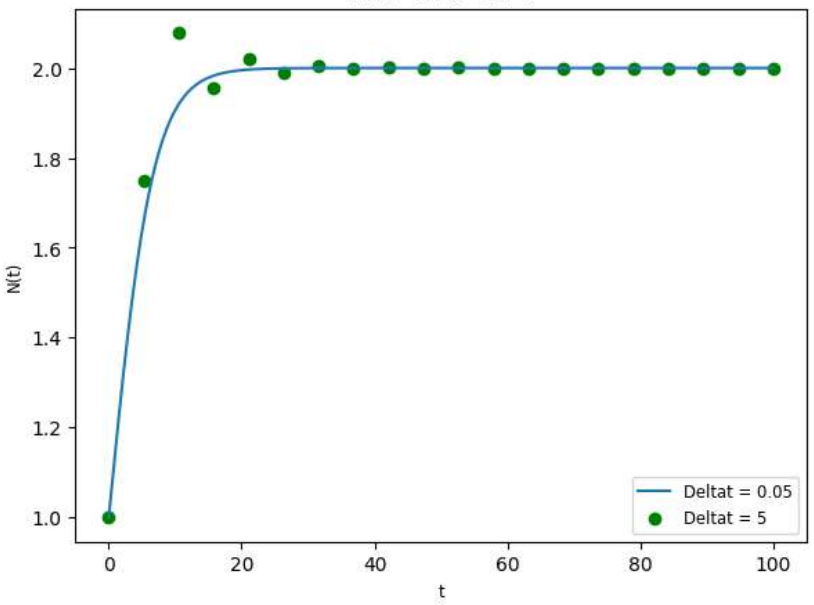

Numerical solutions x(t) of
the Lotka–Volterra model using the
implicit Euler integration scheme
with r=0.3, x0=0.1, k=2
N(t)
t
Deltat = 0.05
Deltat = 5

