

Modeling and Simulation — Lesson 2

Feedback in the Nature

Dynamics of Predator-Prey Interactions

Václav B. Petrák

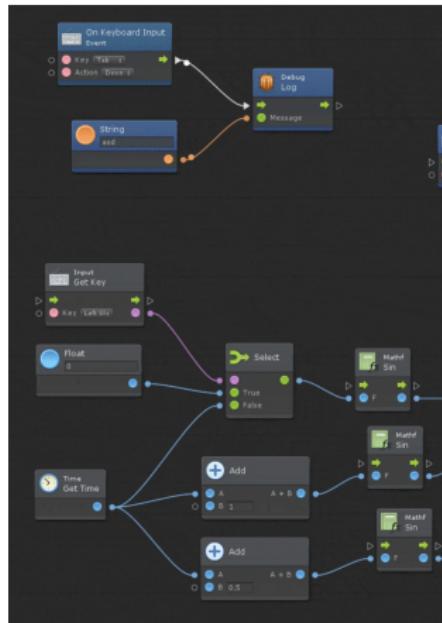
Faculty of Biomedical Engineering
Czech Technical University

April 17, 2025

Introduction

Simulink is visual programming language

- ▶ Visual programming allows users to create programs by manipulating program elements graphically rather than by specifying them textually.
- ▶ A visual representation of algorithms is more intuitive
- ▶ Visual languages help in understanding and designing of systems by visualizing the flow of operations.
- ▶ Visual languages may be more accessible to people without informatics background.



Examples of visual programming languages

- ▶ **Simulink (MathWorks)**: A graphical programming environment by for simulating and analyzing multidomain systems. It offers a drag-and-drop interface with block libraries for system and algorithm modeling, widely used in engineering and aerospace for design and simulation.
- ▶ **LabVIEW (National Instruments)**: A graphical platform for data acquisition, instrument control, and automation. LabVIEW uses a block diagram approach for developing measurement and control systems, suitable for engineers and scientists needing to implement complex systems without deep programming knowledge.
- ▶ **Scratch (MIT Media Lab)**: A block-based visual programming language designed for children and beginners. It allows users to create animations, games, and stories through an intuitive interface, promoting logical thinking and creativity in education.

How Simulink Works

- ▶ Users assemble **block diagrams** by selecting **blocks** from a library and connecting them with **lines** that represent data flow.

How Simulink Works

- ▶ Users assemble **block diagrams** by selecting **blocks** from a library and connecting them with **lines** that represent data flow.
- ▶ At the start of simulation, model specifies the initial states and outputs of the system to be simulated.

How Simulink Works

- ▶ Users assemble **block diagrams** by selecting **blocks** from a library and connecting them with **lines** that represent data flow.
- ▶ At the start of simulation, model specifies the initial states and outputs of the system to be simulated.
- ▶ Simulation runs in loop: At each time step, Simulink computes the new values for the system inputs, states and outputs and updates the model to reflect the computed values. (Think of *for loop* iterations we did earlier)

How Simulink Works

- ▶ Users assemble **block diagrams** by selecting **blocks** from a library and connecting them with **lines** that represent data flow.
- ▶ At the start of simulation, model specifies the initial states and outputs of the system to be simulated.
- ▶ Simulation runs in loop: At each time step, Simulink computes the new values for the system inputs, states and outputs and updates the model to reflect the computed values. (Think of *for loop* iterations we did earlier)
- ▶ At the end of the simulation, the model reflects the final values of the systems inputs, states and outputs

How Simulink Works

- ▶ Users assemble **block diagrams** by selecting **blocks** from a library and connecting them with **lines** that represent data flow.
- ▶ At the start of simulation, model specifies the initial states and outputs of the system to be simulated.
- ▶ Simulation runs in loop: At each time step, Simulink computes the new values for the system inputs, states and outputs and updates the model to reflect the computed values. (Think of *for loop* iterations we did earlier)
- ▶ At the end of the simulation, the model reflects the final values of the systems inputs, states and outputs
- ▶ Simulink provides data logging and display functionality to show data during the simulation.
- ▶ Close integration with MATLAB enables creating Simulink models by coding in Matlab, transfer data to and from Matlab and many other functions.

Applications of Simulink

- ▶ **Engineering:** Used in control systems, signal processing, communications, and robotics,
- ▶ **Automotive:** in the design and test of control systems in vehicles, including engine management systems, advanced driver-assistance systems (ADAS), and autonomous driving technologies.
- ▶ **Aerospace:** designing flight control systems and navigation systems,
- ▶ **Energy Systems:** In the energy sector, Simulink supports the development of renewable energy systems, batteries, power electronics, and smart grid technologies.
- ▶ **Academia:**
 - tool in the teaching
 - modeling tool for many applications, including all topics we discuss in this course.

Task

1. Find a job offering for position that requires knowledge of Simulink — In English or other languages you know
2. Find one academic peer-reviewed research article that uses Simulink.

Libraries offer functionality for various applications

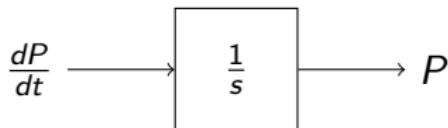
- ▶ **Simulink Standard Library:** Includes basic blocks for mathematical operations, logic, signal routing, and more. Essential for any model.
- ▶ **Simscape:** For physical modeling in domains such as mechanics, electrical, and fluids. Useful for simulating real-world system: Batteries, Electrical systems, Fluid, Mechanical applications
- ▶ **Stateflow:** Adds state machines and flow charts to Simulink models. Ideal for decision logic, mode scheduling, and event handling.
- ▶ **Code generation:** Generate C Code from Simulink models for deployment in a wide variety of applications
- ▶ **Control System Toolbox:** Offers blocks for designing and simulating control system, for models requiring feedback mechanisms.

Task

- ▶ Explores libraries at mathworks.com/products.
- ▶ Find one concrete example that you find interesting.

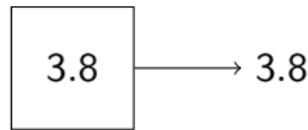
Basic blocks: Integrator

- ▶ The Integrator block in Simulink performs the mathematical integration of an input signal over time.
- ▶ In modeling populations, an integrator is used to simulate the cumulative effect of birth and death rates over time on a population's size.
- ▶ The **input** is a rate of change $\frac{dP}{dt}$ (here as the net growth rate of a population)
- ▶ The **output** is integrator provides the total population P at given time



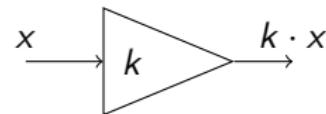
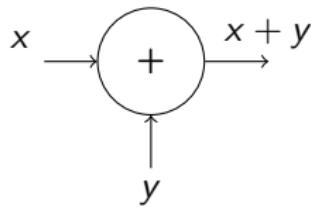
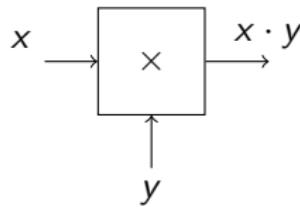
Constant

- ▶ **Constants** in simulations represent fixed values that do not change over time.
- ▶ In population modeling, constants are essential for defining baseline conditions such as carrying capacity, initial population value or intrinsic growth rate



Other common blocks: Products, Sum, and Gain

- ▶ **Product Block:** Multiplies input signals. Used to represent interactions or dependencies between variables.
- ▶ **Sum Block:** Adds or subtracts input signals. Ideal for modeling net changes in a system, such as total population growth.
- ▶ **Gain Block:** Scales an input signal by a constant factor. Represents proportional relationships or amplification factors.



First simulation

In this part you will learn how to

- ▶ Assemble your first model using library blocks
- ▶ Run a simulation
- ▶ Adjust simulation and display settings

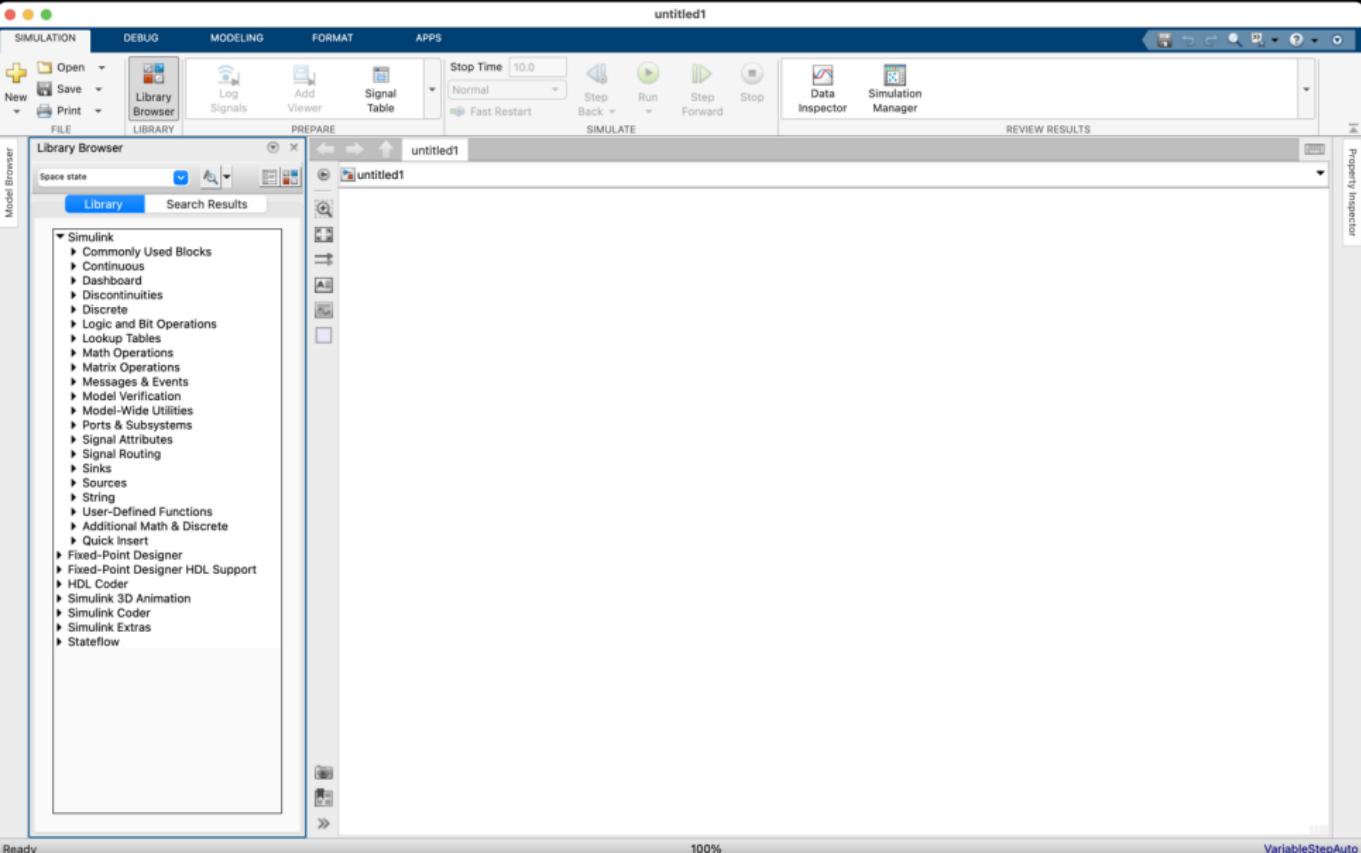
First simulation

In this part you will learn how to

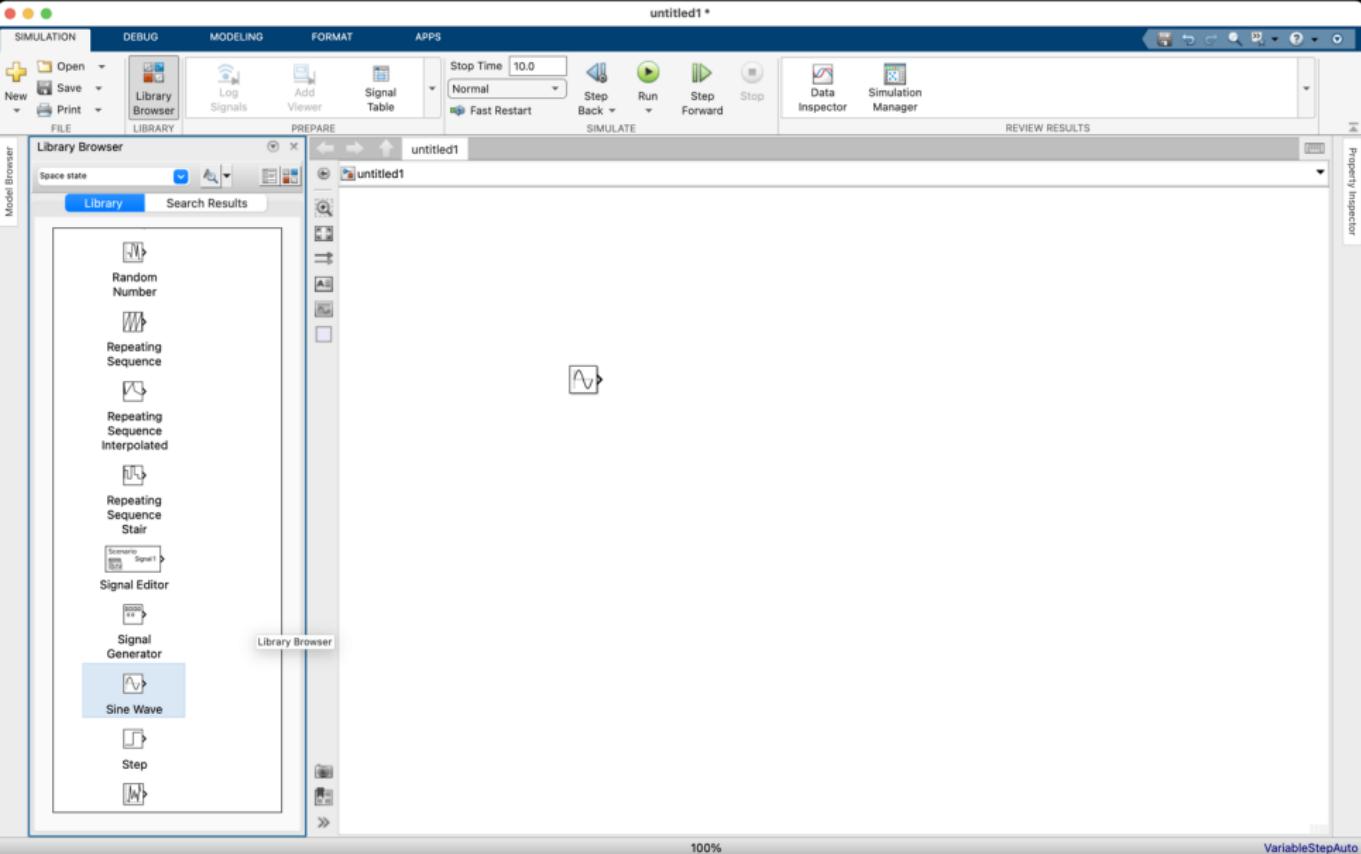
- ▶ Assemble your first model using library blocks
- ▶ Run a simulation
- ▶ Adjust simulation and display settings

We will start with a simple program that generates and displays a sine wave.

