Modeling and Simulation — Lesson 2

# Feedback in the Nature
# Dynamics of Predator-Prey Interactions

Václav B. Petrák

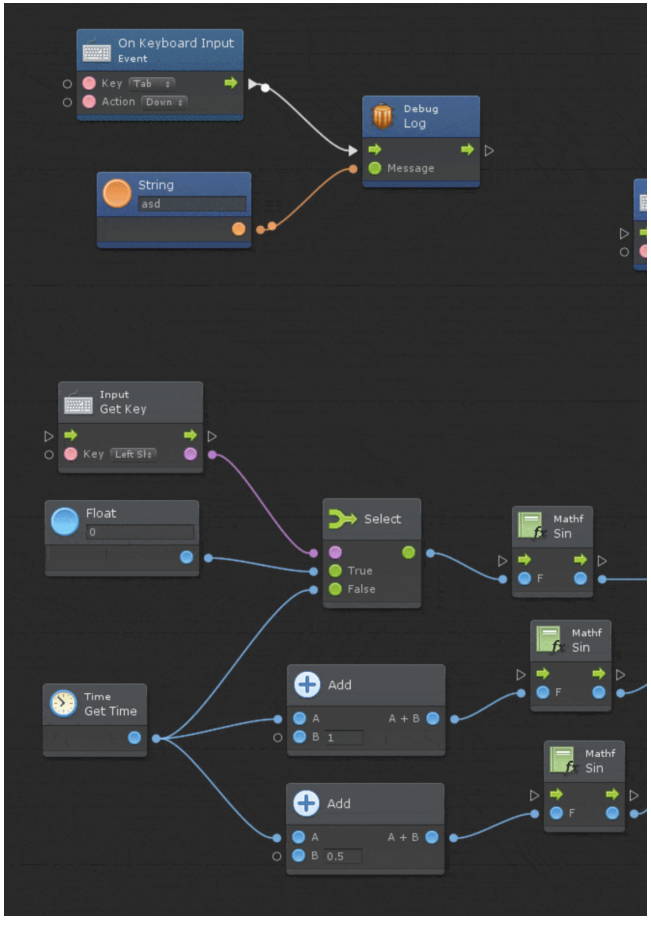
Faculty of Biomedical Engineering
Czech Technical University

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

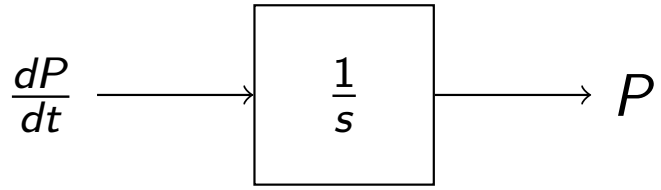
- **Simulink Standard Library:** Includes basic blocks for mathematical operations, logic, signal routing, and more. Essential for any model.
- **Simscape:** For physical modeling in domains such as mechanics, electrical, and fluids. Useful for simulating real-world system: Bateries, Electrical systems, Fluid, Mechanical applicatiosn
- **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 interring.

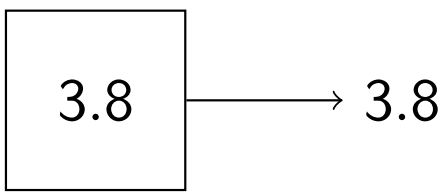
# Basic blocks: Integrator

- 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



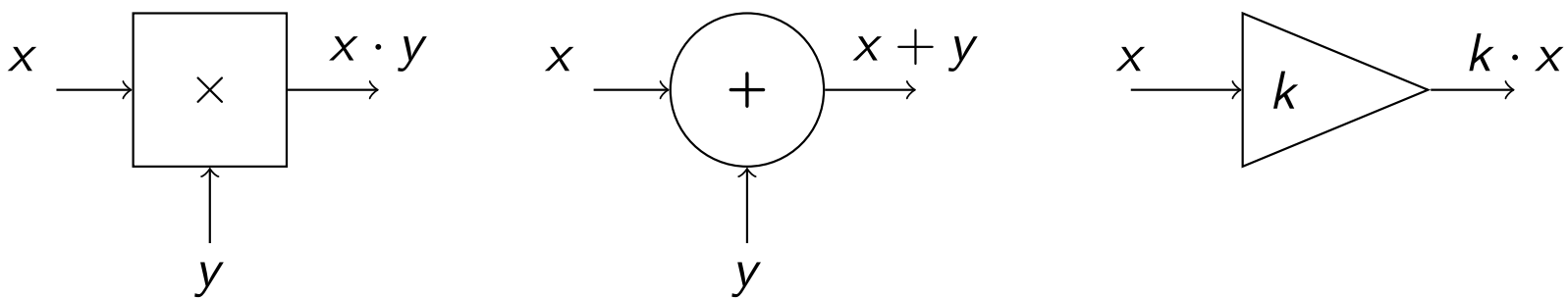
# Constant

- **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 commons 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

