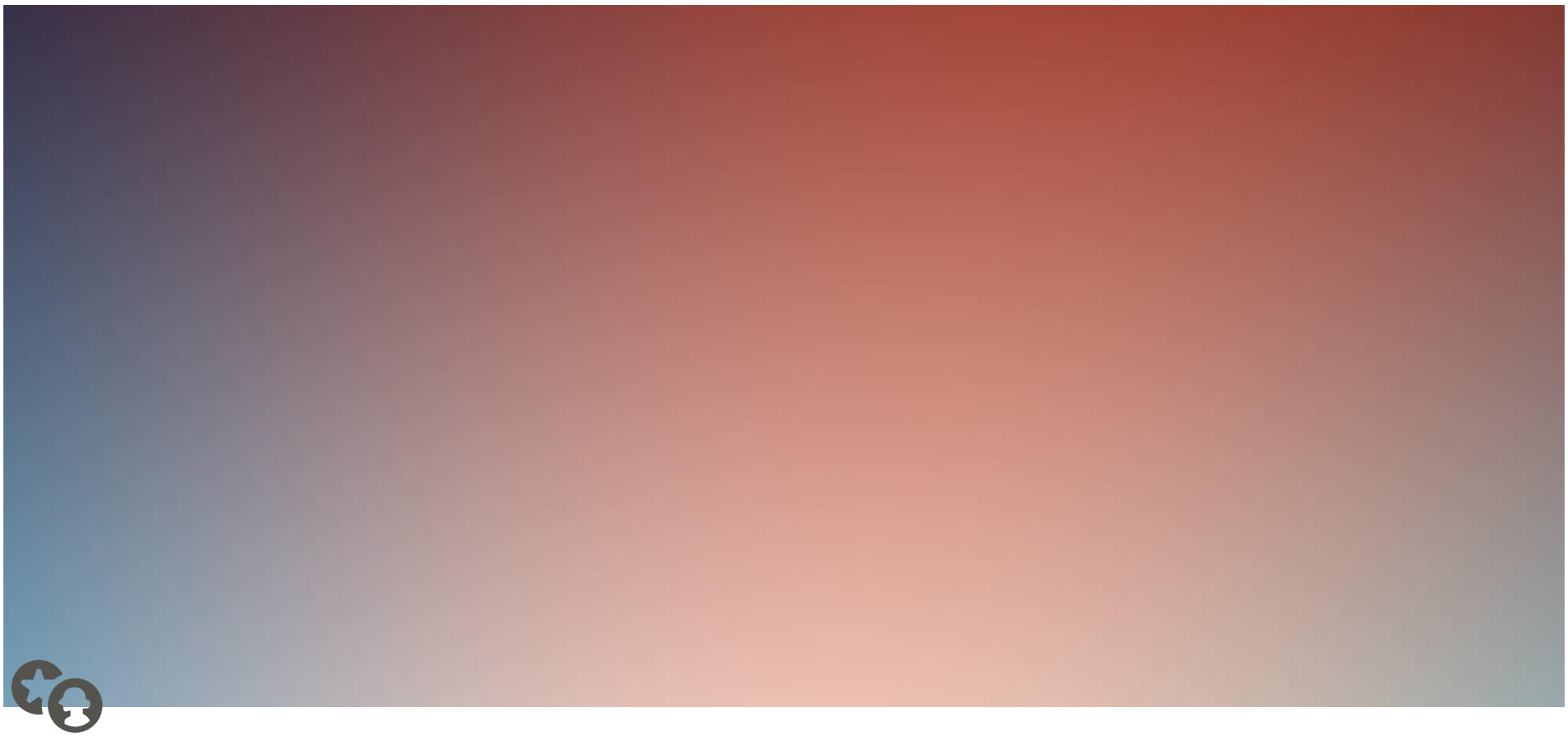


Investment Growth Model

Introduction

In this model, the growth of an investment is modeled with a nonlinear return rate. The return on investment is assumed to be a function of the current value of the investment, leading to a nonlinear equation.