Open Simulink environment



Drag a Sine wave block from the library



Add Scope and connect blocs

The screenshot shows the MATLAB/Simulink environment with the following details:

- Toolbar:** Includes buttons for Simulation, Debug, Modeling, Format, and Apps.
- Simulink Library Browser:** Shows the library browser for untitled1. The Simulink library is expanded, displaying categories like Commonly Used Blocks, Continuous, Dashboard, Discontinuities, Discrete, Logic and Bit Operations, Lookup Tables, Math Operations, Matrix Operations, Messages & Events, Model Verification, Model-Wide Utilities, Ports & Subsystems, Signal Attributes, Signal Routing, and Sinks. Specific blocks are shown under these categories: Display, Floating Scope, Out Bus Element, Record, Out1, Scope, Stop Simulation, Terminator, To File, To Workspace, and VV.Graph.
- Model Window:** The main workspace shows a single block diagram consisting of a sine wave source connected to a scope block.
- Simulate Tab:** The SIMULATE tab is selected, showing options for Stop Time (10.0), Normal, Step Back, Step Forward, Run, Stop, Data Inspector, and Simulation Manager.
- Review Results:** A "REVIEW RESULTS" button is located in the top right corner.
- Status Bar:** At the bottom left, it says "Autosave : Backing up all unsaved models: Started". At the bottom right, it shows "100%" and "VariableStepAuto".

Run your first simulation

SIMULATION DEBUG MODELING FORMAT APPS SCOPE

untitled1 *

New Open Save Print Library Browser Log Signals Add Viewer Signal Table Stop Time 10.0 Normal Step Back Fast Restart Run Step Forward Stop Data Inspector Simulation Manager

REVIEW RESULTS

Library Browser untitled1

Space state Library Search Results

Model Browser Project Inspector

Simulink

- Commonly Used Blocks
- Continuous
- Dashboard
- Discontinuities
- Discrete
- Logic and Bit Operations
- Lookup Tables
- Math Operations
- Matrix Operations
- Messages & Events
- Model Verification
- Model-Wide Utilities
- Ports & Subsystems
- Signal Attributes
- Signal Routing
- Sinks

Display Floating Scope

Out Bus Element Out1

- Record Scope
- STOP

Stop Simulation Terminator

- UNDEFINE NAME SIMOUT To File To Workspace VV Graph

Scope

File Tools View Simulation Help

Ready Sample based T=10.000

Ready 100% auto[VariableStepDiscrete]

Simulate for longer time

We now want to run the same simulation for longer time.

Simulate for longer time

We now want to run the same simulation for longer time.

Adjust *Stop time* to 100 and run again.

Adjust Stop time to 100 and run the simulation again.

SIMULATION DEBUG MODELING FORMAT APPS SCOPE

untitled1 *

Stop Time: 100 Normal Step Back Step Forward Stop

untitled1

untitled1

Scope

File Tools View Simulation Help

Ready Sample based T=100.000

Model Browser

Space state Library Search Results

Simulink

- Commonly Used Blocks
- Continuous
- Dashboard
- Discontinuities
- Discrete
- Logic and Bit Operations
- Lookup Tables
- Math Operations
- Matrix Operations
- Messages & Events
- Model Verification
- Model-Wide Utilities
- Ports & Subsystems
- Signal Attributes
- Signal Routing
- Sinks

Display Floating Scope

Out Bus Element

- Record
- Out1 Scope

Stop Simulation Terminator

- To File
- To Workspace

VV_Graph

100% auto[VariableStepDiscrete]

A screenshot of the MATLAB/Simulink interface. The top menu bar includes FILE, DEBUG, MODELING, FORMAT, APPS, and SCOPE. A toolbar below the menu contains icons for Open, Save, Print, Library Browser, Log Signals, Signal Table, Fast Restart, Step Back, Run, Step Forward, Stop, Data Inspector, and Simulation Manager. The main workspace shows a Scope window titled 'Scope' with a yellow waveform. The waveform starts at 0, peaks at 1, dips to -1, and then oscillates between these values. The x-axis is labeled from 0 to 100 with increments of 10. The y-axis has labels at -1, 0, 0.5, and 1. Below the Scope window, status bars show 'Ready', 'Sample based', and 'T=100.000'. On the left, the Model Browser displays a library browser with categories like Simulink, Commonly Used Blocks, and Out Bus Element. The 'Scope' block under Out Bus Element is highlighted with a blue selection box. Other blocks like Record, Out1, Terminator, To File, To Workspace, and VV_Graph are also listed. The bottom status bar indicates '100%' and 'auto[VariableStepDiscrete]'. The bottom left corner says 'Ready'.

Problem with distortion

Resulting wave looks incorrect — Can you guess
Why?.

Problem with distortion

Resulting wave looks incorrect — Can you guess Why?.

The problem is distortion due to insufficient sampling frequency.

Problem with distortion

Resulting wave looks incorrect — Can you guess Why?.

The problem is distortion due to insufficient sampling frequency.

Low sampling frequency will affect our modeling. We need to adjust Model settings.

Select Modeling and Model Settings

SIMULATION DEBUG MODELING FORMAT APPS SCOPE untitled1 *

Model Advisor Find Compare To Environment Model Data Editor Model Explorer Model Settings Create Model configuration parameters (X) Atomic Enabled Subsystem Triggered Subsystem Function-call Subsystem Group Using Area Update Model Stop Time 100 Normal Run Fast Restart COMPILE SIMULATE

EVALUATE & MANAGE DESIGN SETUP COMPONENT

Library Browser untitled1 Library Search Results

Space state

Simulink
Commonly Used Blocks
Continuous
Dashboard
Discontinuities
Discrete
Logic and Bit Operations
Lookup Tables
Math Operations
Matrix Operations
Messages & Events
Model Verification
Model-Wide Utilities
Ports & Subsystems
Signal Attributes
Signal Routing
Sinks

Display Floating Scope

Out Bus Element Out1
Record Scope

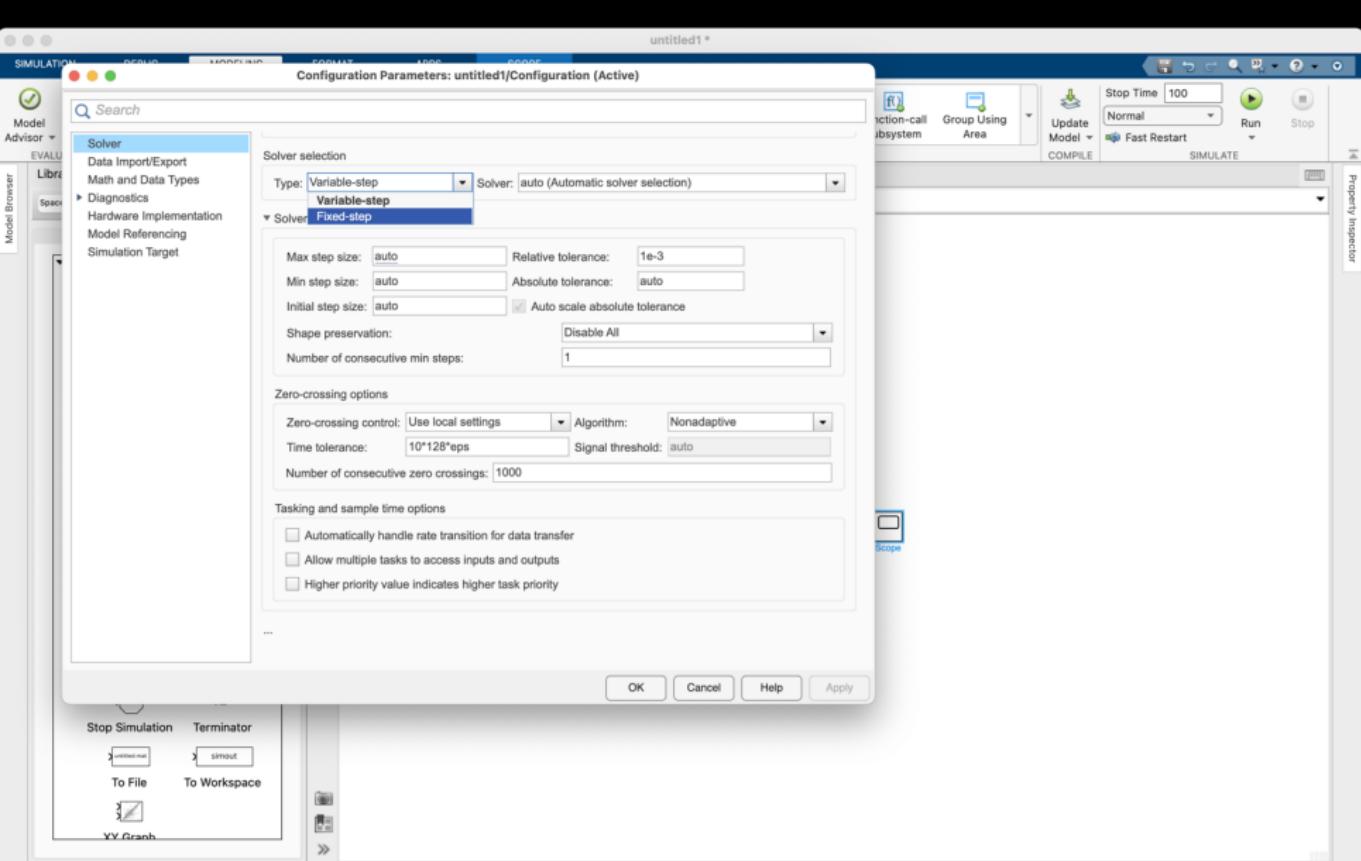
Stop Simulation Terminator
To File Timeout To Workspace

VV Graph

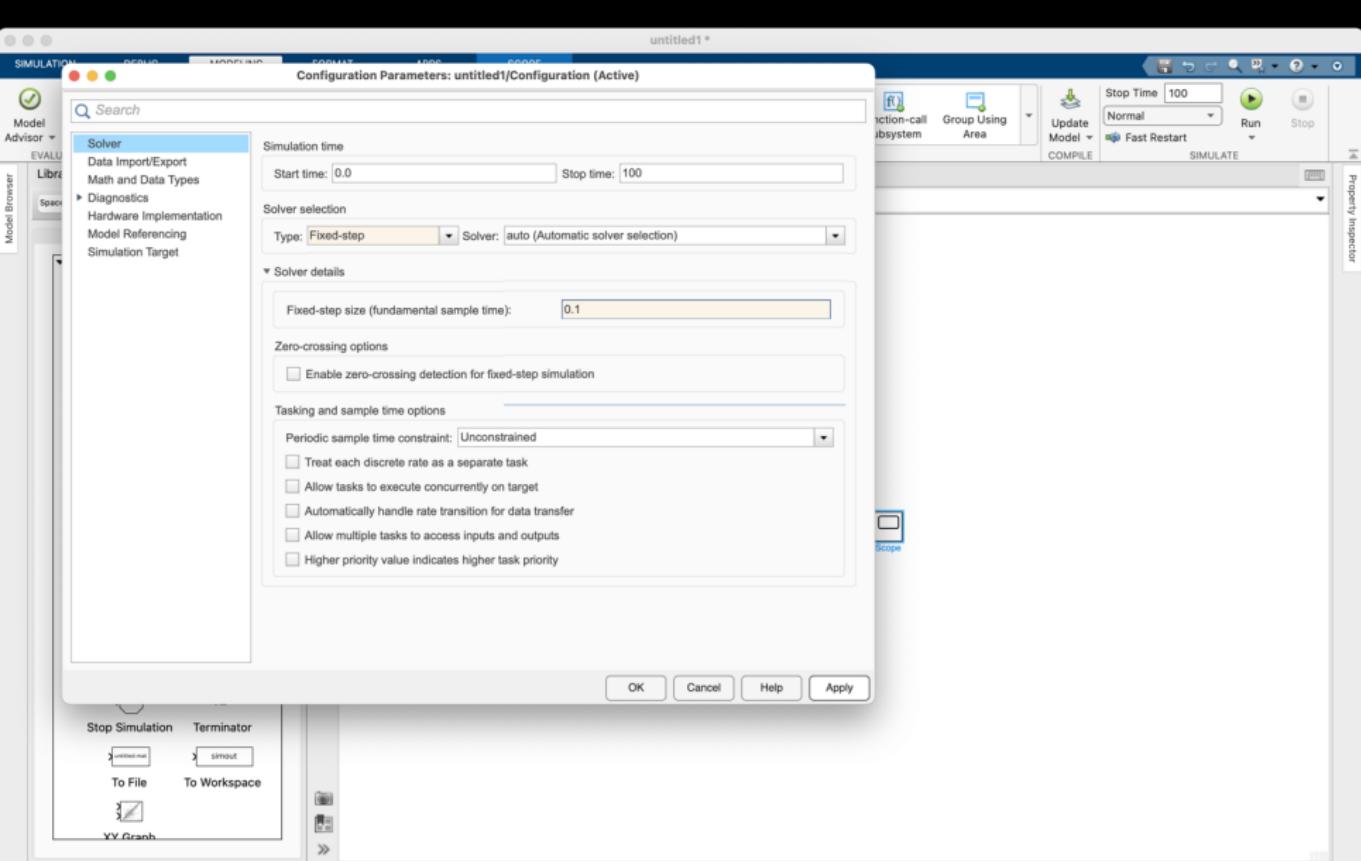
untitled1

Diagram showing a simple model structure: A Display block is connected to a Scope block.