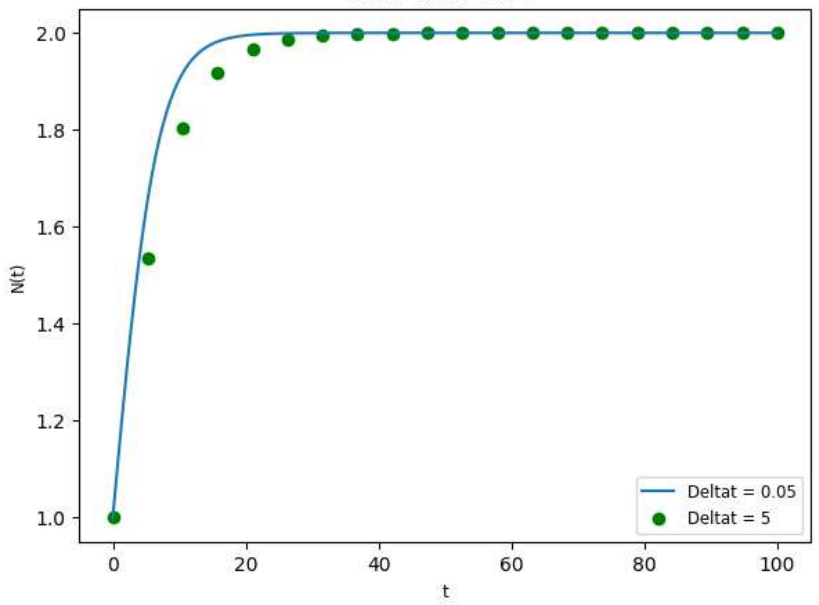

Analytical Solution of dx/dt = x*r - alpha*x^2
r=0.3, k=2.0, x0=0.1
x(t)
Time

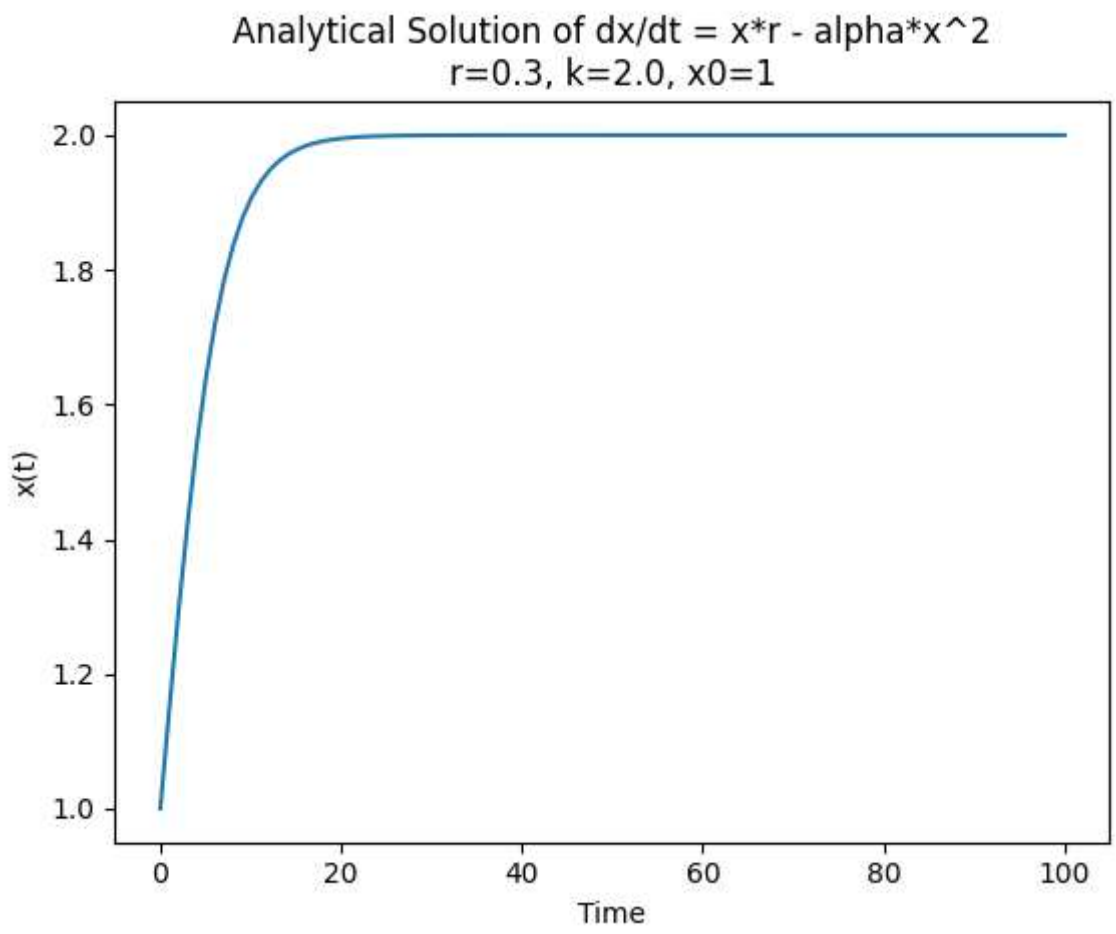

Numerical solutions x(t) of the
Lotka–Volterra model using the
explicit Euler integration scheme
with r=0.3, x0=0.5, k=2
N(t)
t
Deltat = 0.05
Deltat = 5