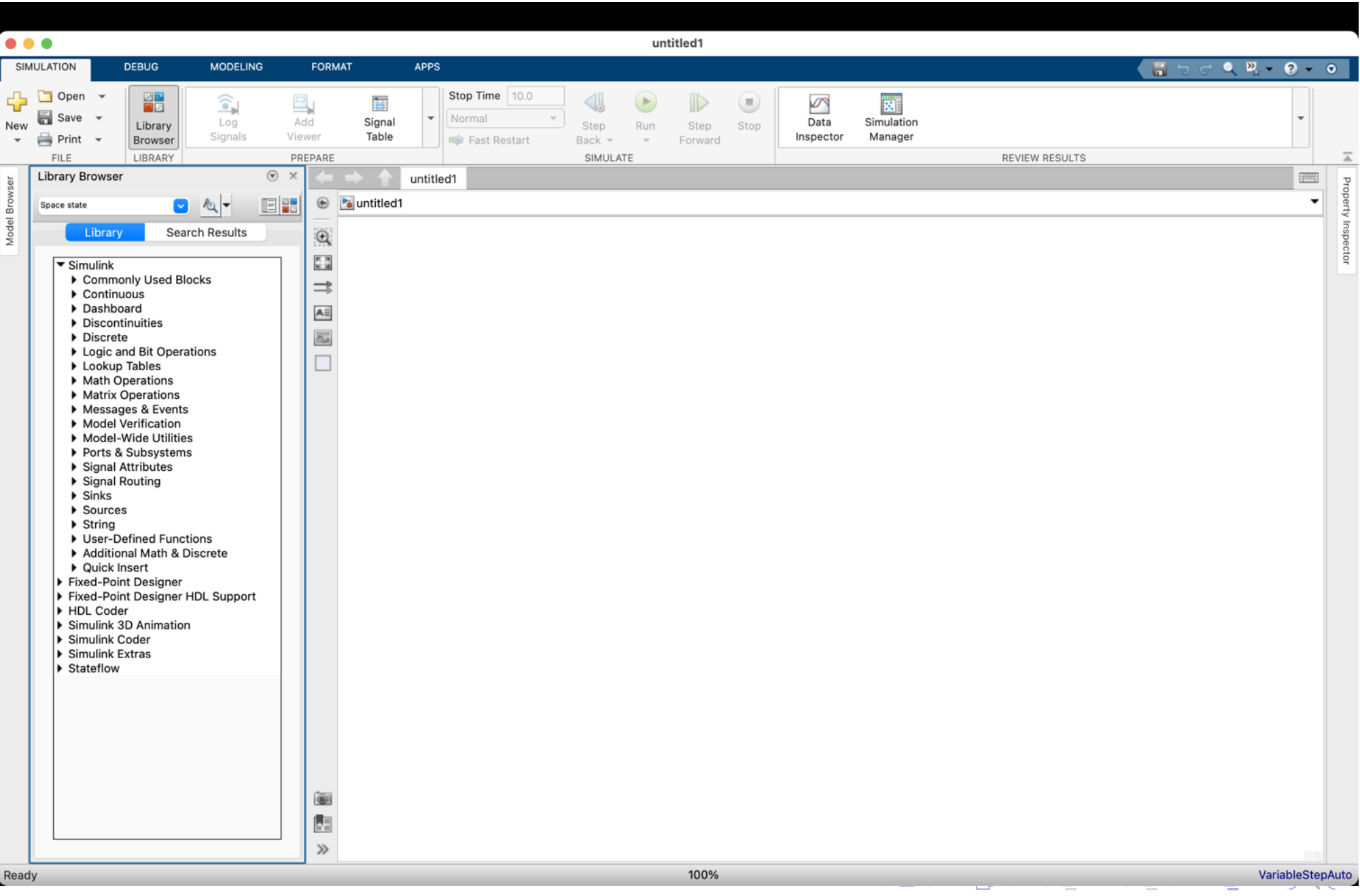
# First simulation

In this part you will learn how to

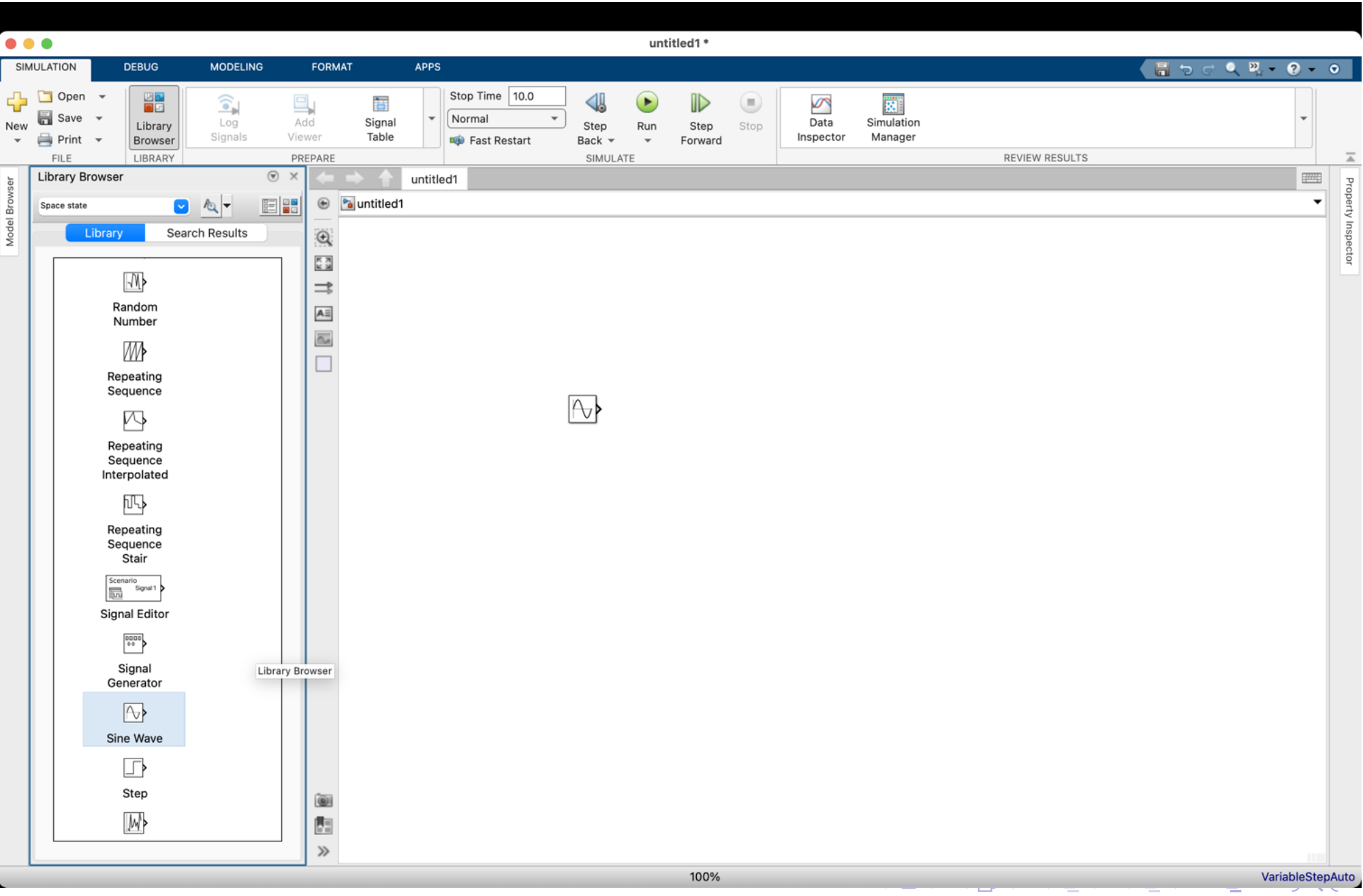
- Assemble your first model using library blocks
- Run a simulation
- Adjust simulation and display settings