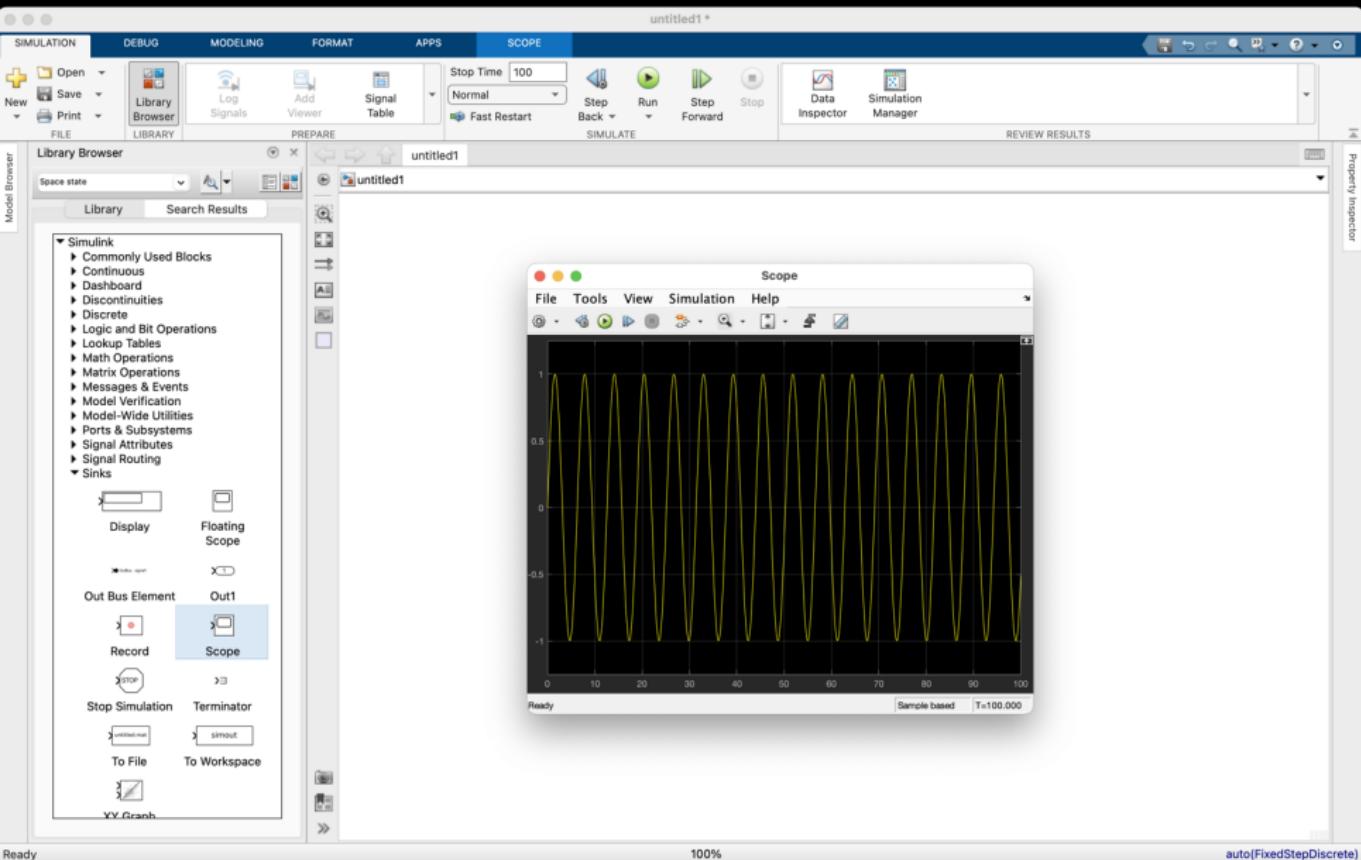
In Solver menu select *Fixed-step* type



Adjust Fixed step size to 0.1



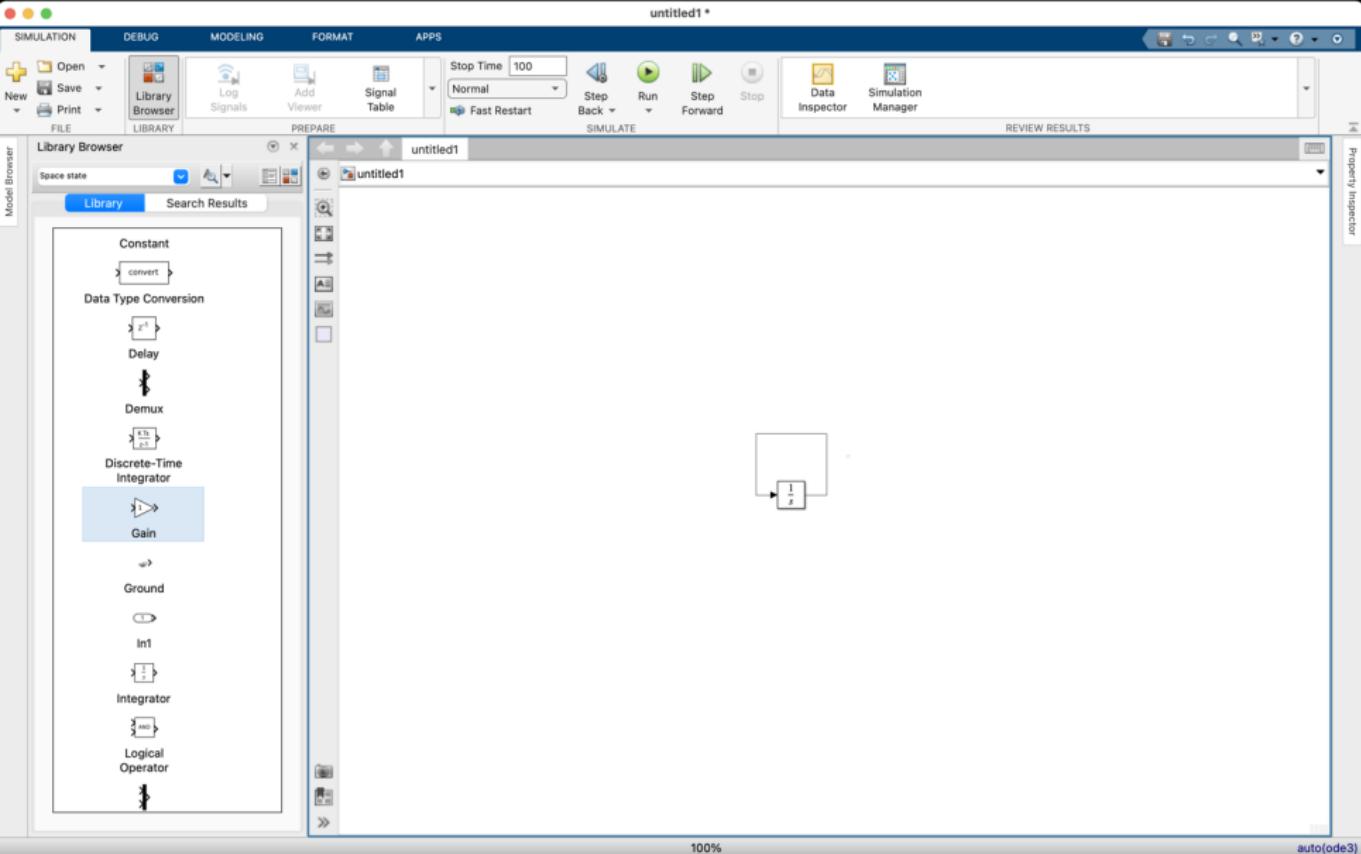
The curve should look correctly now.



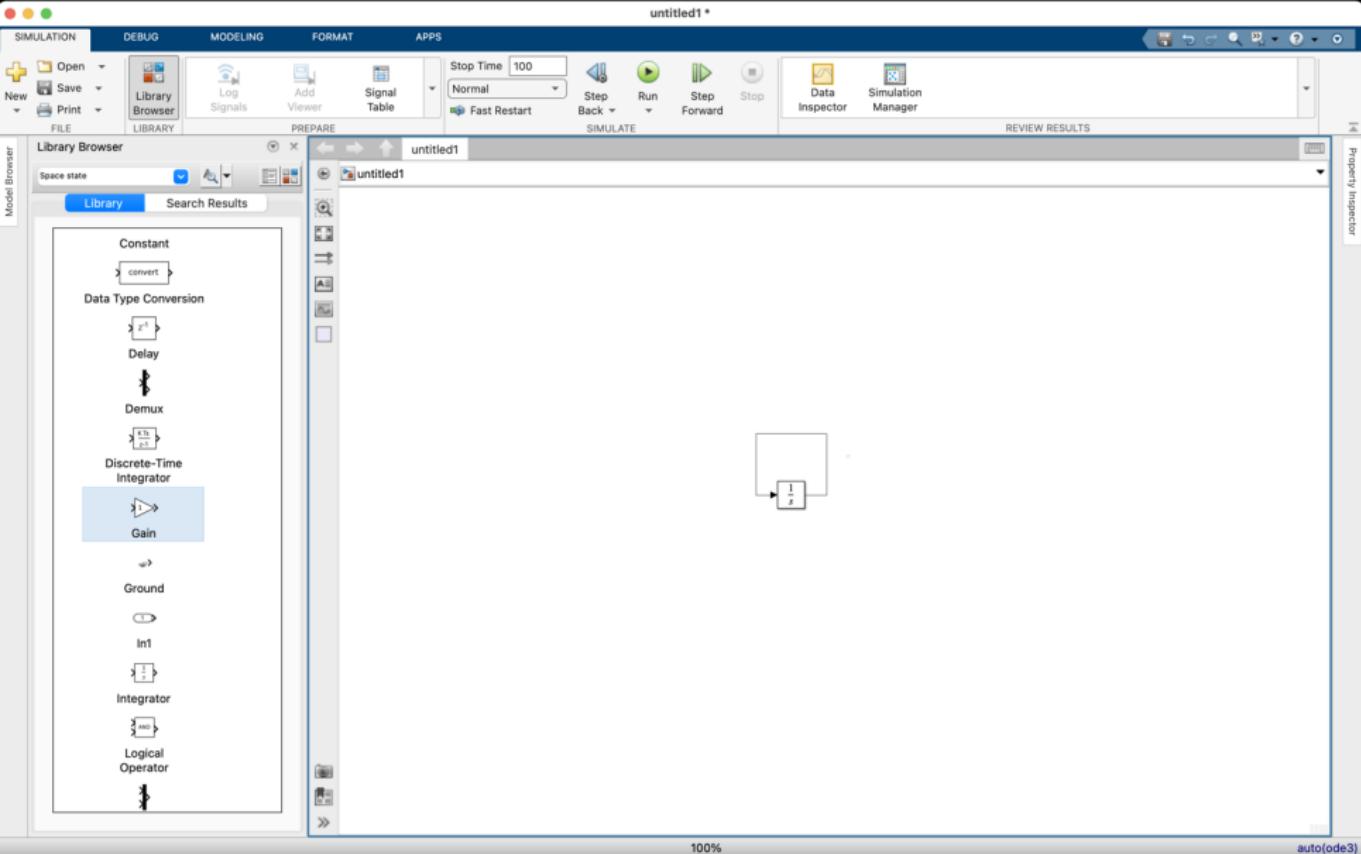
Simulate for longer time

We will now build a Malthusian model.

The core of the model is an integrator



The core of the model is an integrator



Gain element adjust magnitude of the signal.