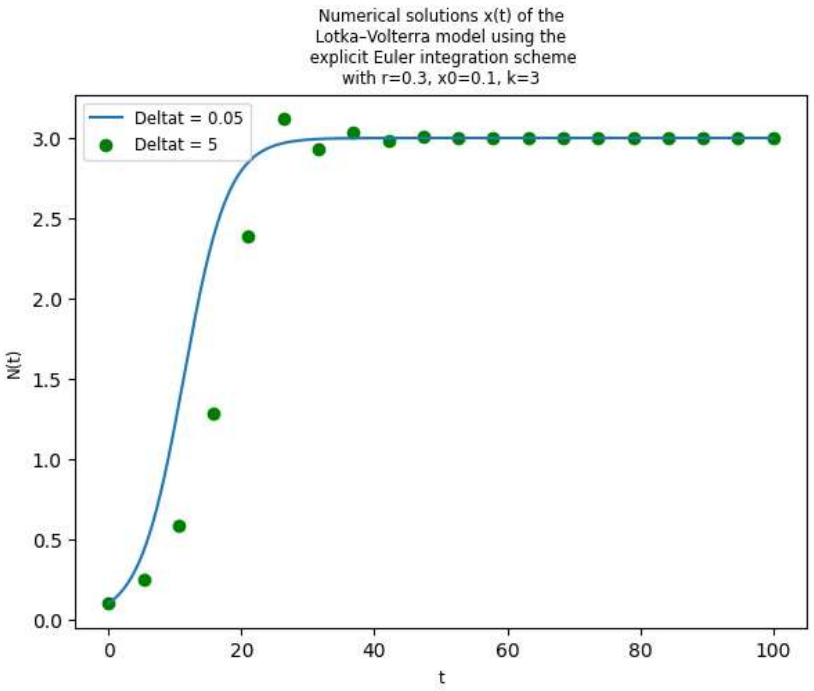

Numerical solutions x(t) of
the Lotka–Volterra model using the
implicit Euler integration scheme
with r=0.3, x0=0.5, k=2
N(t)
t
Deltat = 0.05
Deltat = 5

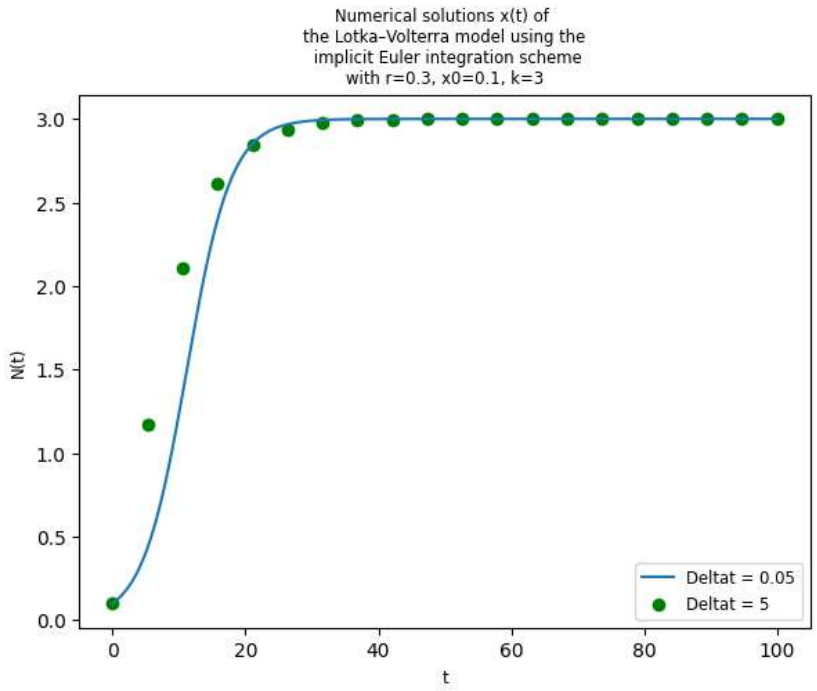

Analytical Solution of dx/dt = x*r - alpha*x^2
r=0.3, k=2.0, x0=0.5
x(t)
Time
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0
20
40
60
80
100