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

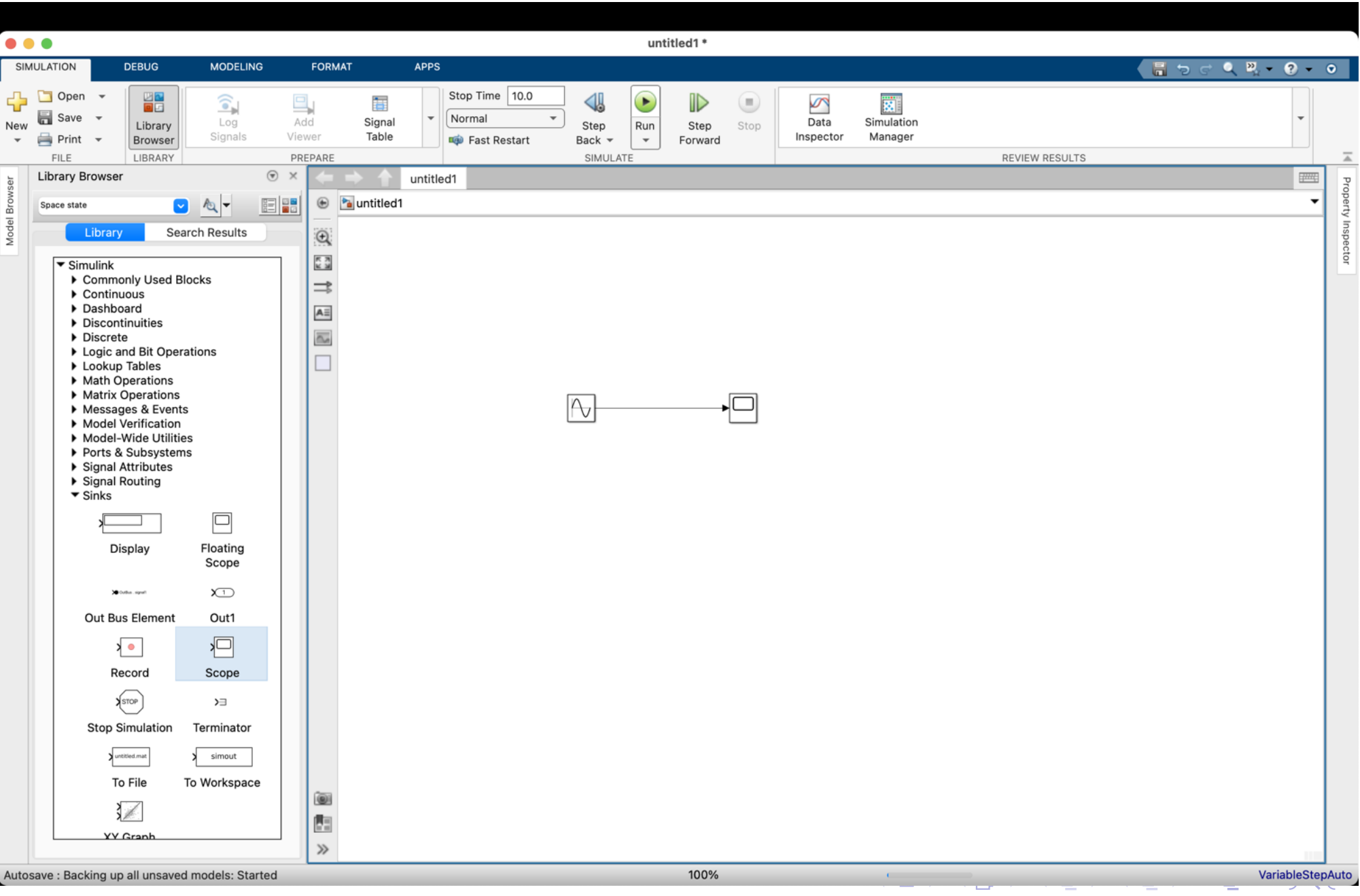
# Open Simulink environment



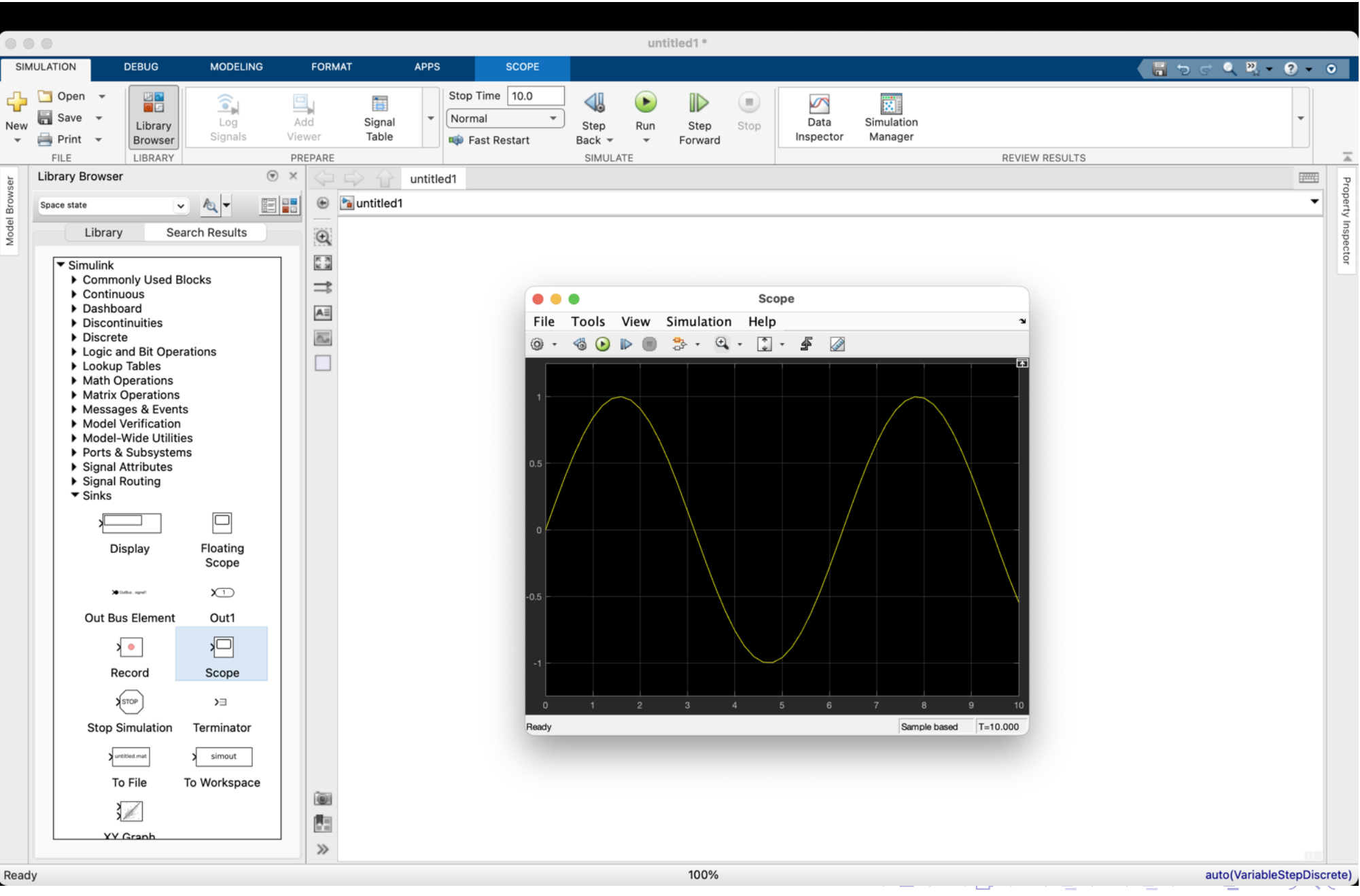
# Drag a *Sine wave* block from the library



