EXPERIMENT NO: 01

EXPERIMENT NAME: EXPONENTIAL GROWTH AND DECAY

OBJECTIVE:

To simulate exponential population growth and decay using a first oderr differentential equation -

$$\frac{dP}{dt} = KP$$

and verify the analytical solution using MATLAB/Simulink.

THEORY:

Population P(t) changes with respect to time due to births and deaths:

- Rate In (births) = bP
- Rate out (deaths) = mP

So,

$$\frac{dP}{dt} = bP - mP = KP$$

Where $k = b - m$

Case

$k>0$: exponential growth
$k<0$: exponential decay

Analytical Solution:

Starting with $\frac{dp}{dt} = kp$

Separate the variables:

$$\frac{dp}{p} = k\,dt$$

Integrate both sides:

$$\ln|P| = kt + C$$

Exponentiate:

$$P(t) = Ae^{kt}$$

Using initial condition $P(0) = P_0$: $A = A_0$

final solution: $P(t) = P_0 e^{kt}$

for this experiment:

$P_0 = 10$

$k = -0.5$

So, expected solution $P(t) = 10e^{-0.5t}$

Q-2

SIMULATION MODEL DESIGN:

Use commonly Used blocks:

- Integrator → represents P(t)
- Gain → represents k
- Constant → initial value P(0)
- Scope → to display P(t) over time

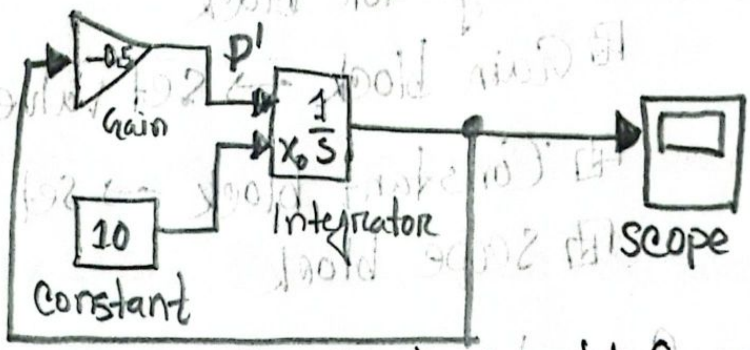


Fig: simulink model for exponential growth & decay. P(0)=10. k=-0.5

Model structure:

- Output of Integrator = P(t)
- Send P(t) into Gain block with gain value k
- Output of gain = kP = input to the integrator
- Constant block gives initial population P_0
- Scope plots the solution.

This forms a closed loop implementing:

$$\frac{dP}{dt} = KP$$

STEPS TO BUILD THE MODEL:

1. OPEN Simulink → Blank Model
2. Add:
   - Integrator block
   - Gain block → set value = −0.5
   - Constant block → set value = 10
   - Scope block
3. Connect:
   - Constant → Integrator (initial condition)
   - Integrator output → Gain input
   - Gain output → Integrator input
   - Integrator output → Scope
4. Set simulation time = 10 seconds
5. Run the simulation.

RESULT:

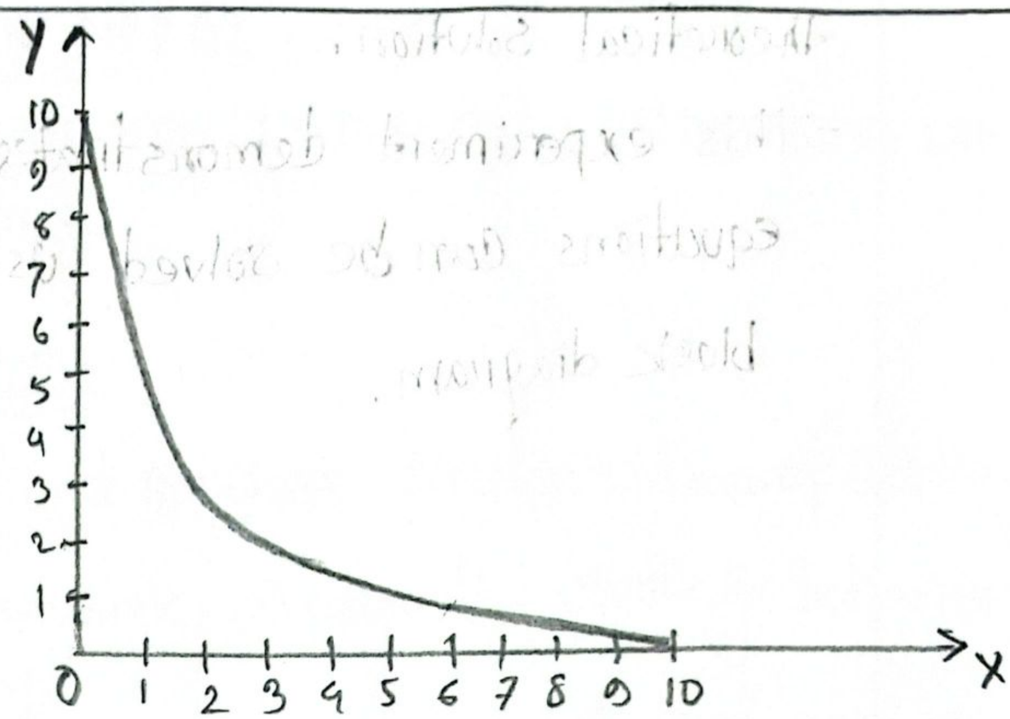


Fig: Solution for the exponential decay with $P(0)=10$ and $K=-0.5$. The simulation time was set at 10.

The output curve matches the analytical expression:

$$P(t) = 10e^{-0.5t}$$

The population decreases smoothly starting from 10 and approaches 0 as time increases.

CONCLUSION: - The population model $\frac{dP}{dt} = kP$ was successfully implemented in Simulink

- For negative k, exponential decay occurs

- This simulated curve matches the

theoretical solution.

- This experiment demonstrates how differential equations can be solved using Simulink block diagram.

EXPERIMENT NO: 02

EXPERIMENT NAME: SIMULATION OF NEWTON'S LAW OF COOLING

OBJECTIVE:

Tö model and simulate Newton's law of cooling using simulink, observe the cooling behavior of a body in an environment, and compare the the simulation with the exact analytical solution. (first order differential equ).

THEORY:

Newton's law of cooling - "Newton proposed that the rate at which an object cools is propor-tional to the temperature difference between the object and it's surroundings.

If

$T(t)$: Temperature of the body at time t

T_a : ambient (room) temperature

T_0 : initial temperature

$k > 0$: cooling constant

Then the differential equation is:

$$\frac{dT}{dt} = -k(T - T_a)$$

Solution of the differential equation:

$$\frac{d}{dt}(T - T_a) = -k(T - T_a)$$

This is exponential decay. Therefore:

$$T(t) - T_a = (T_0 - T_a)e^{-kt}$$

$$T(t) = T_a + (T_0 - T_a)e^{-kt}$$

SIMULATION MODEL DESIGN:

Blocks used

i. Integrator - Computer temperature $T(t)$

ii. Gain Block - Represents constant k

iii. Sum block - Calculates $(T - T_0)$

iv. Constant Blocks - Set values of T_a and T_0

v. Scope - Displays temperature vs time

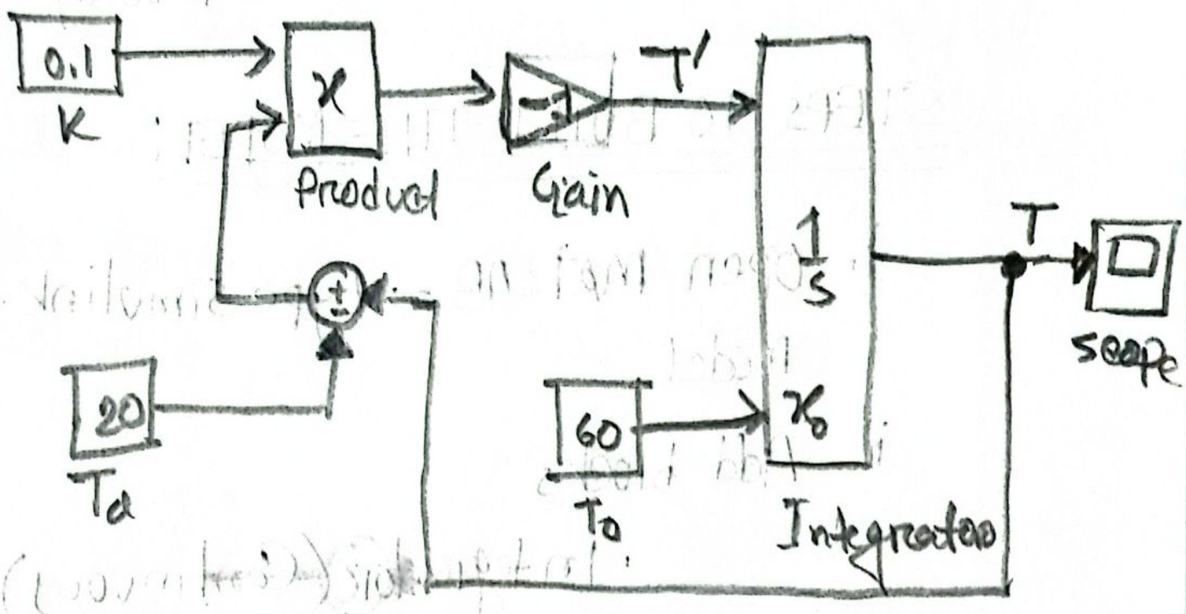


Figure: Simulation model for Newton's Law of Cooling $T' = -k(T - T_a)$, $T(0) = T_0$. Here we set $k = 0.1\ s^{-1}$, $T_a = 20°C$ and $T_0 = 60°C$.

SIMULATION RESULT:

The temperature decreases exponentially. Initially, cooling is rapid.

over time, temperature gradually approaches ambient temperature. The curve never crosses T_a, confirming theoretical behavior.

This validates Newton's law of cooling

STEPS TO BUILD THE MODEL:

i. Open MATLAB → type simulink → Select Blank Model.

ii. Add blocks:

- Integrator (Continuous)
- Sum (Math operations)
- Gain (Math ")
- Constant (for Ta)
- Scope

iii. Set Sum block signs to +- (Computes T−Ta)

iv. Set Gain value to −k (eg. −0.1)

v. Set Constant value $T_a = 20$

vi. Set Integrator initial condition $T_0 = 60$

vii. Connect blocks:

Integrator → Sum(+)

Constant → Sum(-)

Sum → Gain → Integrator

Integrator → Scope.

viii. Set Stop time = 100s, Solver = ode45

ix. Run the simulation & view output on scope.

RESULT:

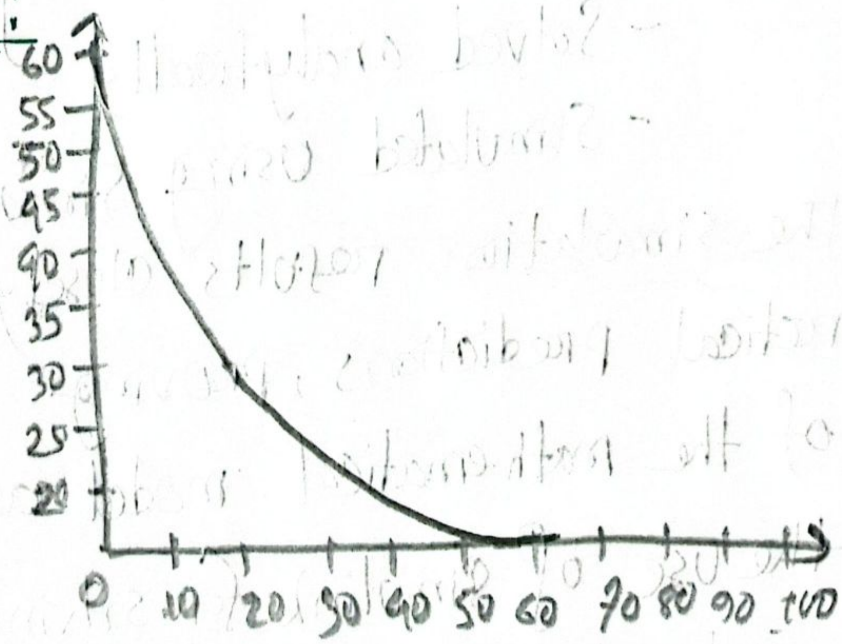


Fig: Solution of Newton's Law of Cooling example.