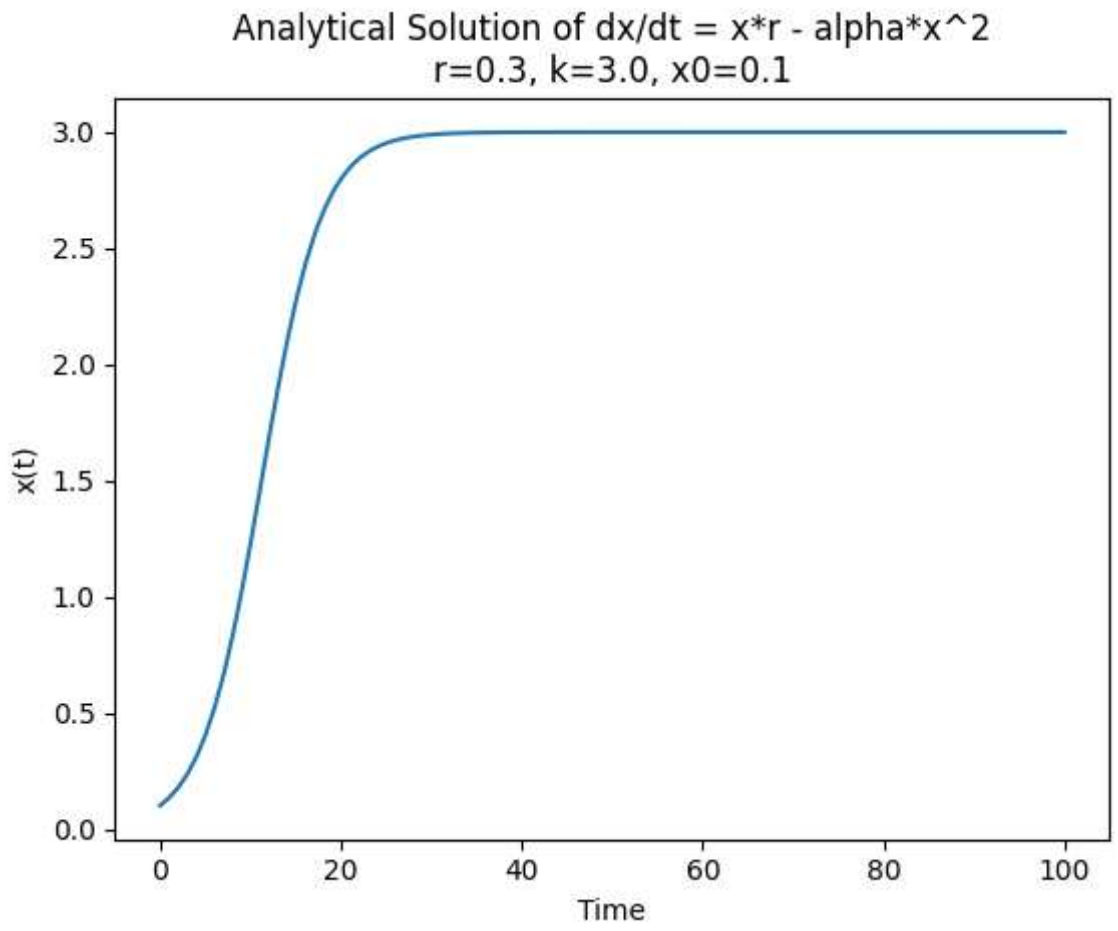

Numerical solutions x(t) of the
Lotka–Volterra model using the
explicit Euler integration scheme
with r=0.3, x0=1, k=2
N(t)
t
Deltat = 0.05
Deltat = 5