# Add *Scope* and conncect blocs



# Run your first simulation



# Simulate for longer time

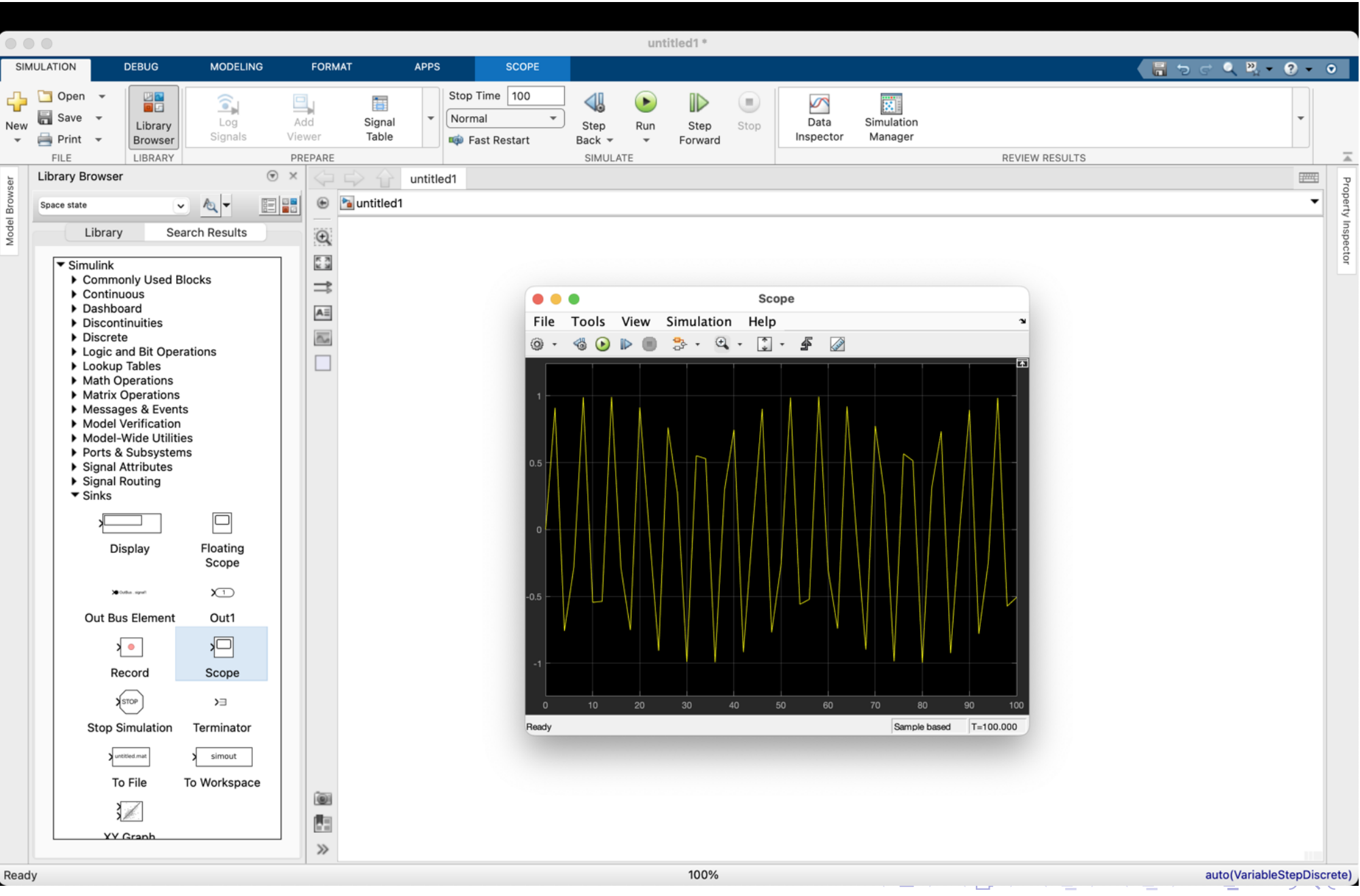
We now want to run the same simulation for longer time.

# 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.



# Problem with distotion

Resulting wave looks incorrect — Can you guess Why?.

# Problem with distotion

Resulting wave looks incorrect — Can you guess Why?.

The problem is distortion due to insufficient sampling frequency.