SIMULATION DEBUG MODELING FORMAT APPS untitled1 *

Open Save Log Signal Add Table Stop Time 100 Normal Step Back Step Forward Run Stop

Print Library Browser Signals Viewer Fast Restart Data Inspector Simulation Manager

untitled1

Space state

untitled1

Library Search Results

Model Browser Project Inspector

Constant

Delay

Demux

Discrete-Time Integrator

Gain

Ground

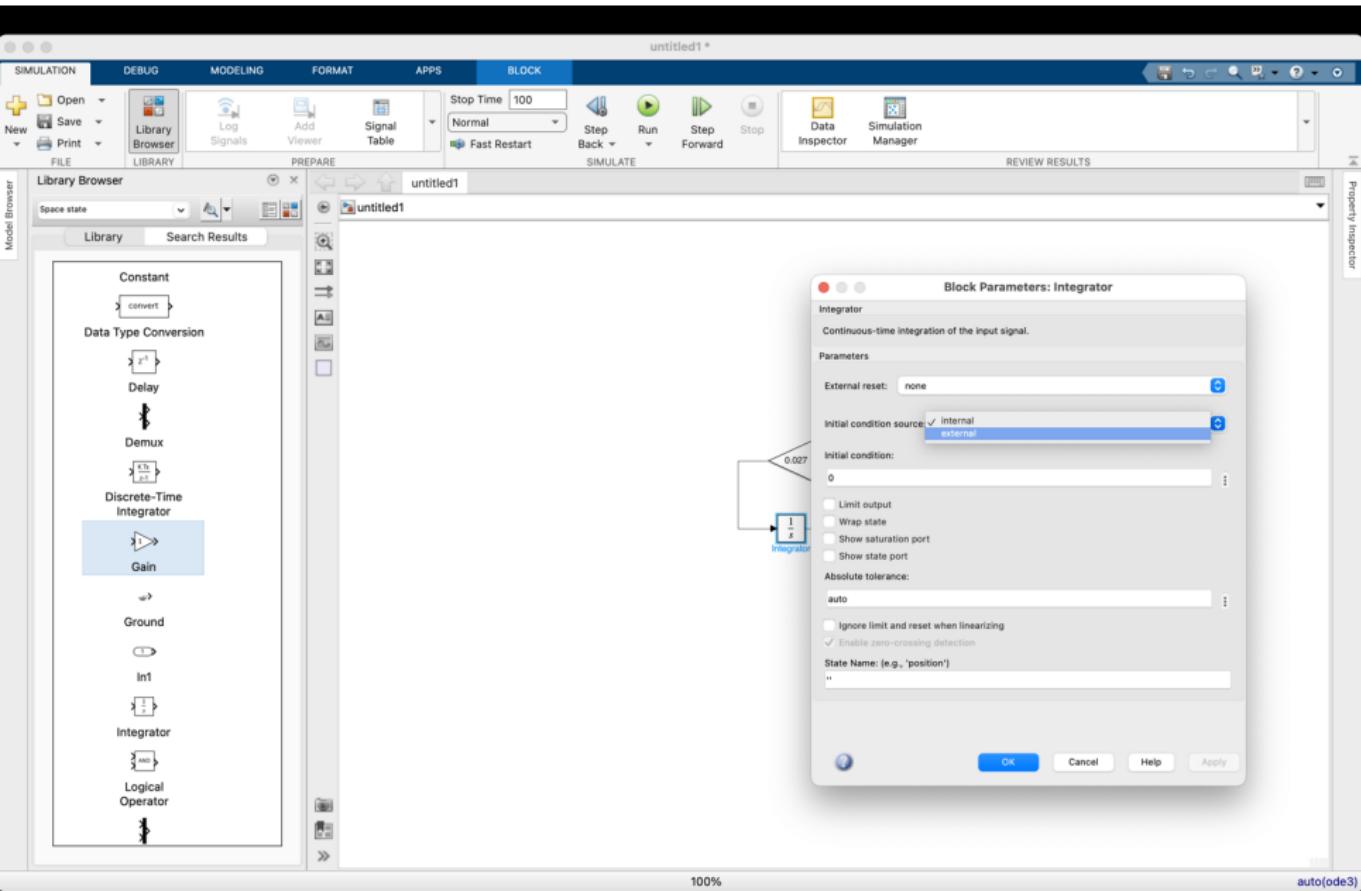
In1

Integrator

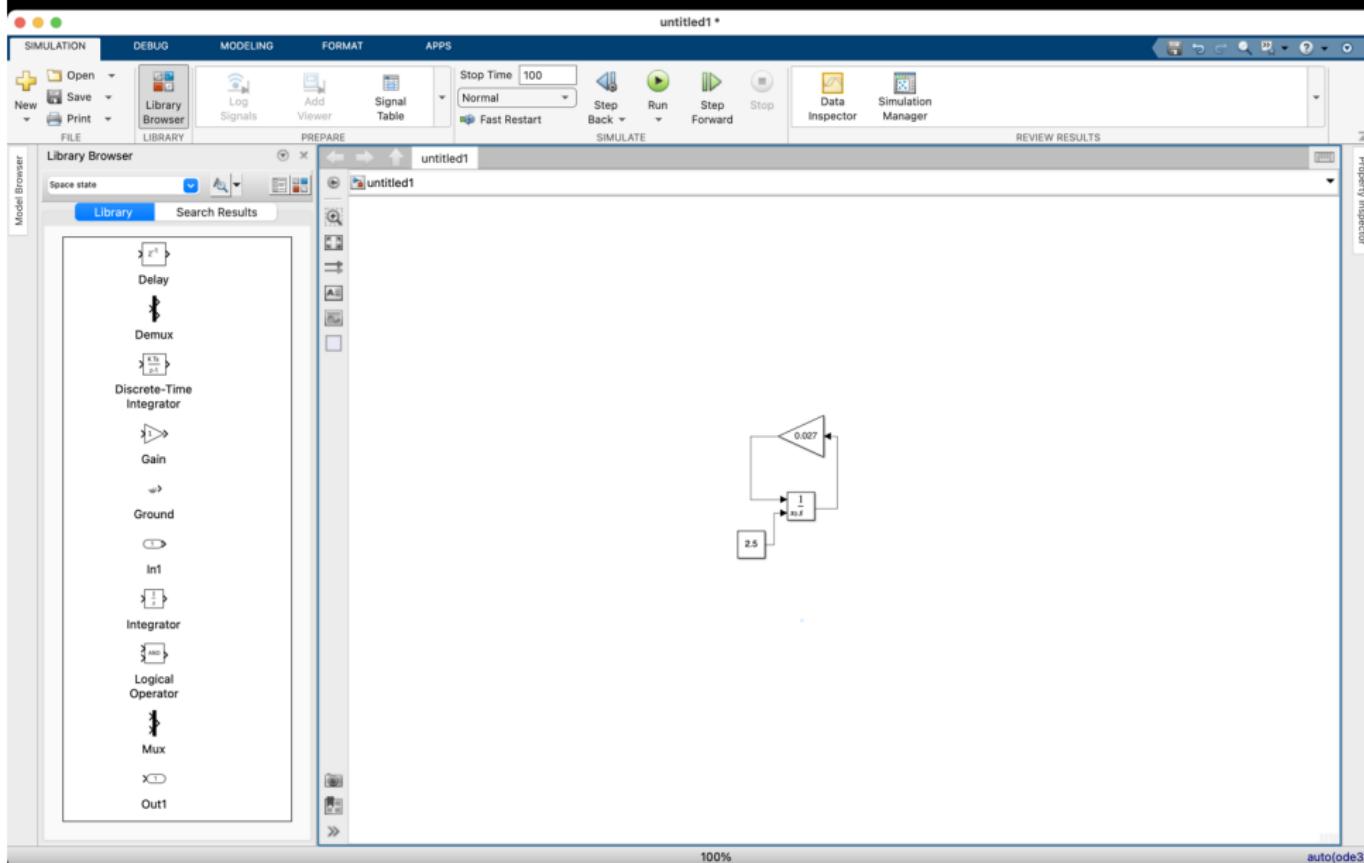
Logical Operator

The screenshot shows a Simulink workspace with a toolbar at the top and a library browser on the left. The library browser has several categories: Constant, Data Type Conversion, Delay, Demux, Discrete-Time Integrator, Gain, Ground, In1, Integrator, and Logical Operator. The 'Gain' block is selected and highlighted with a blue border. On the right side of the screen, a Simulink model window titled 'untitled1' is open, displaying a single block diagram. The diagram consists of a 'Gain' block with the value '0.027' and a 'Discrete-Time Integrator' block connected in series. The model window also shows simulation parameters like 'Stop Time' set to 100 and 'Normal' mode selected.

Initial population can be a block parameter or external constant.



Initial population can be a block parameter or external constant.



We add display and labels

SIMULATION DEBUG MODELING FORMAT APPS SCOPE

untitled1 *

Open Save Print Library Browser Log Signals Add Viewer Signal Table Stop Time 100 Normal Fast Restart Step Back Run Step Forward Stop Data Inspector Simulation Manager

PREPARE SIMULATE REVIEW RESULTS

untitled1

Space state Library Search Results

Model Browser

Simulink

- Commonly Used Blocks
- Continuous
- Dashboard
- Discontinuities
- Discrete
- Logic and Bit Operations
- Lookup Tables
- Math Operations
- Matrix Operations
- Messages & Events
- Model Verification
- Model-Wide Utilities
- Ports & Subsystems
- Signal Attributes
- Signal Routing
- Sinks

Display Floating Scope

Out Bus Element

- Record
- Scope

Stop Simulation Terminator

- To File
- To Workspace

YY Graph

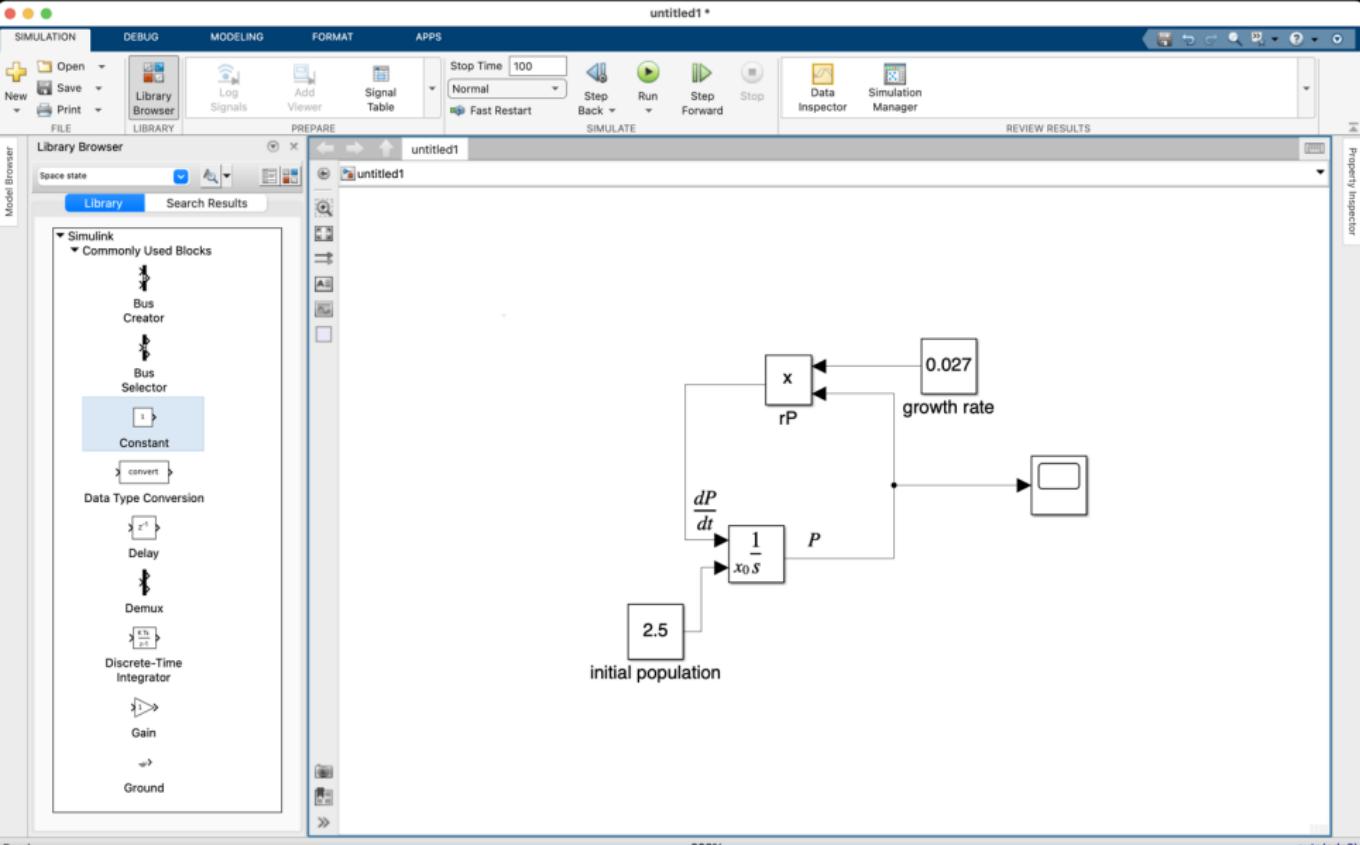
Scope

File Tools View Simulation Help

Ready Sample based T=100.000

203% auto(ode3)