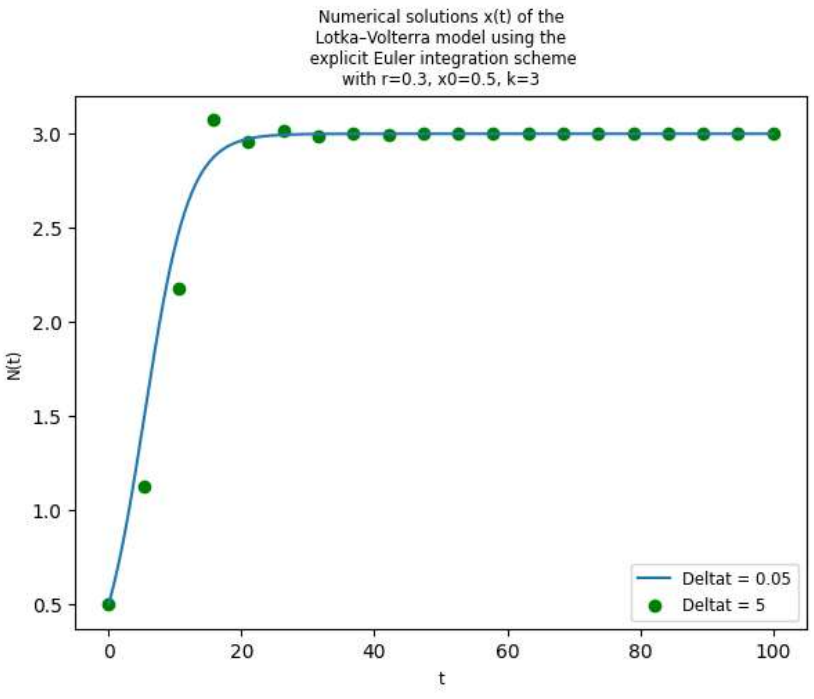

Numerical solutions x(t) of
the Lotka–Volterra model using the
implicit Euler integration scheme
with r=0.3, x0=1, k=2
N(t)
t
Deltat = 0.05
Deltat = 5

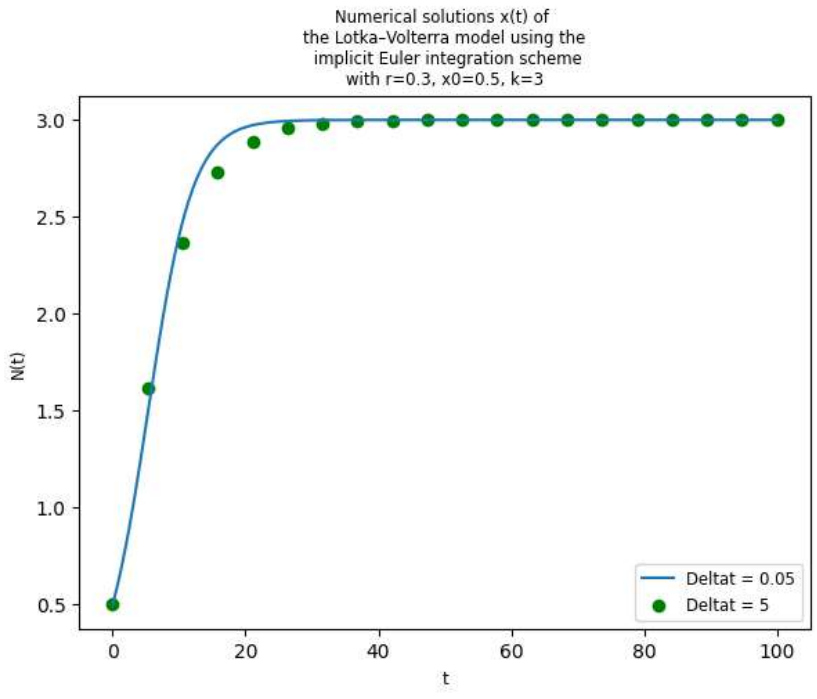

Analytical Solution of dx/dt = x*r - alpha*x^2
r=0.3, k=2.0, x0=1
x(t)
Time
2.0
1.8
1.6
1.4
1.2
1.0
0
20
40
60
80
100

