AP 6

Predator prey model

$$\frac{dx}{dt} = \alpha x - \beta xy$$

$$\frac{dy}{dt} = \delta xy - \gamma y$$

Where:

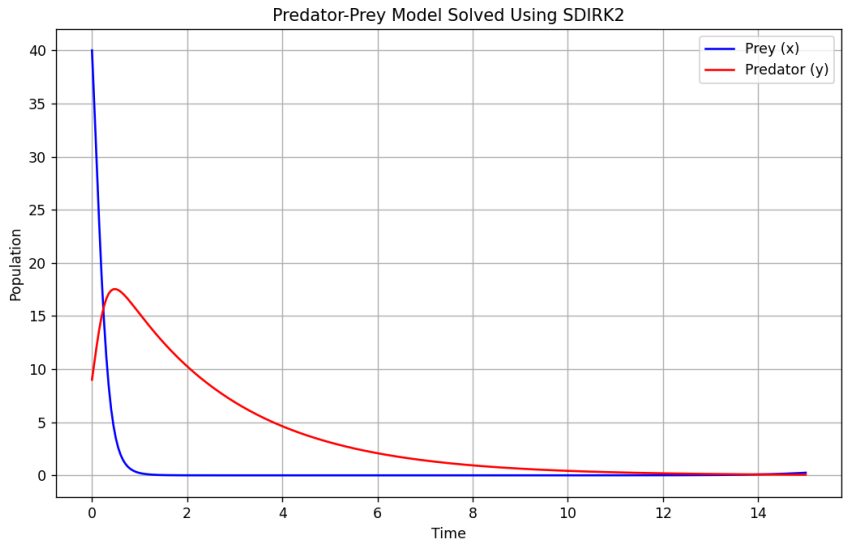
- $x(t)$: prey population (e.g., rabbits)
- $y(t)$: predator population (e.g., foxes)
- $\alpha > 0$: natural growth rate of prey in the absence of predators
- $\beta > 0$: rate of predation proportional to the prey and predator populations
- $\delta > 0$: reproduction rate of predators proportional to the predation rate
- $\gamma > 0$: natural death rate of predators in the absence of prey

This system models the interaction between a prey population and a predator population, leading to oscillatory behavior typical of predator-prey systems.

https://en.wikipedia.org/wiki/List_of_nonlinear_ordinary_differential_equations

Result:

The output demonstrates a stable numerical solution for the predator-prey model using the SDIRK2 method. The plot illustrates the oscillatory nature of the populations and validates that the numerical method is capable of handling the nonlinear dynamics.



Code:

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

# predator-prey model
def predator_prey(t, z, alpha, beta, delta, gamma):
    x, y = z
    dxdt = alpha * x - beta * x * y
    dydt = delta * x * y - gamma * y
    return np.array([dxdt, dydt])

# SDIRK2 method Butcher table
stages = 2
A = np.array([
    [0.2928932188134524, 0],
    [0.7071067811865476, 0.2928932188134524]
])
b = np.array([0.7071067811865476, 0.2928932188134524])
c = np.array([0.2928932188134524, 1.0])

# Parameters for the predator-prey model
alpha = 1.1
beta = 0.4
delta = 0.1
gamma = 0.4

# Initial conditions and time span
x0, y0 = 40, 9
z0 = np.array([x0, y0])
T = 15
N = 300
t = np.linspace(0, T, N)
h = T / N

# Initialize solution arrays
solution = np.zeros((N, 2))
solution[0] = z0

# DIRK solver implementation
for n in range(N - 1):
    z_n = solution[n]
```

```
    stages_values = np.zeros((stages, 2))
    for i in range(stages):
        stage = z_n.copy()
        for _ in range(10):
            sum_a = sum(A[i, j] * predator_prey(t[n] + c[j] * h,
stages_values[j], alpha, beta, delta, gamma) for j in range(i))
            stage = z_n + h * sum_a + h * A[i, i] * predator_prey(t[n] + c[i] *
h, stage, alpha, beta, delta, gamma)
        stages_values[i] = stage

    sum_b = sum(b[j] * predator_prey(t[n] + c[j] * h, stages_values[j], alpha,
beta, delta, gamma) for j in range(stages))
    solution[n + 1] = z_n + h * sum_b

# Plot the results
plt.figure(figsize=(10, 6))
plt.plot(t, solution[:, 0], label="Prey (x)", color="blue")
plt.plot(t, solution[:, 1], label="Predator (y)", color="red")
plt.title("Predator-Prey Model Solved Using SDIRK2")
plt.xlabel("Time")
plt.ylabel("Population")
plt.legend()
plt.grid()
plt.show()
```