Model

Let ***X(t)*** represent the amount of money invested at time ***t***, and let ***Y(t)*** represent the rate of return on the investment, which is assumed to depend on ***X(t***).

$$\frac{dX}{dt} = rX(t)\left(1 - \frac{X(t)}{K}\right) \qquad \frac{dY}{dt} = \alpha Y(t)\left(1 - \frac{Y(t)}{C}\right)$$

- ***X(t)*** - is the investment value at time ***t***.
- ***Y(t)*** **-** is the rate of return on the investment at time ***t***.
- ***r*** - is the base growth rate.
- ***K*** - is a constant that represents the maximum potential of the investment value.
- ***α*** - is the constant growth rate of the return.
- ***C*** - is a constant that represents a saturation point for the return.

Numerical Solution

The **numerical solution** of a system of ODEs involves approximating the solution over discrete time steps because an analytical solution may not be available or practical.

In this case, the **DIRK (Diagonal Implicit Runge-Kutta)** method is used to solve the system of two nonlinear ODEs:

1. The method breaks the time interval into small steps (with size Δt) and iteratively calculates the state of the system at each time step.
2. It computes the solution in stages, using intermediate values that depend on the previous ones (implicit method).
3. The **DIRK(2,1)** method uses two stages per step, with a weighted average of the intermediate results to update the solution.

This process is repeated until the end of the time span, resulting in an approximation of how the variables (investment and return rate) evolve over time.

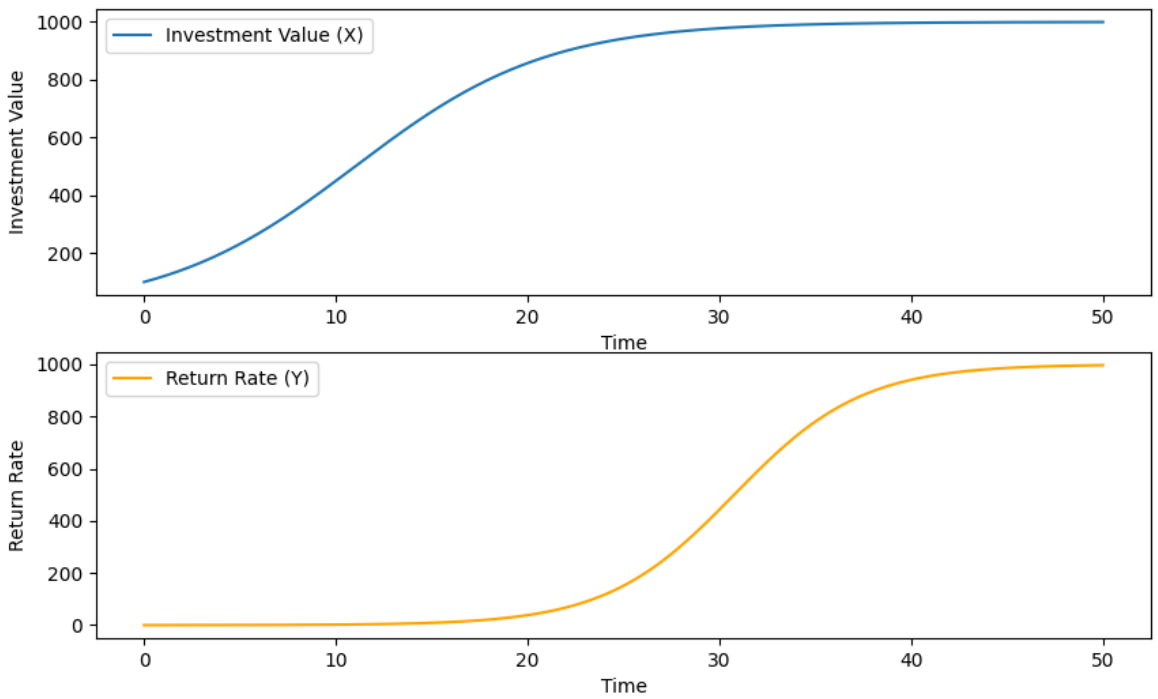
The solution is then visualized by plotting the investment value ***X(t***) and return rate ***Y(t)*** against time.