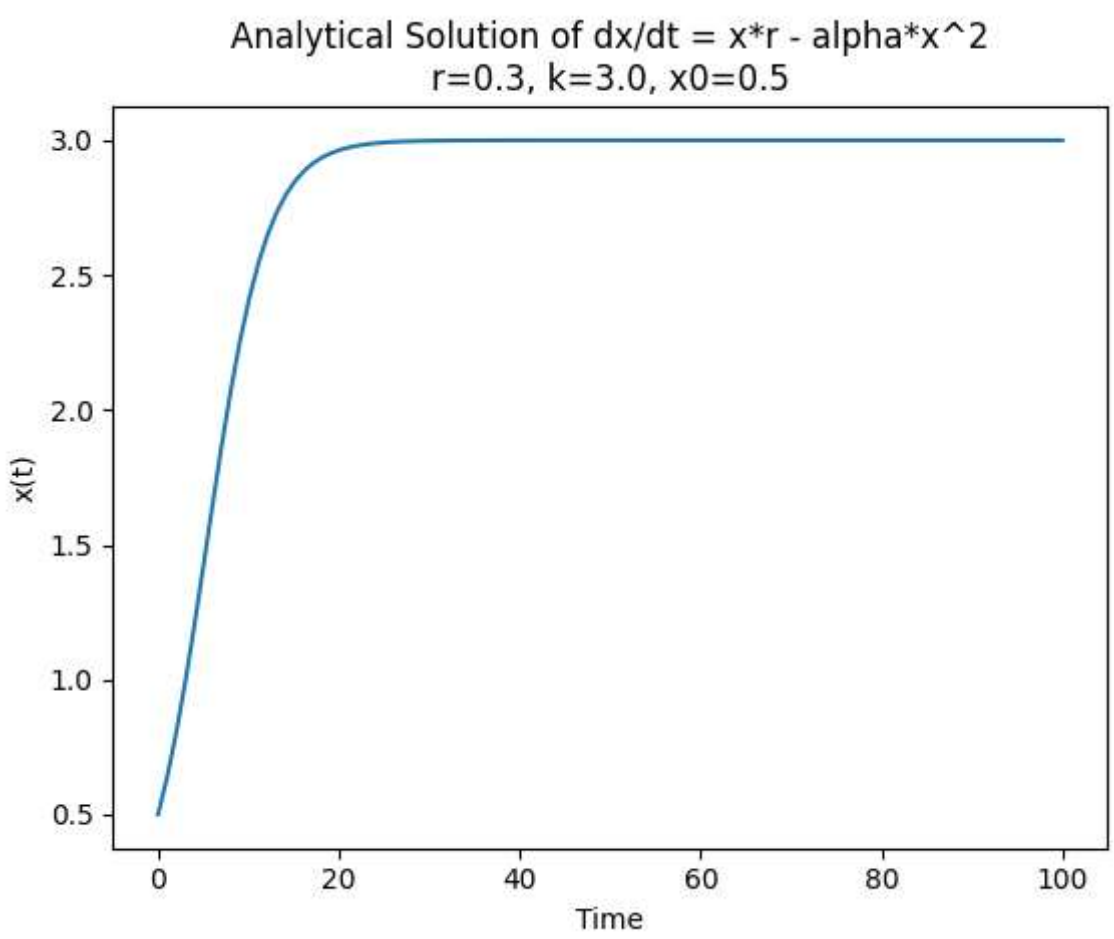

Numerical solutions x(t) of the
Lotka–Volterra model using the
explicit Euler integration scheme
with r=0.3, x0=0.1, k=3
Deltat = 0.05
Deltat = 5
N(t)
t

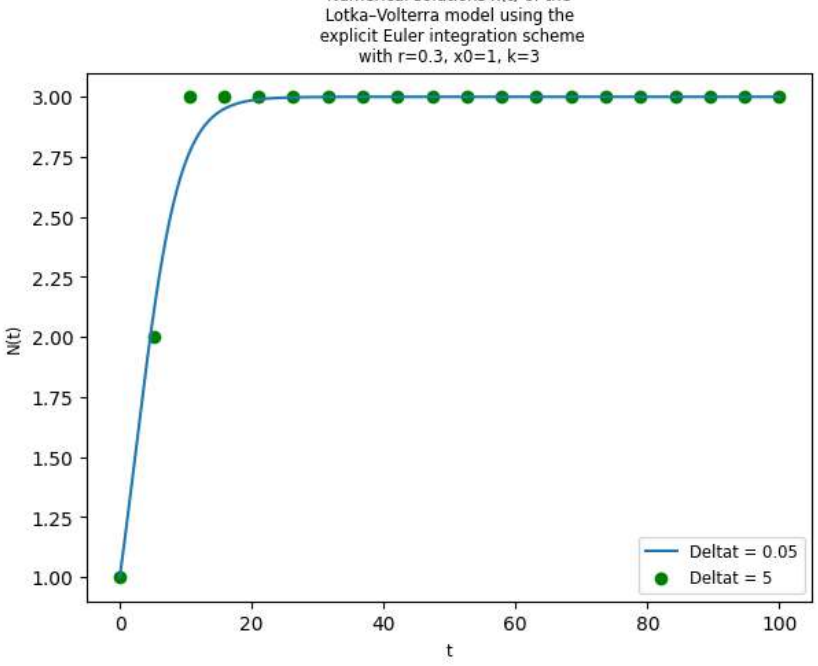

Numerical solutions x(t) of
the Lotka–Volterra model using the
implicit Euler integration scheme
with r=0.3, x0=0.1, k=3
Deltat = 0.05
Deltat = 5
N(t)
t