Resulting population growth



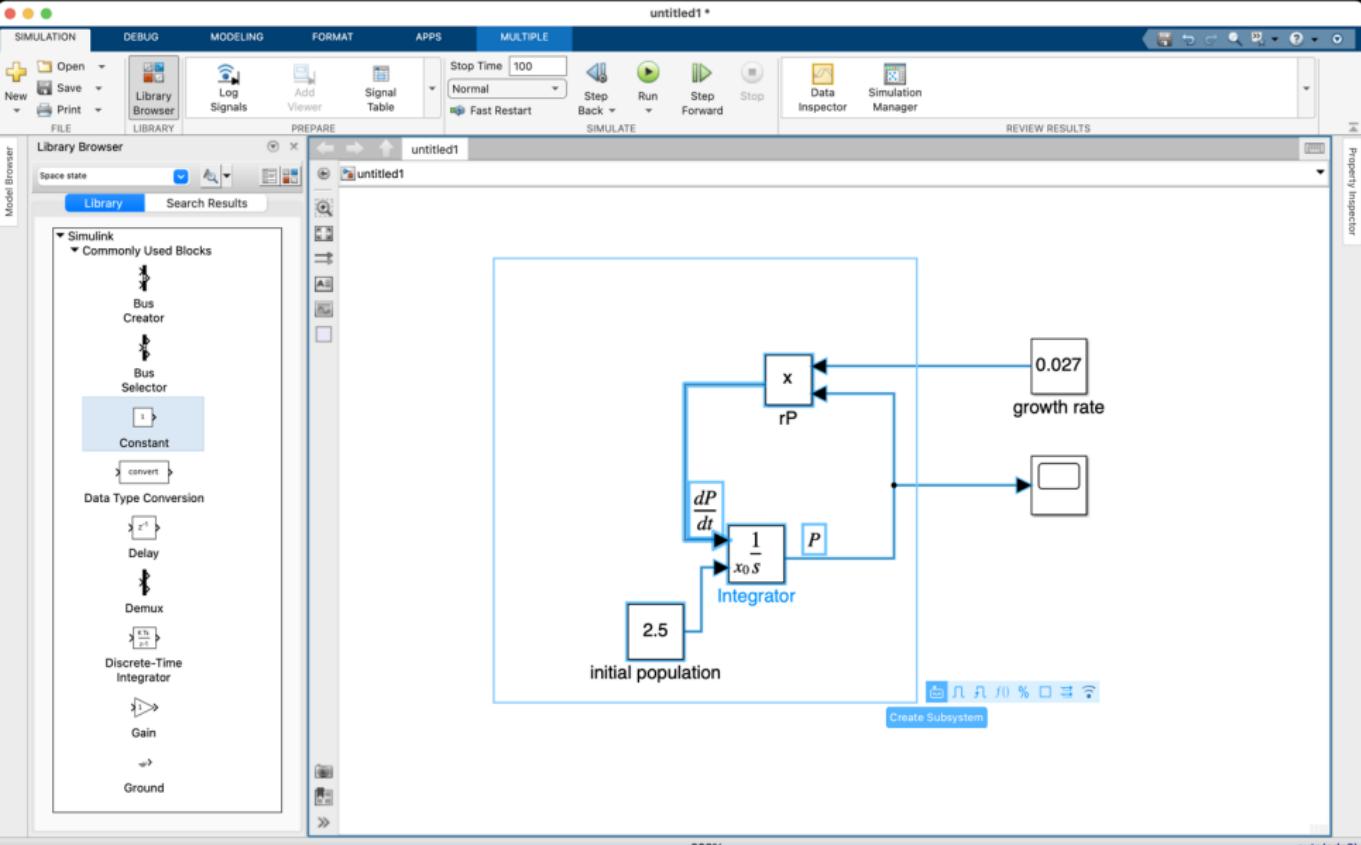
Simulate for longer time

- ▶ We may create a subsystems from our diagram
- ▶ Subsystems are equivalent of functions in Matlab
- ▶ Subsystems are used to re-use and organize code

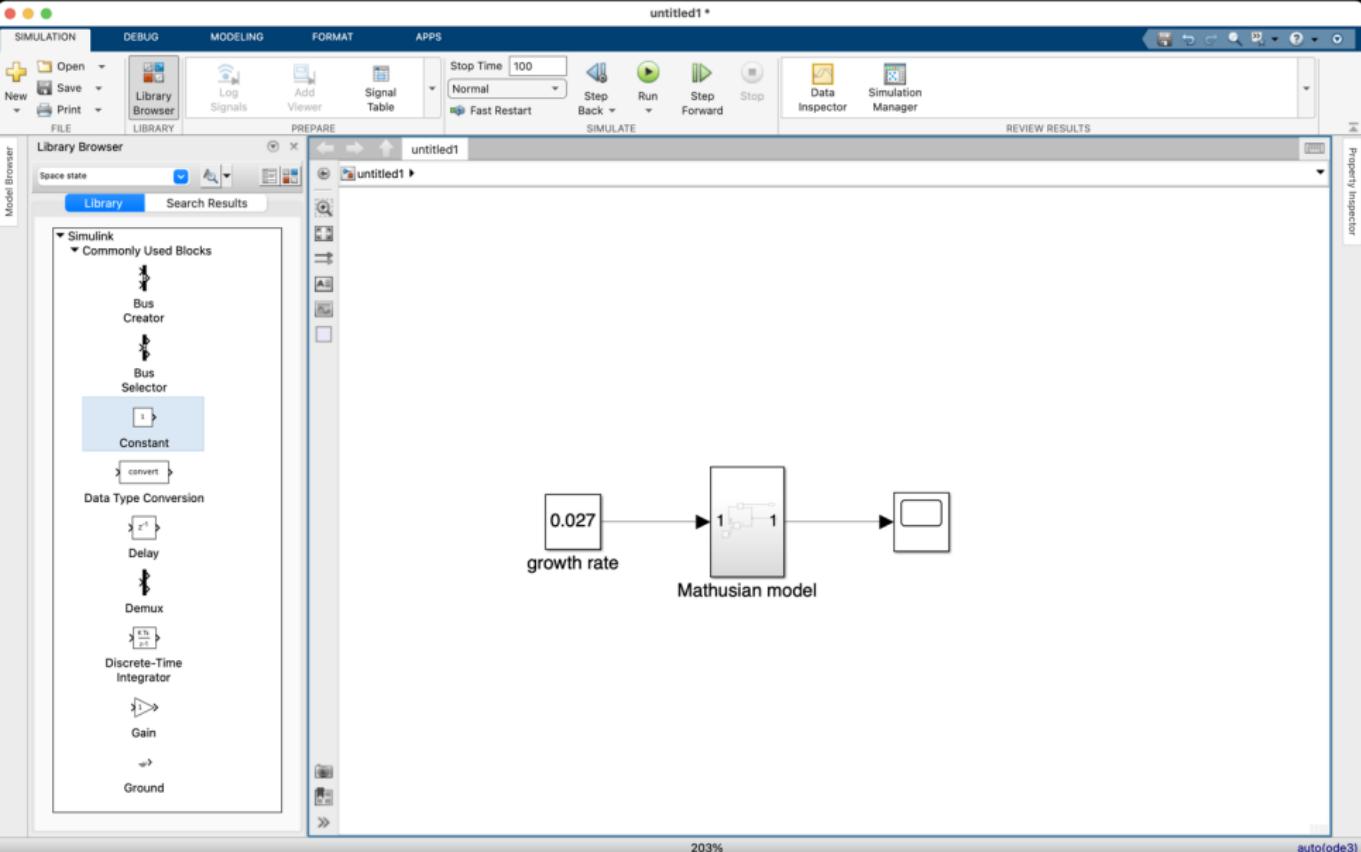
Simulate for longer time

- ▶ We may create a subsystems from our diagram
- ▶ Subsystems are equivalent of functions in Matlab
- ▶ Subsystems are used to re-use and organize code
- ▶ Here, we will create a subsystem from our code to run several scenarios in parallel for different growth rate

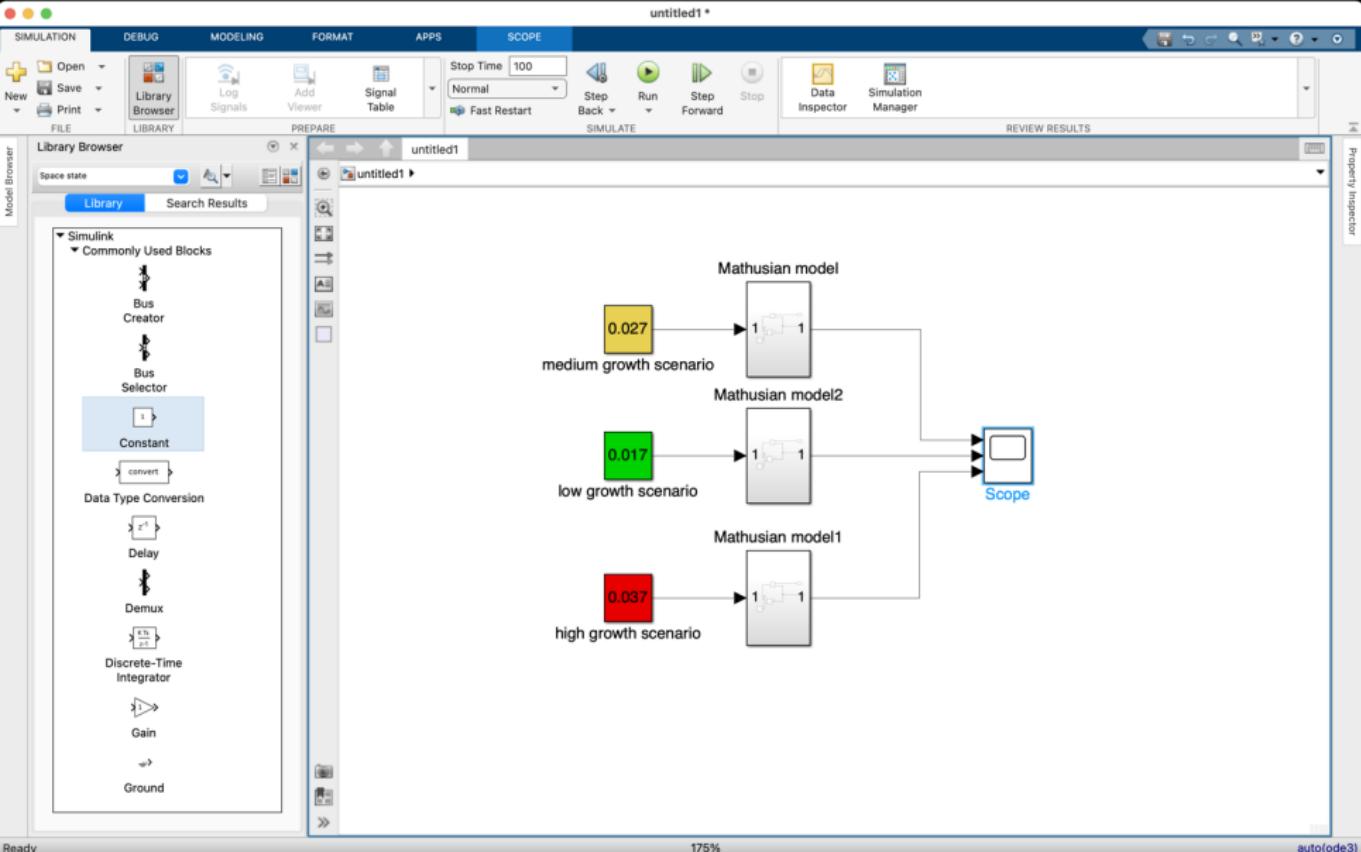
We replace gain block and select relevant parts