The temperature decreases exponentially. Initially, colling is rapid

Over time, temperature gradually

approaches ambient temperature.

The curve never crosses Ta, Confirming theoretical behavior.

CONCLUSION:

Newton's Law of Cooling was successfully:

- Modeled mathmatically
- Solved analytically
- Simulated using Simulink

The simulation results closely match theoretical predictions, proving the correctness of the mathematical model and validating the use of Simulink for solving FODE (first order differential equation).

EXPERIMENT NO: 03

EXPERIMENT NAME: SIMULATION OF FREE FALL WITH QUADRATIC DRAG

OBJECTIVE: To model and simulate the motion of an object in free fall under gravity with air resistance proportional to the square of velocity, and analyze the terminal velocity behavior.

THEORY: Free fall without air resistance-

In ideal free fall:

$$\ddot{y}(t) = -g$$

where, $y(t)$ = Position, g : acceleration due to gravity (9.8 m/s2), Sign convention: upward is Positive, so downward acceleration is negative.

Free fall with air resistance:

Real Objects experience drag force. For high speed motion (like falling objects), drag is approximately proportional to v^2:

$$f(v) = bv^2$$

Drag always opposes motion. If the object is falling downward ($v<0$), drag is positive upward.

Newton's Second Law with drag:

$$m\ddot{y} = -mg + f(v)$$

Let $v = \dot{y}$. Divide by mass.

$$\dot{v} = -g + \frac{b}{m} v^2$$

Define: $k = \frac{b}{m}$

Thus the main governing equation $\dot{v} = kv^2 - g$

This is a first order nonlinear differential equation.

Analytical Solution:

$$\dot{v} = kv^2 - g$$

$$\frac{dv}{kv^2-g} = dt$$

let, $\alpha^2 = \frac{g}{k}$

$$\frac{1}{kv^2-g} = \frac{1}{k(v^2-\alpha^2)} \Rightarrow \frac{1}{v^2-\alpha^2} = \frac{1}{2\alpha}\left(\frac{1}{v-\alpha} - \frac{1}{v+\alpha}\right)$$

[by partial fraction dicomposition]

$$\Rightarrow t + C = \frac{1}{2\alpha k} \ln\left(\frac{v-\alpha}{v+\alpha}\right) \text{ [integrating]}$$

$$\Rightarrow v(t) = \alpha \frac{1 - Ae^{2\alpha kt}}{1 + Ae^{2\alpha kt}}$$

where $A = \frac{\alpha - v_0}{\alpha + v_0}$

Terminal velocity

As $t \to \infty$: $v(t) \to -\infty = -\sqrt{g/k}$

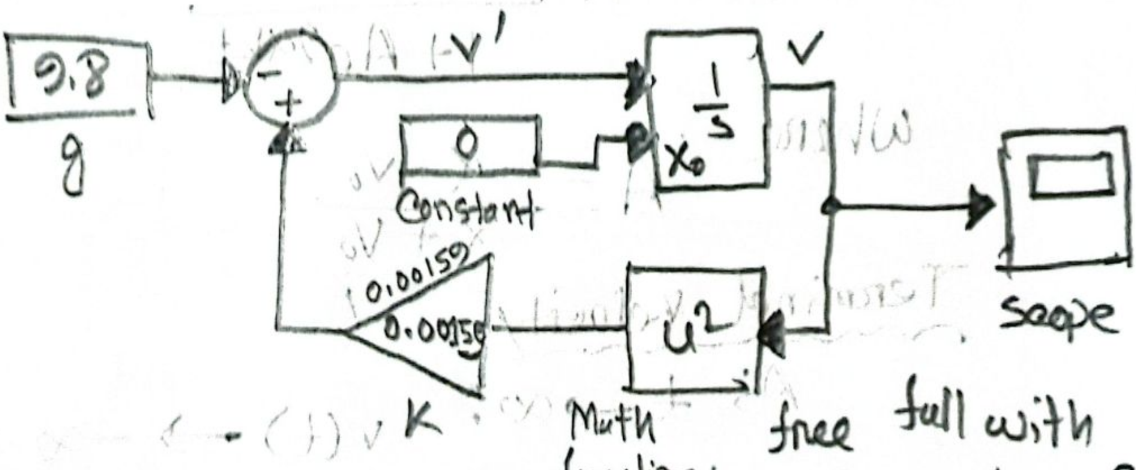
This is the max possible falling speed the object never exceeds it.