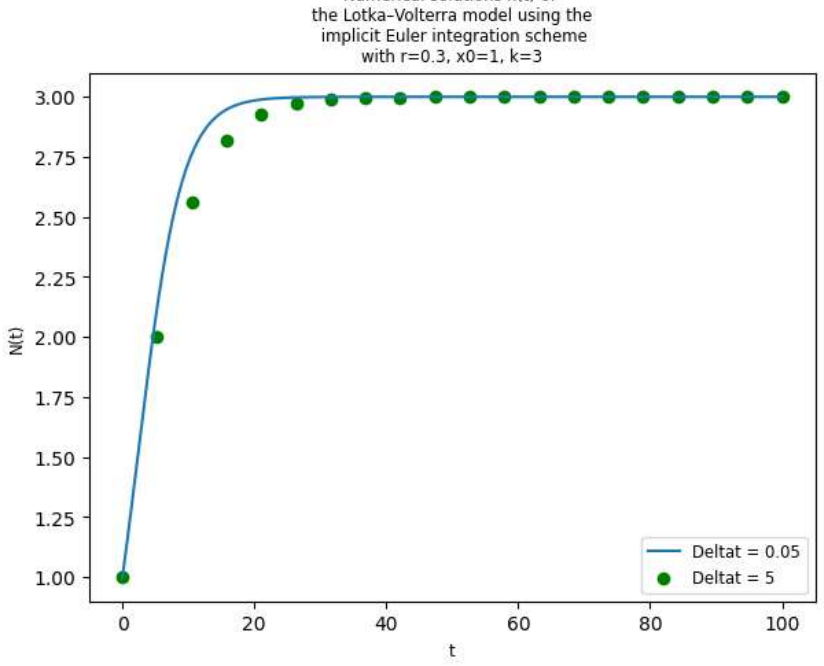

Analytical Solution of dx/dt = x*r - alpha*x^2
r=0.3, k=3.0, x0=0.1
x(t)
Time

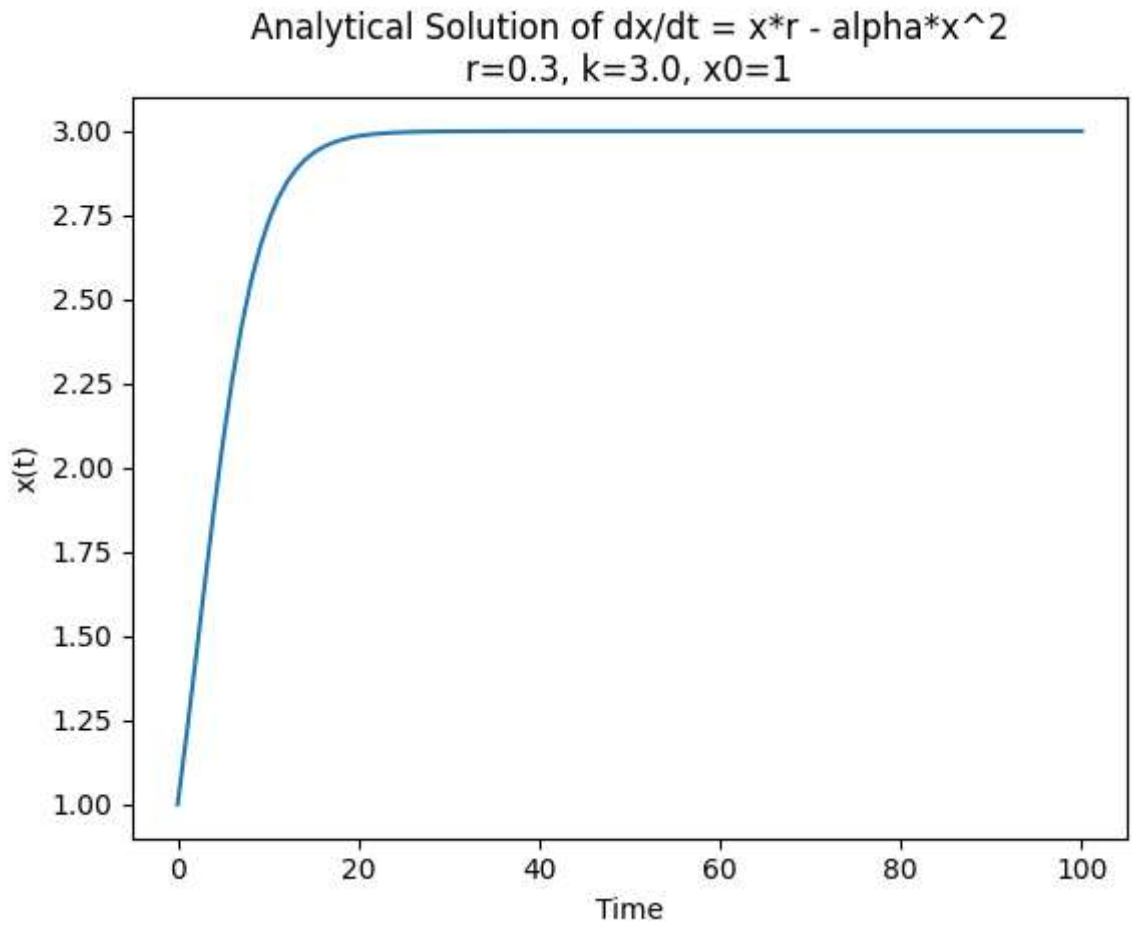

Numerical solutions x(t) of the
Lotka–Volterra model using the
explicit Euler integration scheme
with r=0.3, x0=0.5, k=3
Deltat = 0.05
Deltat = 5
N(t)
t


Numerical solutions x(t) of
the Lotka–Volterra model using the
implicit Euler integration scheme
with r=0.3, x0=0.5, k=3
Deltat = 0.05
Deltat = 5
N(t)
t


Analytical Solution of dx/dt = x*r - alpha*x^2
r=0.3, k=3.0, x0=0.5
x(t)
Time
3.0
2.5
2.0
1.5
1.0
0.5
0
20
40
60
80
100


Numerical solutions x(t) of the
Lotka–Volterra model using the
explicit Euler integration scheme
with r=0.3, x0=1, k=3
N(t)
t
Deltat = 0.05
Deltat = 5


Numerical solutions x(t) of
the Lotka–Volterra model using the