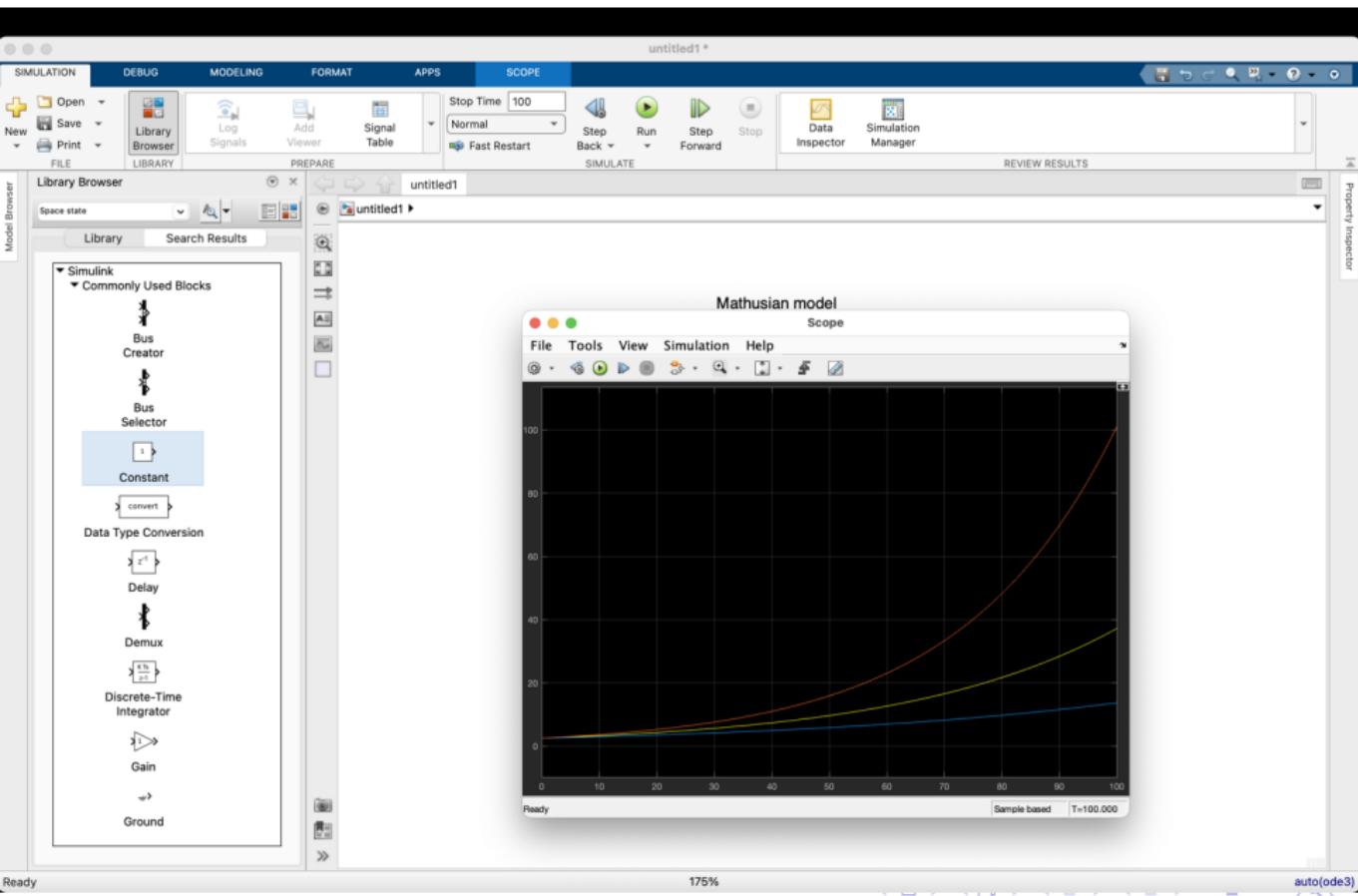
Resulting subsystem



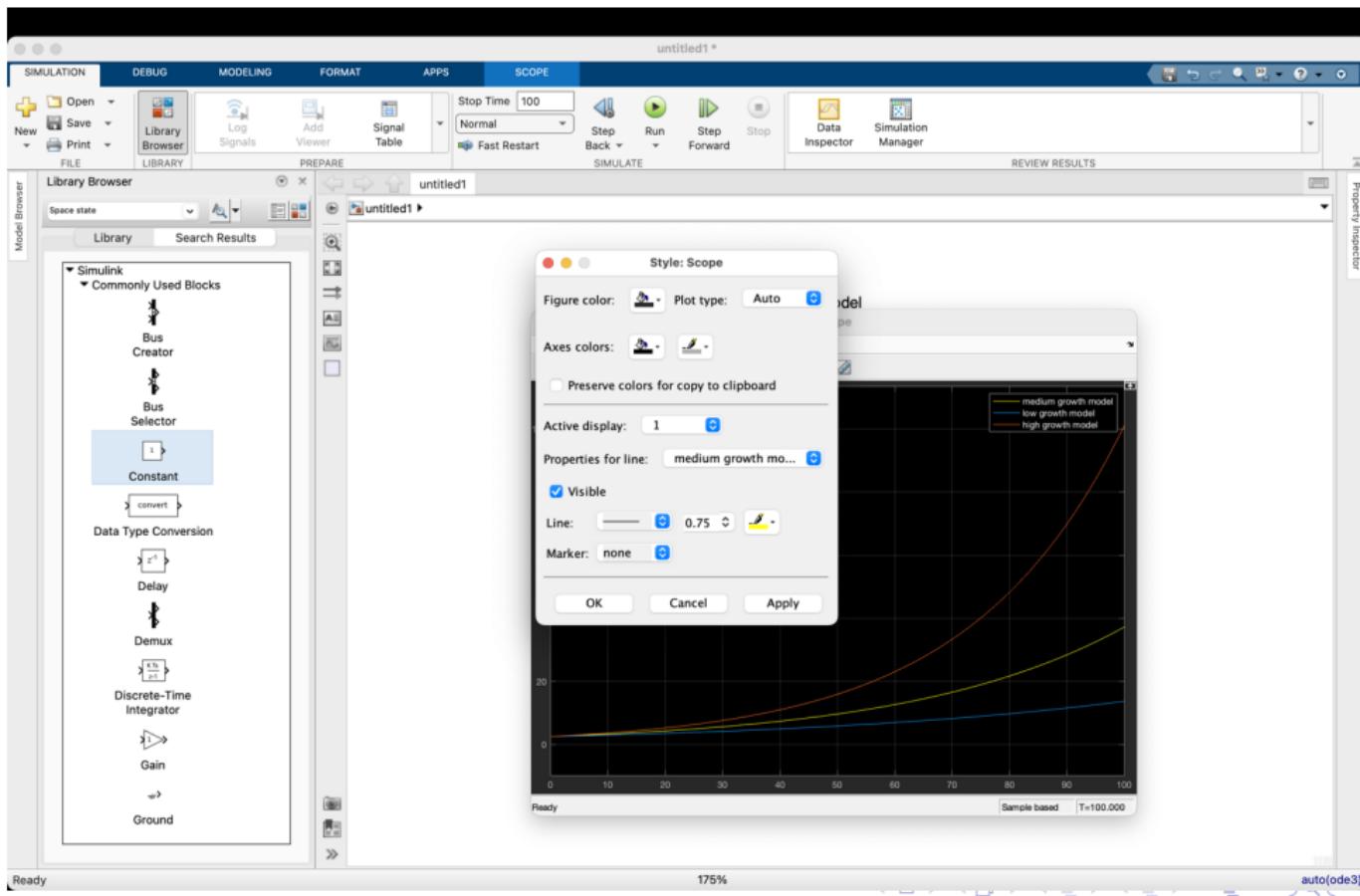
We can duplicate subsystems for parallel simulations



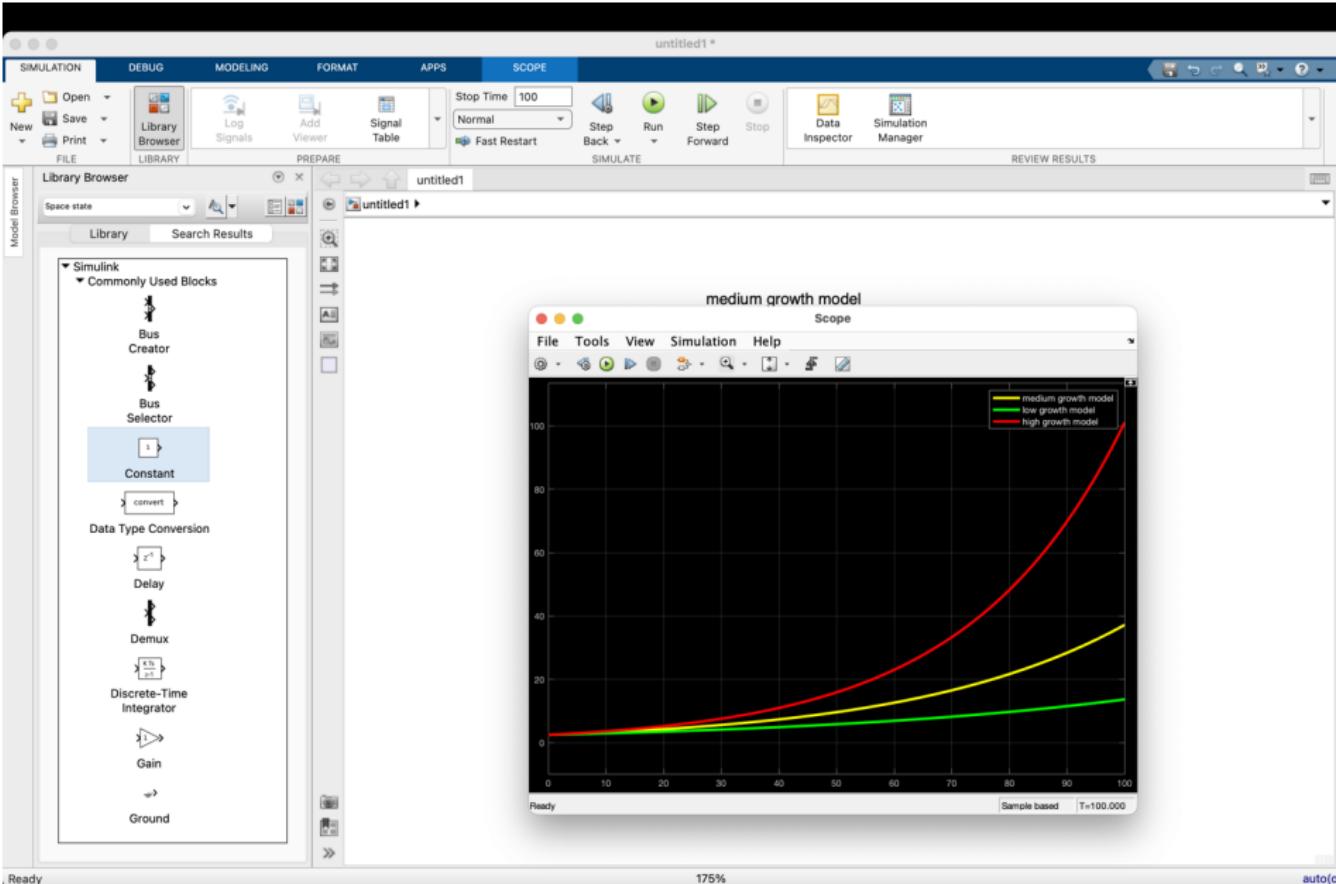
Plots can be easily adjusted



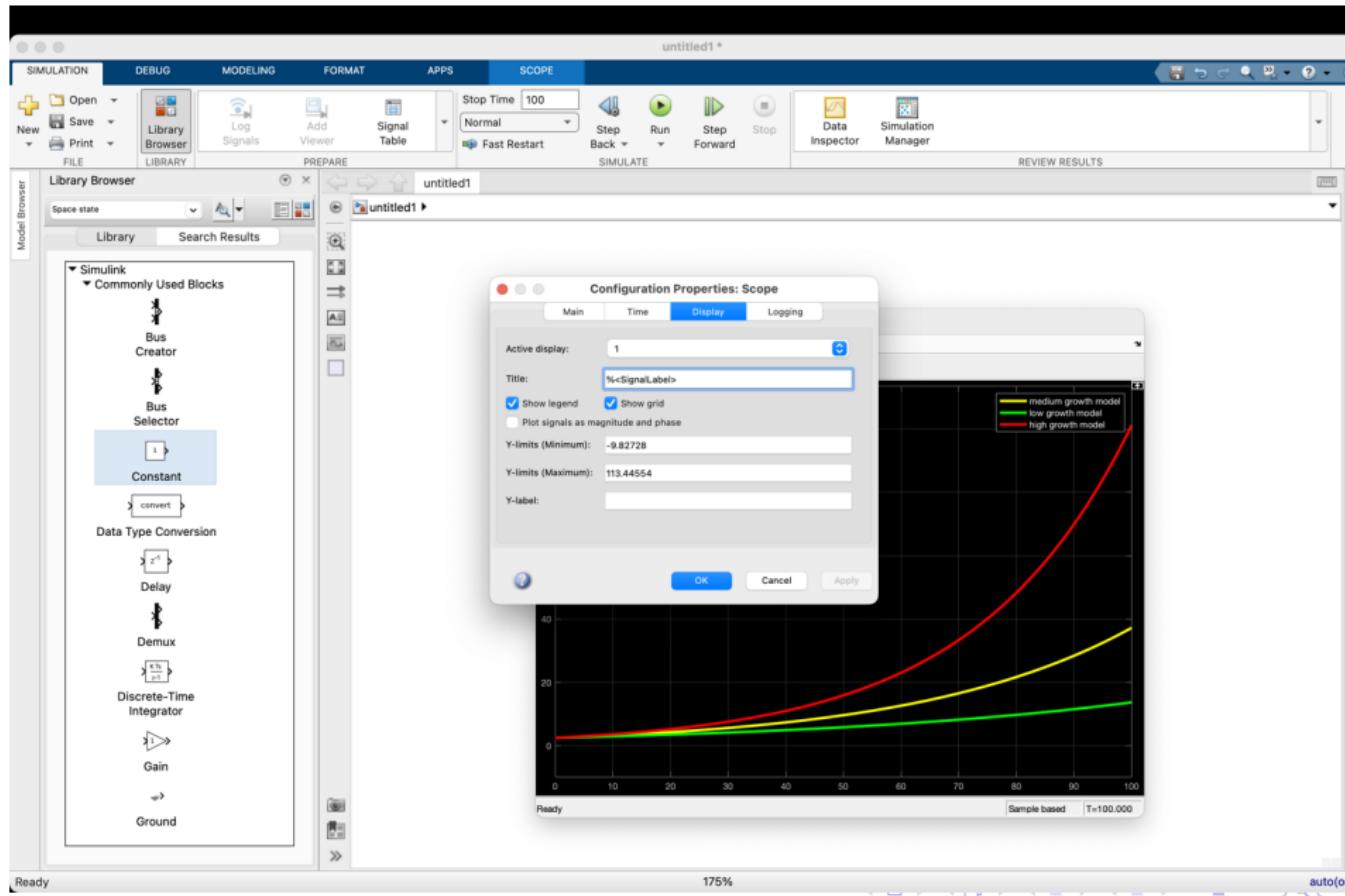
Plots can be easily adjusted



Plots can be easily adjusted



Plots can be easily adjusted



Plots can be easily adjusted

SIMULATION DEBUG MODELING FORMAT APPS SCOPE untitled1 *

New Open Library Browser Log Signals Add Viewer Signal Table Stop Time 100 Normal Step Back Run Step Forward Fast Restart Data Inspector Simulation Manager FILE PREPARE REVIEW RESULTS

untitled1

untitled1

Model Browser

Space state Library Search Results

Simulink
Commonly Used Blocks
Bus Creator
Bus Selector
Constant
convert
Data Type Conversion
Delay
Demux
Discrete-Time Integrator
Gain
Ground