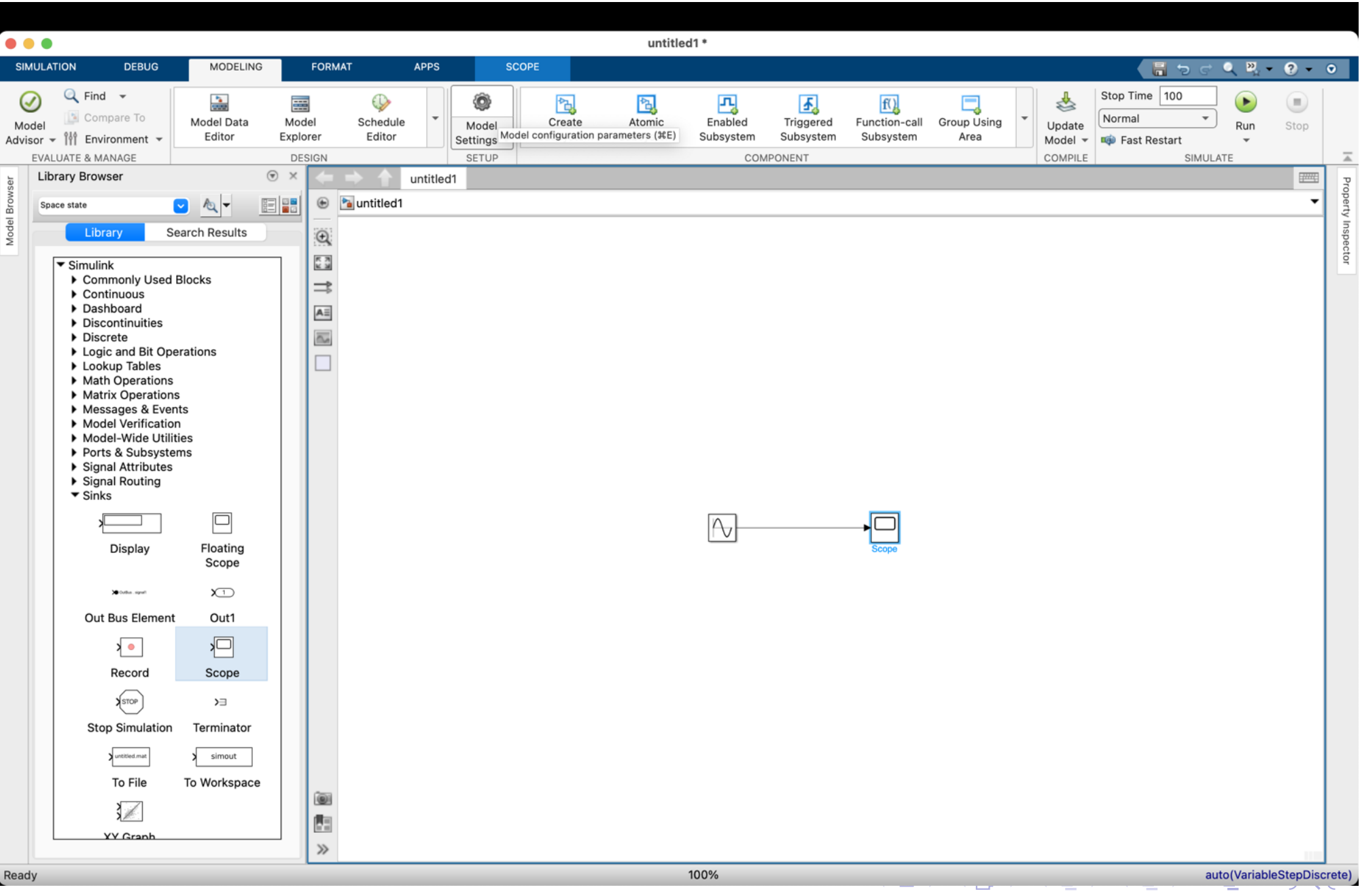
# Problem with distotion

Resulting wave looks incorrect — Can you guess Why?.