Implementation

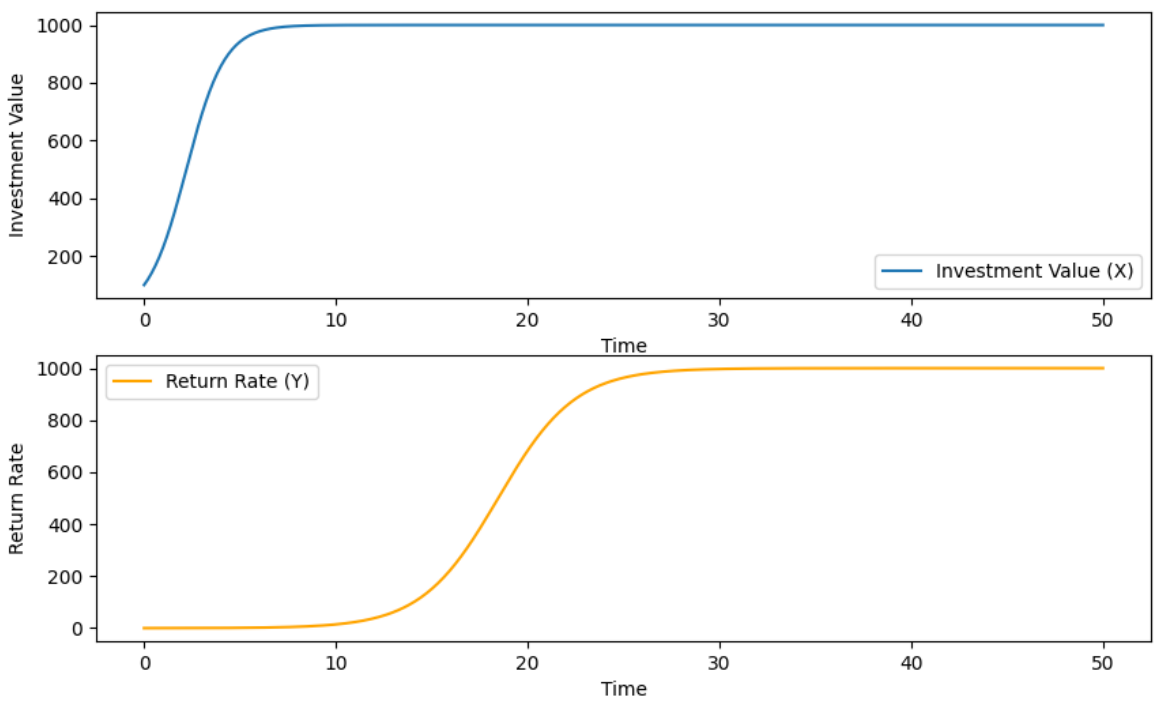
```
# DIRK(2,1) method
def dirk_method(f, t_span, y0, r, K, alpha, C, dt):
    t0, tf = t_span
    n_steps = int((tf - t0) / dt)
    t_values = np.linspace(t0, tf, n_steps)
    y_values = np.zeros((n_steps, len(y0)))

    # Initial condition
    y_values[0] = y0

    # Butcher tableau for DIRK(2,1): a11=1, b1=1/2, b2=1/2
    a11 = 1
    b1 = 0.5
    b2 = 0.5

    for i in range(1, n_steps):
        t = t_values[i - 1]
        y = y_values[i - 1]

        # First stage
        k1 = f(t, y, r, K, alpha, C)

        # Second stage
        k2 = f(t + dt, y + dt * a11 * k1, r, K, alpha, C)

        # Update solution
        y_values[i] = y + dt * (b1 * k1 + b2 * k2)

    return t_values, y_values
```