SIMULATION MODEL:

Differential equation, $\dot{v} = kv^2 - g$

from commonly use blocks:

Integrator (for velocity)
Gain (for k)
Math function (square)
Constant (for g and k)
Sum block (optional)
Scope (output)



free fall with Drag $v' = Kv^2 - g$

fig: Model for free fall with drag as described by $\dot{v} = Kv^2 - g$

model Structure -

- velocity output $v(t)$ is squared using Math function : u2

- Square is multiplied by gain = K
- Substract g to compute $\dot{v} = Kv^2 - g$
- Feed into Integrator to obtain $v(t)$
- Display solution using scope

Output:

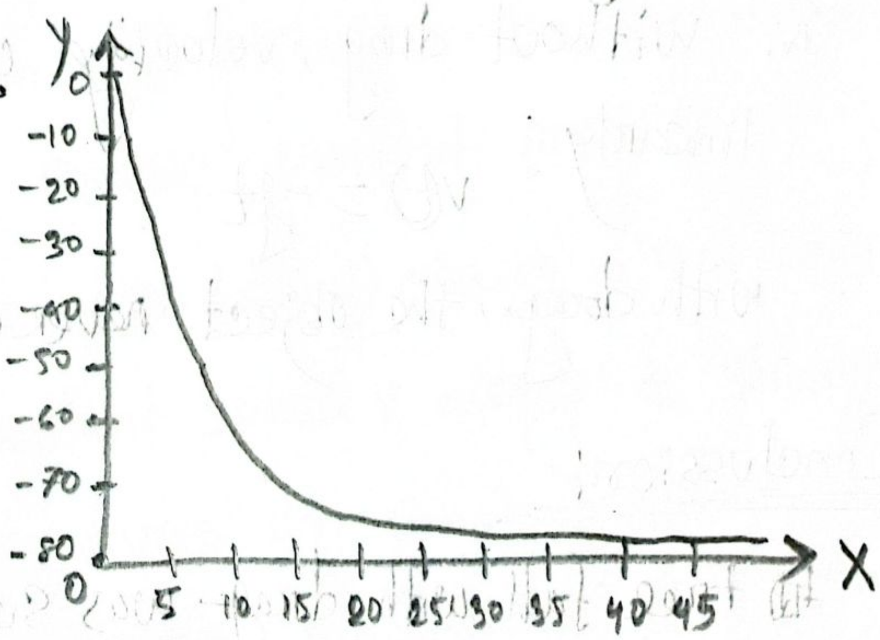


Fig: Solution for free fall with drag with K = 0.00159 starting from rest.

$$v_t = -\sqrt{g/K} = -\sqrt{\frac{9.8}{0.00159}} \approx -78 \text{ m/s}$$

i. The Simulation shows that velocity increases in magnitude until reaching terminal velocity.

$Kv^2 = g$

ii. The simulink result matches the analytical prediction

iii. Quadratic drag significantly limits the falling speed.

iv. without drag, velocity would increases linearly: $v(t) = -gt$

with drag, the object never exceeds a finite speed

Conclussion:

- Free fall with drag was successfully modeled using simulink.
- The nonlinear ODE $\dot{v} = Kv^2 - g$ was implemented correctly using block based simulation.
- Analytical and numerical results match, Confirming the model's correctness.
- The simulation demonstrates terminal velocity & the phy effect of Drag on motion

EXPERIMENT NO: 04

EXPERIMENT NAME: SIMULATION OF PURSUIT DYNAMICS.