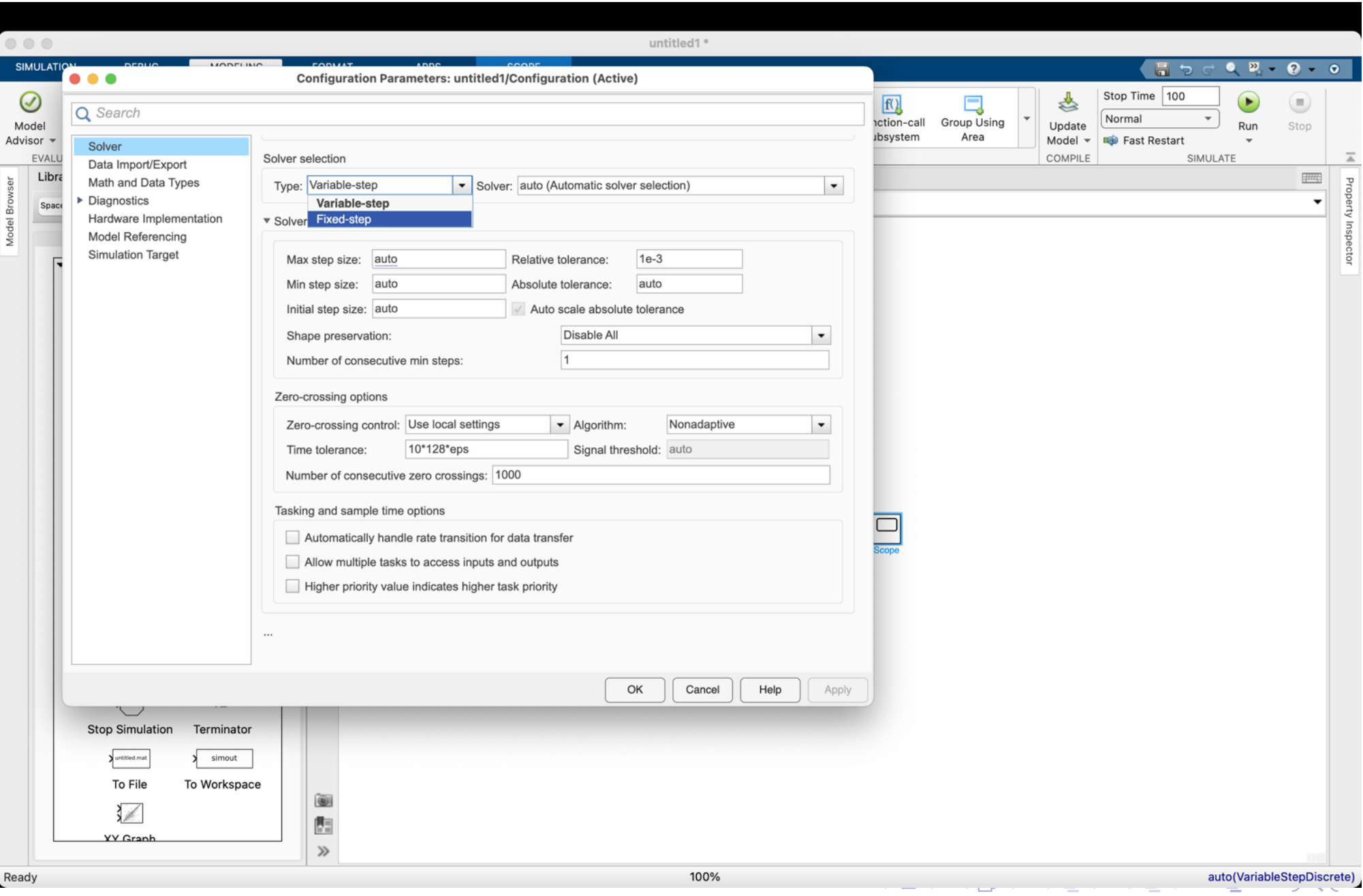
The problem is distortion due to insufficient sampling frequency.

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

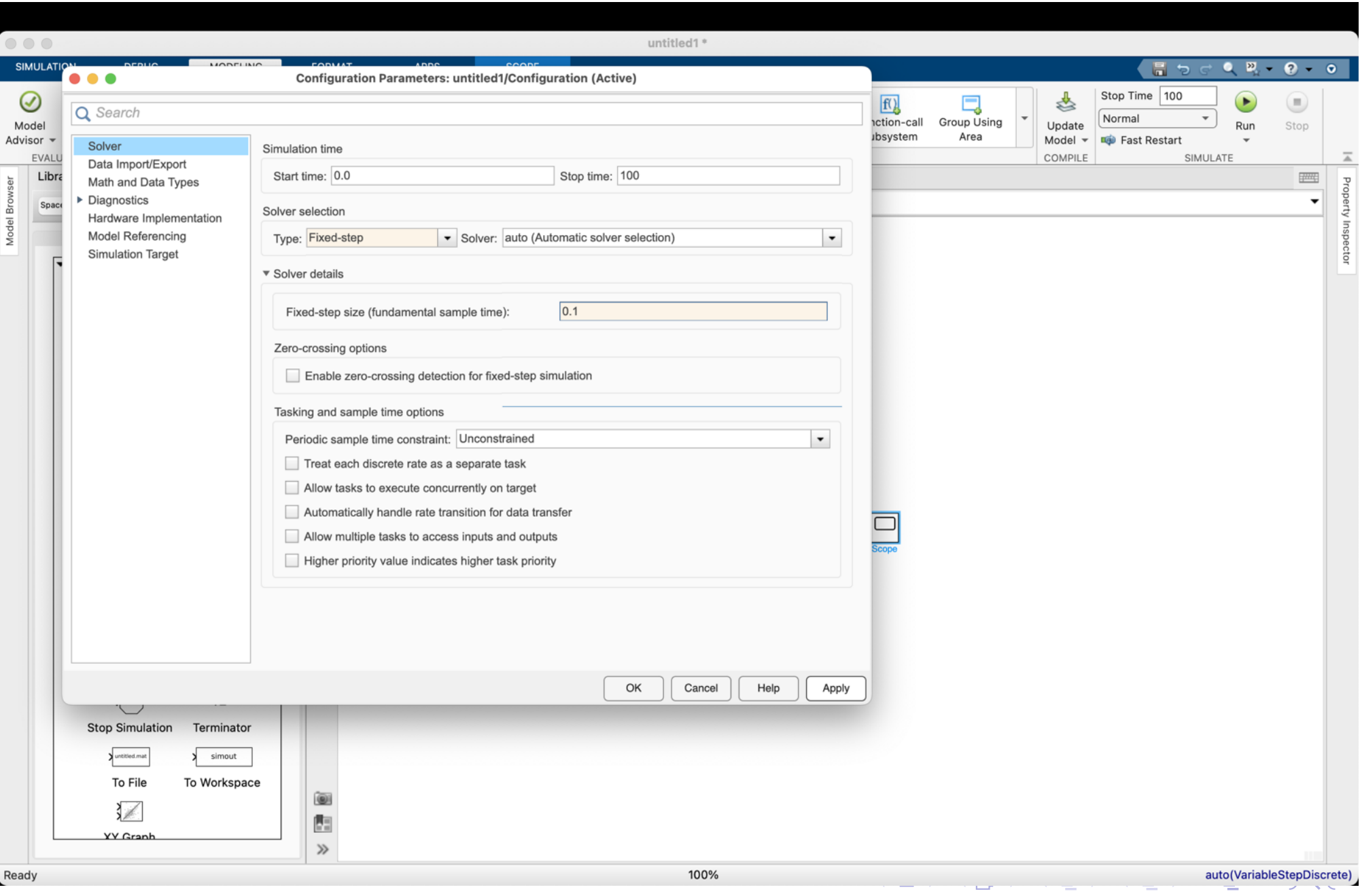
# Select *Modeling* and *Model Settings*