Visualization

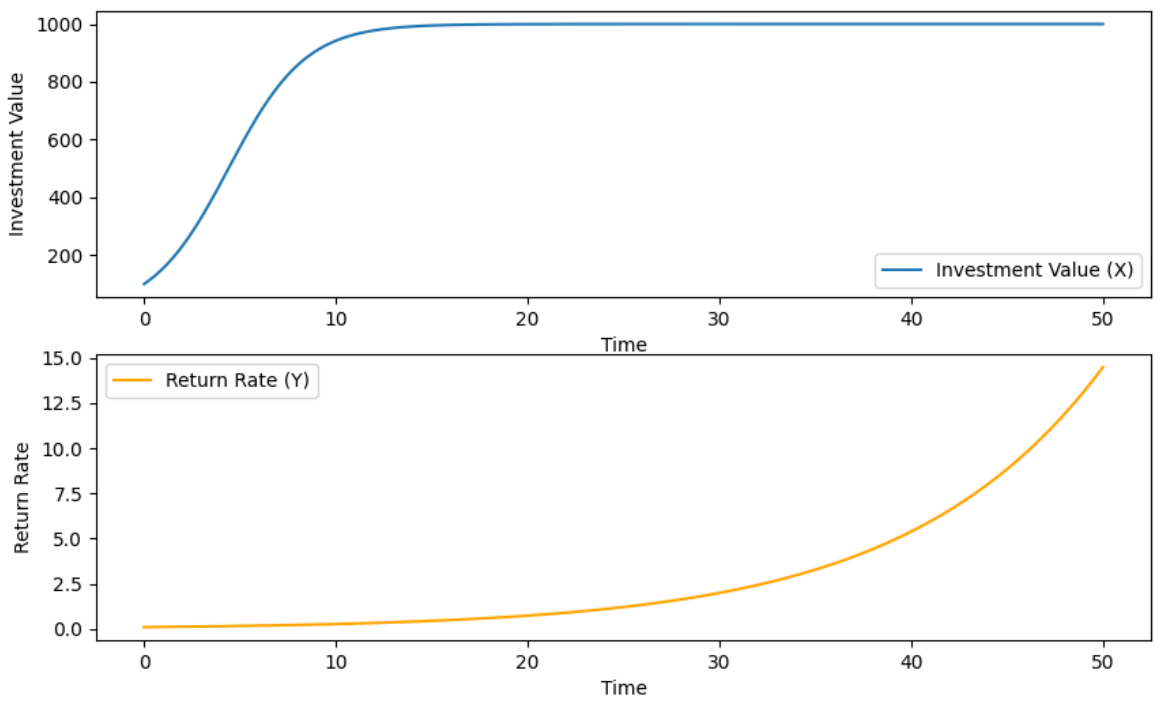
- $X_0 = 1000, Y_0 = 1000,$ `r = 0.2`, `K = 1000`, `a = 0.3`, `C = 1000`



- $X_0 = 1000, Y_0 = 1000,$ `r = 1`, `K = 1000`, `a = 0.5`, `C = 1000`



- $X_0 = 1000, Y_0 = 1000,$ `r = 0.5`, `K = 1000`, `a = 0.3`, `C = 1000`



Conclusion

The graphs show that:

1. **Investment Value (X):** Increases over time, following logistic growth. It slows as it approaches the carrying capacity (K), stabilizing near that value.
2. **Return Rate (Y):** Starts low and gradually increases before leveling off, indicating a saturation point determined by the carrying capacity (C).
3. **Parameter Effects:** Larger values for carrying capacity slow growth, while smaller values lead to faster saturation.

In conclusion, the system models logistic growth, with both investment and return rate stabilizing over time.