OBJECTIVE: Pursuit curves describe the path traced by a body (the pursuer) moving toward a fixed or moving target. This experiment simulates a classic pursuit problem:

A dog at point (x, y) chases a cat moving with constant speed v along a straight line. The dog runs at constant speed ω and always directs its motion along the line of sight to the cat's current position.

goal: (i) Model the dog's path using a differential equ.

(ii) Simulate the system in simulink

(iii) Determine under what conditions the dog catches the cat.

THEORY: At $t=0$:

Cat: start at $(a,0)$, moves along $x=a$ with speed $v \rightarrow$ position: (a, vt)

Dog: starts at $(0,0)$; runs at speed w along line of sight to cat.

Dog's path: $(x(t), y(t))$ or $y = y(x)$

ex-5-1

EXPERIMENT NO: 05

EXPERIMENT NAME: STUDY OF THE LOGISTIC EQUATION (POPULATION GROWTH MODEL)

OBJECTIVE: To study the nonlinear Logistic population growth Model, derive its simplified form using rescaling, and analyze its behavior using the differential equation:

$$\frac{dy}{dt} = ky - cy^2$$

THEORY:

Basic population model

A simple population model assumes constant birth and death rates:

$$\frac{dy}{dt} = by - my$$

where, b = birth rate, m = mortality rate

let, $k = b - m$

Then the equation becomes exponential

growth:
$$\frac{dy}{dt} = ky.$$

This model is unrealistic for large populations because it ignores resource limitations.

Modified mortality rate:

To model real world population dynamics, mortality increases with population:

$$m = \tilde{m} + cy$$

where $\tilde{m}$ = natural mortality

cy = extra deaths due to overcrowding

Substituting: $\frac{dy}{dt} = by - (\tilde{m} + cy)y$

Simplifying: $\frac{dy}{dt} = (b - \tilde{m})y - cy^2$

let: $k = b - \tilde{m}$

Then the logistic equ: $\frac{dy}{dt} = ky - cy^2$

Carrying capacity $K = \frac{k}{c}$

$dy/dt = ky - cy^2$ Can be written as $\frac{dz}{dt} = kz(1-z)$

Ex-5 2

$y = \alpha z$

$\Rightarrow \frac{dy}{dt} = \alpha \frac{dz}{dt} \Rightarrow \alpha \frac{dz}{dt} = k\alpha z - c\alpha^2 z^2$

$\Rightarrow \frac{dz}{dt} = kz - c\alpha z^2 \Rightarrow \frac{dz}{dt} = kz\left(1 - \frac{c\alpha}{k} z\right)$

here, $\frac{c\alpha}{k} = 1$

$\therefore \alpha = k/c$

$\therefore \frac{dz}{dt} = kz(1-z)$ [Reduced logistic equation)

SIMULATION:

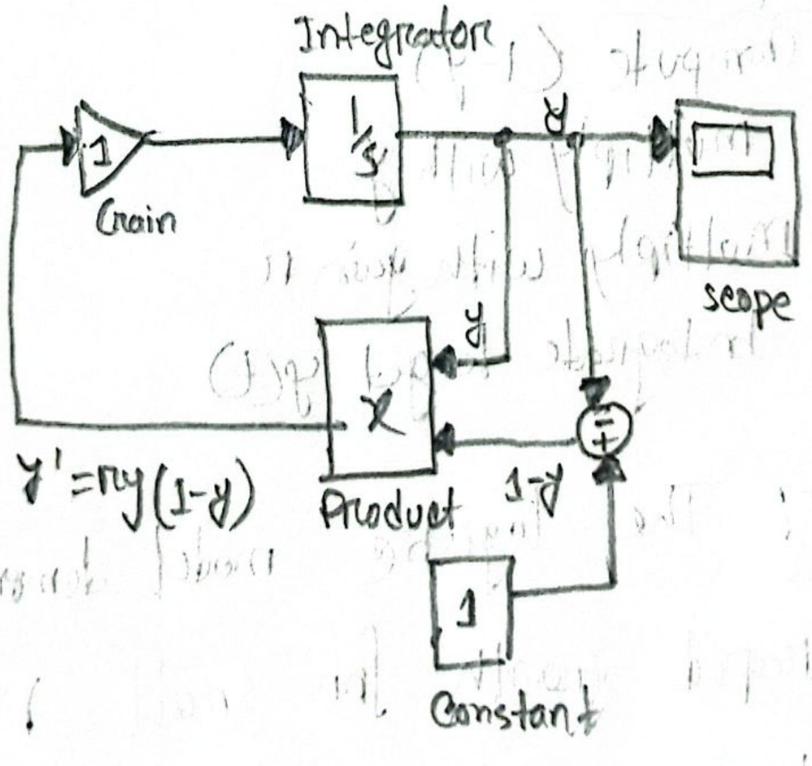


Fig: Simulink model for the logistic equation, $y' = ry(1-y)$.