implicit Euler integration scheme
with r=0.3, x0=1, k=3
N(t)
t
Deltat = 0.05
Deltat = 5


Analytical Solution of dx/dt = x*r - alpha*x^2
r=0.3, k=3.0, x0=1
x(t)
Time
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0
20
40
60
80
100

14.7

b) Implicit scheme

$$x_{n+1} = x_n + \left(r\, \tilde{x}_{n+1}\left(1 - \frac{\tilde{x}_{n+1}}{K}\right)\right) dt$$

$$x_{n+1} - \frac{r\, x_{n+1}(K)\,dt + r\,(x_n+1)^2\,dt}{K} = x_n$$

$$K\,x_{n+1} - r\,x_{n+1}(K)\,dt + r\,(x_n+1)^2\,dt = x_n K$$

$$r\,(x_n+1)^2\,dt + x_{n+1}\left(K - rK\,dt\right) - x_n K$$

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$\frac{-(K - rK\,dt) \pm \sqrt{(K - r\,dt)^2 + 4\,r\,dt\,x_n K}}{2\,r\,dt}$$

$$x_{n+1} = +K\,\frac{-(1 - r\,dt) \pm \sqrt{(1 - r\,dt)^2 + 4r\,dt\,\frac{x_n}{K}}}{2\,r\,dt}$$