Configuration Properties: Scope

Main Time Display Logging

Active display: 1

Title: Malthusian model - three scenarios

Show legend Show grid
 Plot signals as magnitude and phase

Y-limits (Minimum): -9.82728

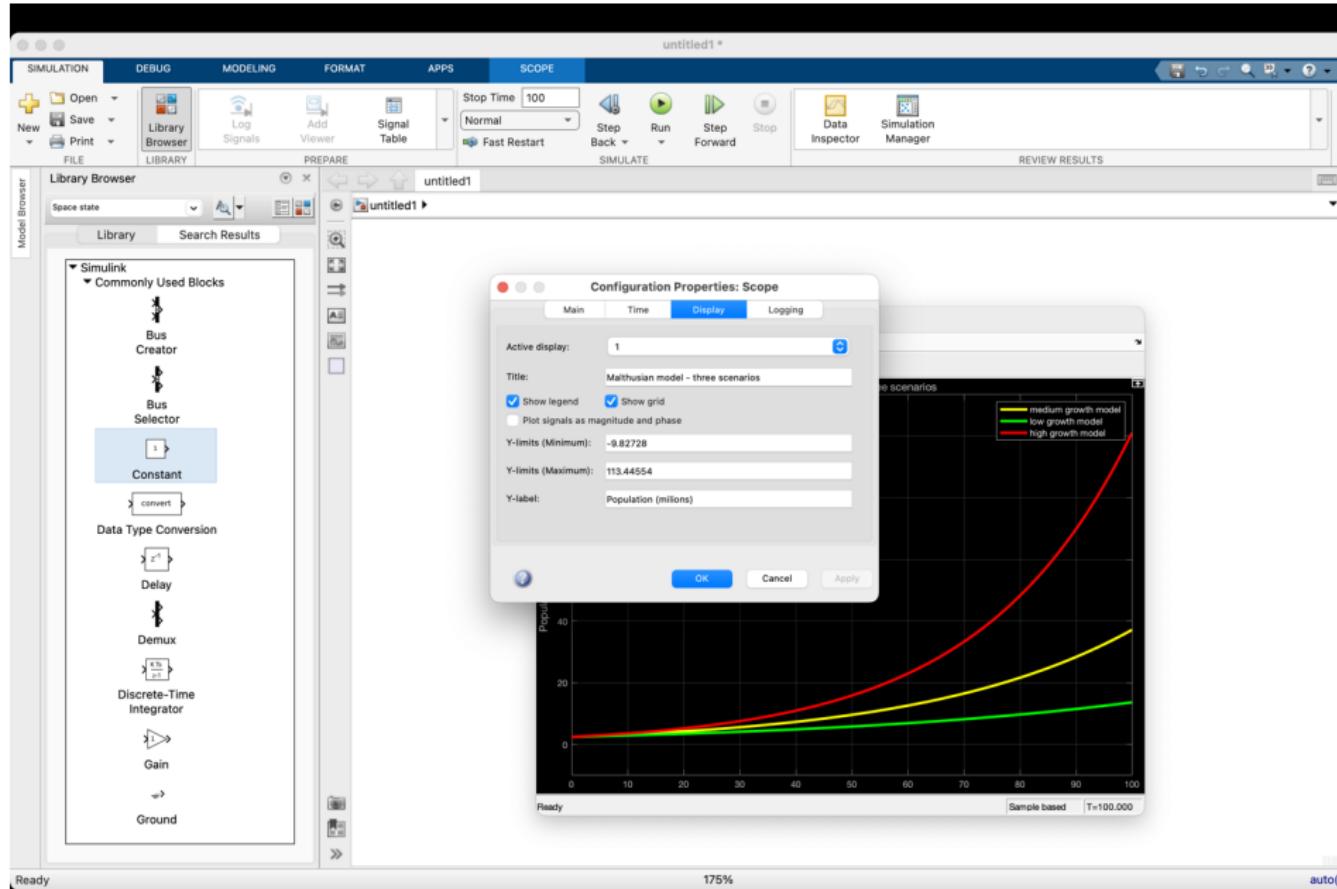
Y-limits (Maximum): 113.44554

Y-label: Population (millions)

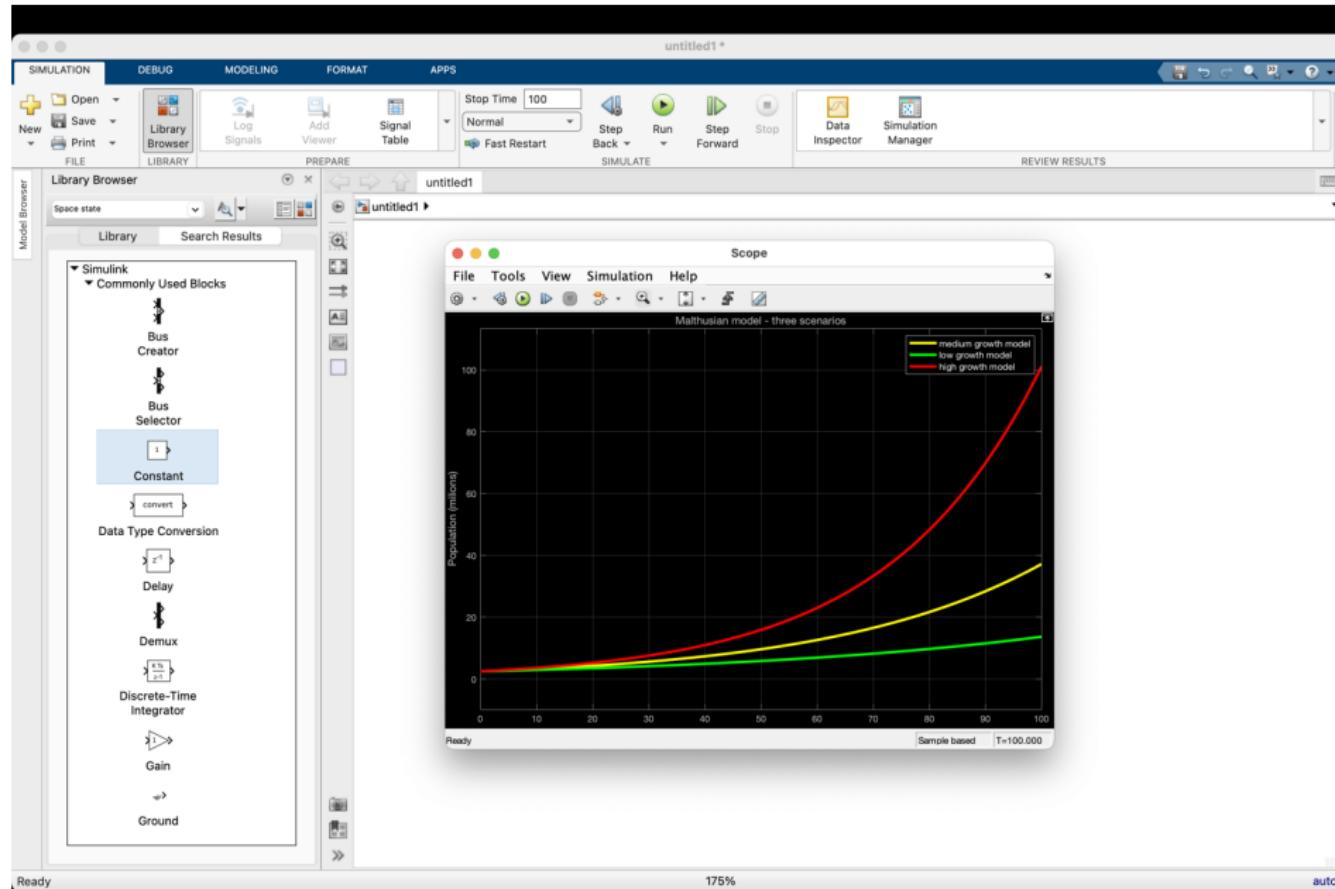
OK Cancel Apply

Population

Ready Sample based T=100.000



Plots can be easily adjusted



Experiments with blocks and settings

SIMULATION DEBUG MODELING FORMAT APPS untitled1 *

Open Save Library Browser Log Signals Add Viewer Signal Table Stop Time 10 Normal Fast Restart Step Back Run Step Forward Stop Data Inspector Simulation Manager

FILE Print PREPARE REVIEW RESULTS

untitled1

Model Browser Library Search Results

Space state

Repeating Sequence Stair Scenario Signal

Signal Editor

Sine Wave

Signal Generator

Step

Uniform Random Number

Waveform Generator

String User-Defined Functions Additional Math & Discrete Quick Insert Fixed-Point Designer Fixed-Point Designer HDL Support HDL Coder Simulink 3D Animation Simulink Coder Simulink Extras Stateflow

x

x

x^2

Scope

untitled1

File Tools View Simulation Help

Scope

0 1 2 3 4 5 6 7 8 9 10

0 0.5 1 -1 -0.5

Ready Sample based Offset=0 T=10.000

200%

auto(FixedStepDiscrete)

The screenshot shows the MATLAB/Simulink environment. At the top, the menu bar includes FILE, SIMULATION, DEBUG, MODELING, FORMAT, and APPS. The SIMULATION tab is selected, showing options like Open, Save, Library Browser, Log Signals, Add Viewer, Signal Table, Stop Time (set to 10), Normal, Fast Restart, Step Back, Run, Step Forward, Stop, Data Inspector, and Simulation Manager. Below the menu is a toolbar with icons for these functions.

The main workspace contains a model named 'untitled1'. The model consists of a Sine Wave block, a Gain block labeled 'x', and a Square block labeled ' x^2 '. The output of the Sine Wave block goes to the Gain block 'x'. The output of the Gain block 'x' goes to the Square block ' x^2 '. The output of the Square block ' x^2 ' is displayed in a Scope block. A signal line from the Gain block 'x' also branches off to the Scope block.

To the left of the workspace is the Model Browser, which lists various blocks and tools. The 'Sine Wave' block is currently selected, indicated by a blue highlight. Other listed blocks include Repeating Sequence Stair, Signal Editor, Signal Generator, Step, Uniform Random Number, and Waveform Generator. A large section of the browser lists additional toolboxes and features: String, User-Defined Functions, Additional Math & Discrete, Quick Insert, Fixed-Point Designer, Fixed-Point Designer HDL Support, HDL Coder, Simulink 3D Animation, Simulink Coder, Simulink Extras, and Stateflow.

Below the workspace is a figure window titled 'Scope' showing a plot of two signals over time. The x-axis ranges from 0 to 10, and the y-axis ranges from -1 to 1. One signal is a blue sine wave starting at 0, peaking at approximately 0.9 at x=2, crossing zero at x=3.5, reaching a minimum of about -1.1 at x=5, and returning to zero at x=7. The other signal is a yellow square wave that follows the same pattern but is scaled by the factor 'x' (approximately 1.5 times larger).

At the bottom of the screen, there are status bars for 'Ready', '200%', and 'auto(FixedStepDiscrete)'. The overall interface is dark-themed.

Experiments with blocks and settings

SIMULATION DEBUG MODELING FORMAT APPS

untitled1 *

untitled1

Stop Time: 10
Normal
Step Back
Run
Step Forward
Stop
Data Inspector
Simulation Manager

FILE LIBRARY PREPARE REVIEW RESULTS

Library Browser

Space state

untitled1

Model Browser

Operator

- Mux
- Out1
- Product
- Relational Operator
- Saturation
- Scope
- Subsystem
- Sum
- Switch
- Terminator

Product

```
graph LR; A[2] --> P1[Product]; B[4] --> P1; P1 --> C[8]
```

Addition

```
graph LR; A[2] --> S1[Sum]; B[4] --> S1; S1 --> C[6]
```

Division

```
graph LR; A[2] --> D1[Divide]; B[4] --> D1; D1 --> C[0.5]
```

Subtraction

```
graph LR; A[2] --> S2[Subtract]; B[4] --> S2; S2 --> C[-2]
```

Multiple inputs

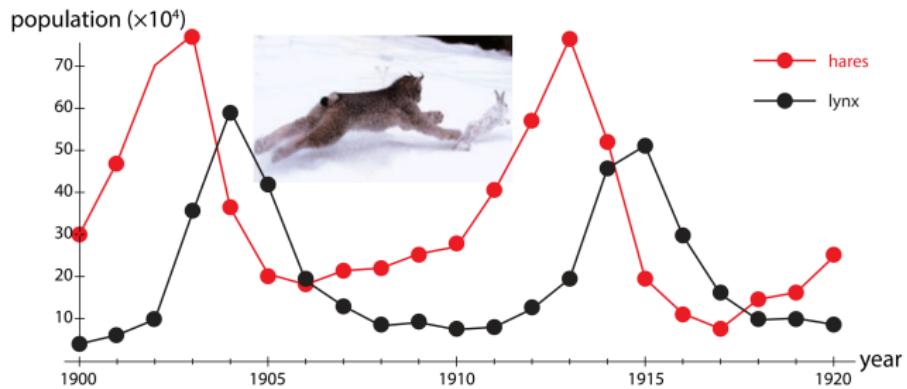
```
graph LR; S3[Sum] --> Out1[ ]; S3 --> Out2[ ]
```

```
graph LR; S4[Sum] --> Out3[ ]
```

125%

auto/FixedStepDiscrete

Predator-Prey



Overview

- ▶ The Lotka-Volterra equations, also known as the predator-prey equations, model the dynamics of biological systems in which two species interact, one as a **predator** and the other as **prey**.
- ▶ They were proposed independently by Alfred J. Lotka in 1925 and Vito Volterra in 1926.
- ▶ Model is characterized by **oscillations** in the population size of both predator and prey,
- ▶ Peak of the predator's oscillation lags behind the peak of the prey's oscillation.