Model : we simulate $\frac{dy}{dt} = ry(1-y)$

with $r = 1$, $y(0) = 0.1$

Required blocks :

i. Integrator ,

ii. Product block

iii. Sum block

iv. Gain block (r)

v. Constant block

vi. Scope

Block operation :

Compute $(1-y)$

multiply with y

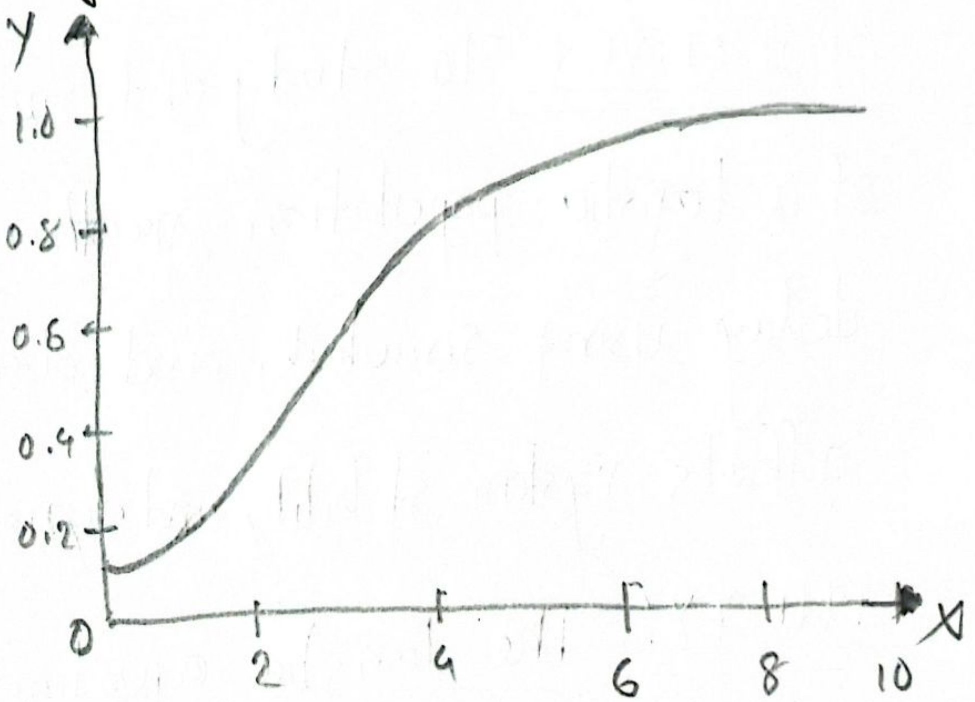
multiply with gain r

Integrate to get $y(t)$

RESULT : The logistic model demonstrates rapid growth for small populations limiting behavior for large populations stable carrying capacity .

nonlinear dynamics governed by population crowding.

Simulation output confirms the theoretical S-shaped logistic curve.



CONCLUTION: The logistic equation provide a more realistic model of population growth than the exponential model. By including a nonlinear mortality term, it predicts bounded population behavior. The parameter reduction proves the models universality.

EXPERIMENT NO: 06

EXPERIMENT NAME: LOGISTIC EQUATION WITH DELAY

~~THEORY: To study and s~~

OBJECTIVE: To study and simulate the behavior of a logistic population growth model with time delay using simulink, and observe how delay affects system stability and dynamics.

THEORY: The logistic equation is a population growth model that considers limited resources.

Standard logistic equation is

$$\frac{dy}{dt} = ry(t)(1 - y(t))$$

where, $y(t)$ = Population at time t.

r = growth rate.