Assumptions

The model makes several simplifying assumptions but captures the essential dynamics of predator-prey interactions:

Assumptions

The model makes several simplifying assumptions but captures the essential dynamics of predator-prey interactions:

- ▶ the prey population will grow exponentially when the predator is absent (unrealistic in a long term but may be precise for a long period)

Assumptions

The model makes several simplifying assumptions but captures the essential dynamics of predator-prey interactions:

- ▶ the prey population will grow exponentially when the predator is absent (unrealistic in a long term but may be precise for a long period)
- ▶ the predator population will starve in the absence of the prey population — there is no option of switching to another type of prey (may be realistic for some circumstances)

Assumptions

The model makes several simplifying assumptions but captures the essential dynamics of predator-prey interactions:

- ▶ the prey population will grow exponentially when the predator is absent (unrealistic in a long term but may be precise for a long period)
- ▶ the predator population will starve in the absence of the prey population — there is no option of switching to another type of prey (may be realistic for some circumstances)
- ▶ predators can consume any quantities of prey (unrealistic for large quantities)

Assumptions

The model makes several simplifying assumptions but captures the essential dynamics of predator-prey interactions:

- ▶ the prey population will grow exponentially when the predator is absent (unrealistic in a long term but may be precise for a long period)
- ▶ the predator population will starve in the absence of the prey population — there is no option of switching to another type of prey (may be realistic for some circumstances)
- ▶ predators can consume any quantities of prey (unrealistic for large quantities)
- ▶ both populations are moving randomly through a homogeneous, static, environment (may be approximately true for some environments)

The Mathematical Model

The Lotka-Volterra equations are a pair of first-order, nonlinear, differential equations given by:

$$\frac{dN}{dt} = rN - \beta NP \quad (1)$$

$$\frac{dP}{dt} = m\beta NP - dP \quad (2)$$

The Mathematical Model

The Lotka-Volterra equations are a pair of first-order, nonlinear, differential equations given by:

$$\frac{dN}{dt} = rN - \beta NP \quad (1)$$

$$\frac{dP}{dt} = m\beta NP - dP \quad (2)$$

where:

- ▶ N is population number of prey (e.g., rabbits),

The Mathematical Model

The Lotka-Volterra equations are a pair of first-order, nonlinear, differential equations given by:

$$\frac{dN}{dt} = rN - \beta NP \quad (1)$$

$$\frac{dP}{dt} = m\beta NP - dP \quad (2)$$

where:

- ▶ N is population number of prey (e.g., rabbits),
- ▶ P is the number of predators (e.g., foxes),
- ▶ r is the prey growth rate

The Mathematical Model

The Lotka-Volterra equations are a pair of first-order, nonlinear, differential equations given by:

$$\frac{dN}{dt} = rN - \beta NP \quad (1)$$

$$\frac{dP}{dt} = m\beta NP - dP \quad (2)$$

where:

- ▶ N is population number of prey (e.g., rabbits),
- ▶ P is the number of predators (e.g., foxes),
- ▶ r is the prey growth rate
- ▶ d is the death rate of predators

The Mathematical Model

The Lotka-Volterra equations are a pair of first-order, nonlinear, differential equations given by:

$$\frac{dN}{dt} = rN - \beta NP \quad (1)$$

$$\frac{dP}{dt} = m\beta NP - dP \quad (2)$$

where:

- ▶ N is population number of prey (e.g., rabbits),
- ▶ P is the number of predators (e.g., foxes),
- ▶ r is the prey growth rate
- ▶ d is the death rate of predators
- ▶ m is the proportionality constant

The Mathematical Model

The Lotka-Volterra equations are a pair of first-order, nonlinear, differential equations given by:

$$\frac{dN}{dt} = rN - \beta NP \quad (1)$$

$$\frac{dP}{dt} = m\beta NP - dP \quad (2)$$

where:

- ▶ N is population number of prey (e.g., rabbits),
- ▶ P is the number of predators (e.g., foxes),
- ▶ r is the prey growth rate
- ▶ d is the death rate of predators
- ▶ m is the proportionality constant
- ▶ β attack efficiency of the predator

Prey equations

The prey population is given by:

$$\frac{dN}{dt} = rN - \beta NP \quad (3)$$

Prey equations

The prey population is given by:

$$\frac{dN}{dt} = rN - \beta NP \quad (3)$$

In the absence of predators, the equation becomes:

$$\frac{dN}{dt} = rN \quad (4)$$

which represents the Malthusian model. The prey population would grow exponentially with the growth rate r .

Prey equations

The prey population is given by:

$$\frac{dN}{dt} = rN - \beta NP \quad (3)$$

In the absence of predators, the equation becomes:

$$\frac{dN}{dt} = rN \quad (4)$$

which represents the Malthusian model. The prey population would grow exponentially with the growth rate r .

Item The reduction in prey population due to predators is captured by the expression:

$$-\beta NP \quad (5)$$

Prey equations

The prey population is given by:

$$\frac{dN}{dt} = rN - \beta NP \quad (3)$$

In the absence of predators, the equation becomes:

$$\frac{dN}{dt} = rN \quad (4)$$

which represents the Malthusian model. The prey population would grow exponentially with the growth rate r .

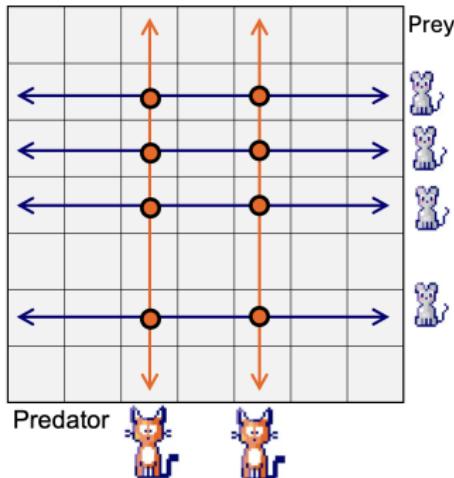
Item The reduction in prey population due to predators is captured by the expression:

$$-\beta NP \quad (5)$$

Here, NP is proportional to the number of encounters. An increased number of predators, an increased prey population, and greater searching efficiency of the predators would all lead to a higher removal rate of prey. β is attack efficiency of the predator. Higher β means higher probability of prey being killed.

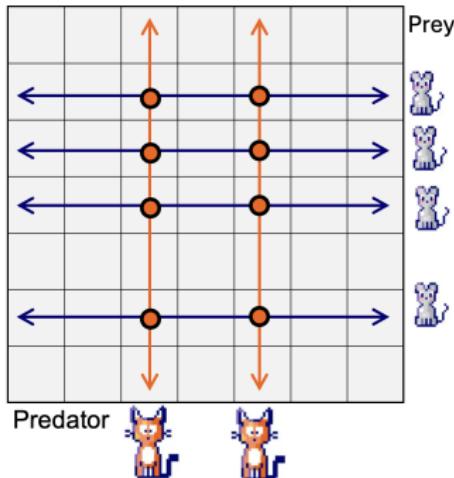
Predator-prey encounters

Picture below illustrates the concept of predators-prey encounters as $N \cdot P$ product.



Predator-prey encounters

Picture below illustrates the concept of predators-prey encounters as $N \cdot P$ product.



The concept would not work for predators hunting in packs or for prey staying in herds.

Predator equations

The predator population is given by:

$$\frac{dP}{dt} = m\beta NP - dP \quad (6)$$

Predator equations

The predator population is given by:

$$\frac{dP}{dt} = m\beta NP - dP \quad (6)$$

If the prey population N is 0, the equation becomes:

$$\frac{dN}{dt} = -dN \quad (7)$$

which represents dying of prey at exponential rate d .

Predator equations

The predator population is given by:

$$\frac{dP}{dt} = m\beta NP - dP \quad (6)$$

If the prey population N is 0, the equation becomes:

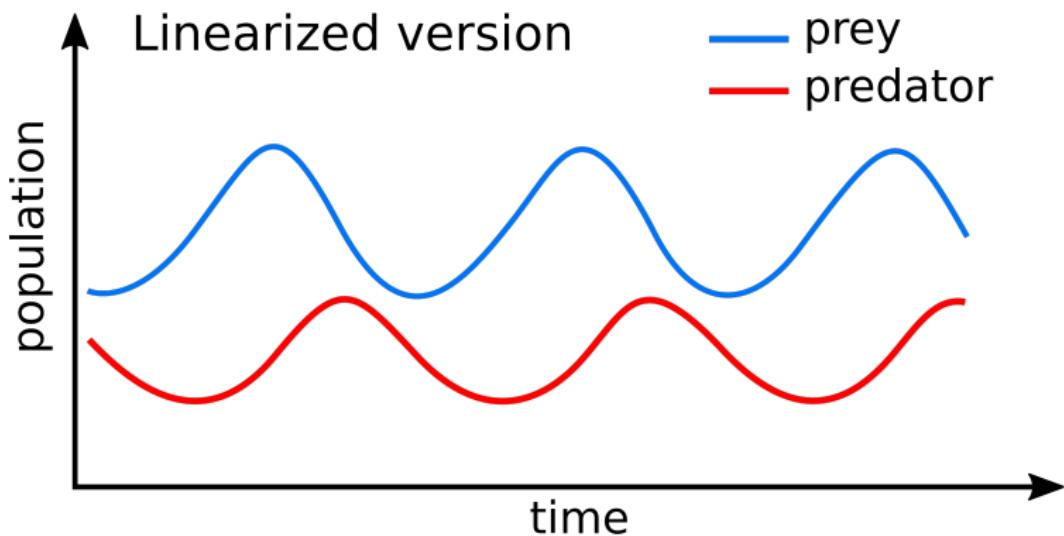
$$\frac{dN}{dt} = -dN \quad (7)$$

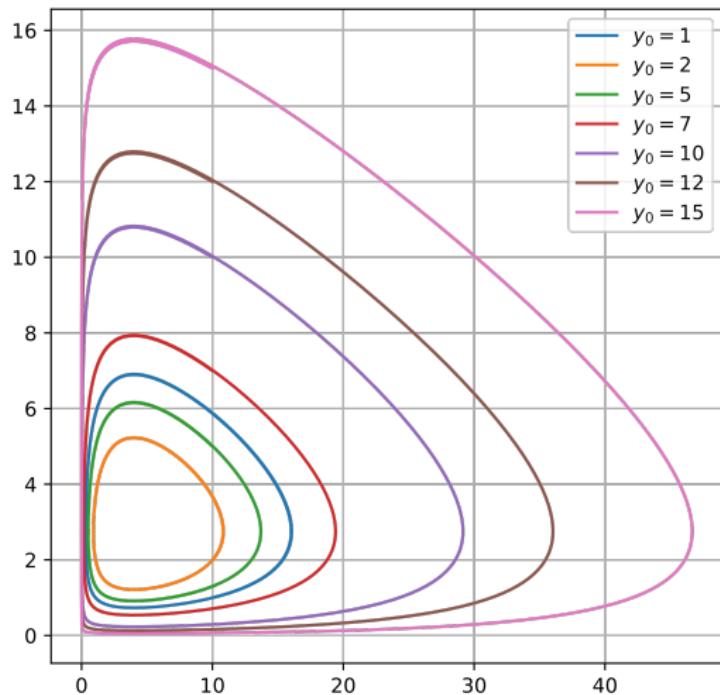
which represents dying of prey at exponential rate d .

The growth of predator population growth by eating of prey

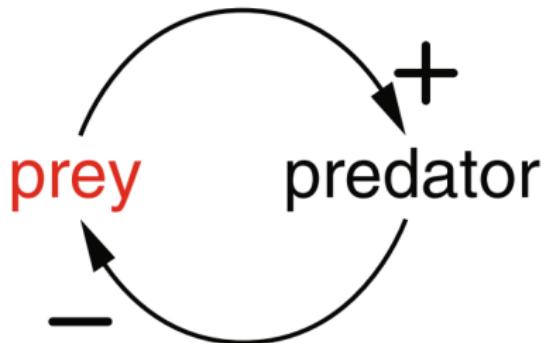
$$+m\beta NP \quad (8)$$

Here we see again βNP . Proportionality constant m tells how many prey individuals are required for increase of predator population by one.





Concept of feedback



Biological Implications

- ▶ The model predicts oscillations in the populations of predator and prey.
- ▶ The fluctuation of the two species is periodic,
- ▶ The amplitude and period of these oscillations depend on the initial conditions and the parameters of the model.
- ▶ The *average* numbers of the two species are constant
- ▶ If we destroy individuals of both species proportionately to their number, the average number of individuals of the prey grows and the average number of predators diminishes.

Conclusion

- ▶ The Lotka-Volterra predator-prey model provides a framework for understanding the dynamics of biological systems with predator-prey interactions.
- ▶ Despite its simplicity, it offers valuable insights into the oscillatory nature of ecological populations and has motivates further research into more complex and realistic models.

Exercise: Building a Predator-Prey Model in Simulink

Use Simulink to construct and analyze the Lotka-Volterra model.

1. Create a new Simulink model and build the system using two integrator blocks representing the populations of predators and prey.
2. Use function blocks to implement the differential equations for each population.
3. Set initial populations for predators and prey using constant blocks.

Experiment with the following parameters and observe the effects on the population dynamics.

1. Adjust the prey growth rate r to see how it affects the population cycles.
2. Study the impact the predator death rate d)
3. Modify the attack efficiency β and observe the changes in the frequency and amplitude of the oscillations.
4. Experiment with the conversion efficiency m

Plot predator and prey population in time and phase-space plot for the predator-prey system.