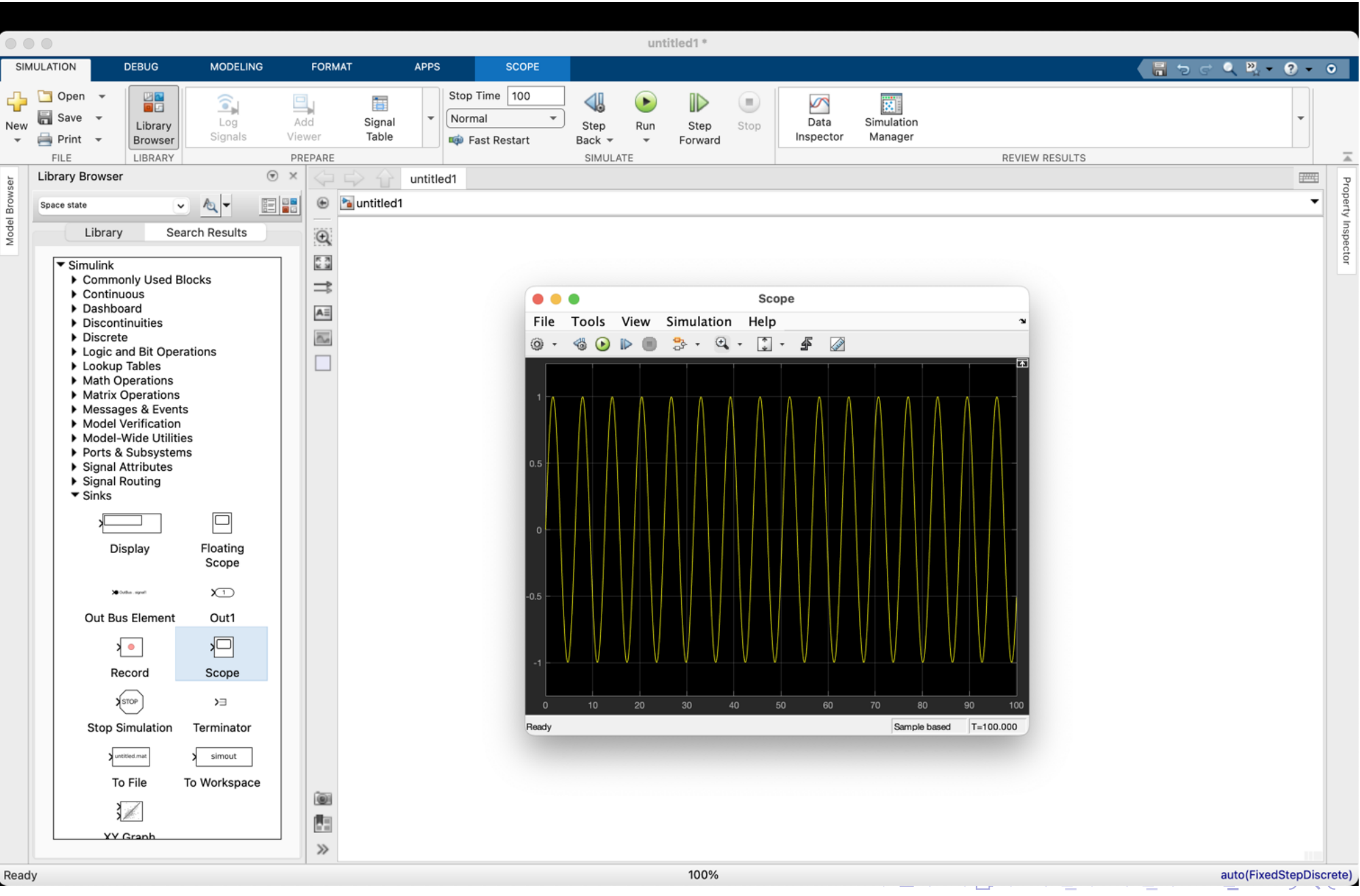
# In *Solver* menu select *Fixed-step* type



# Adjust *Fixed step* size to 0.1



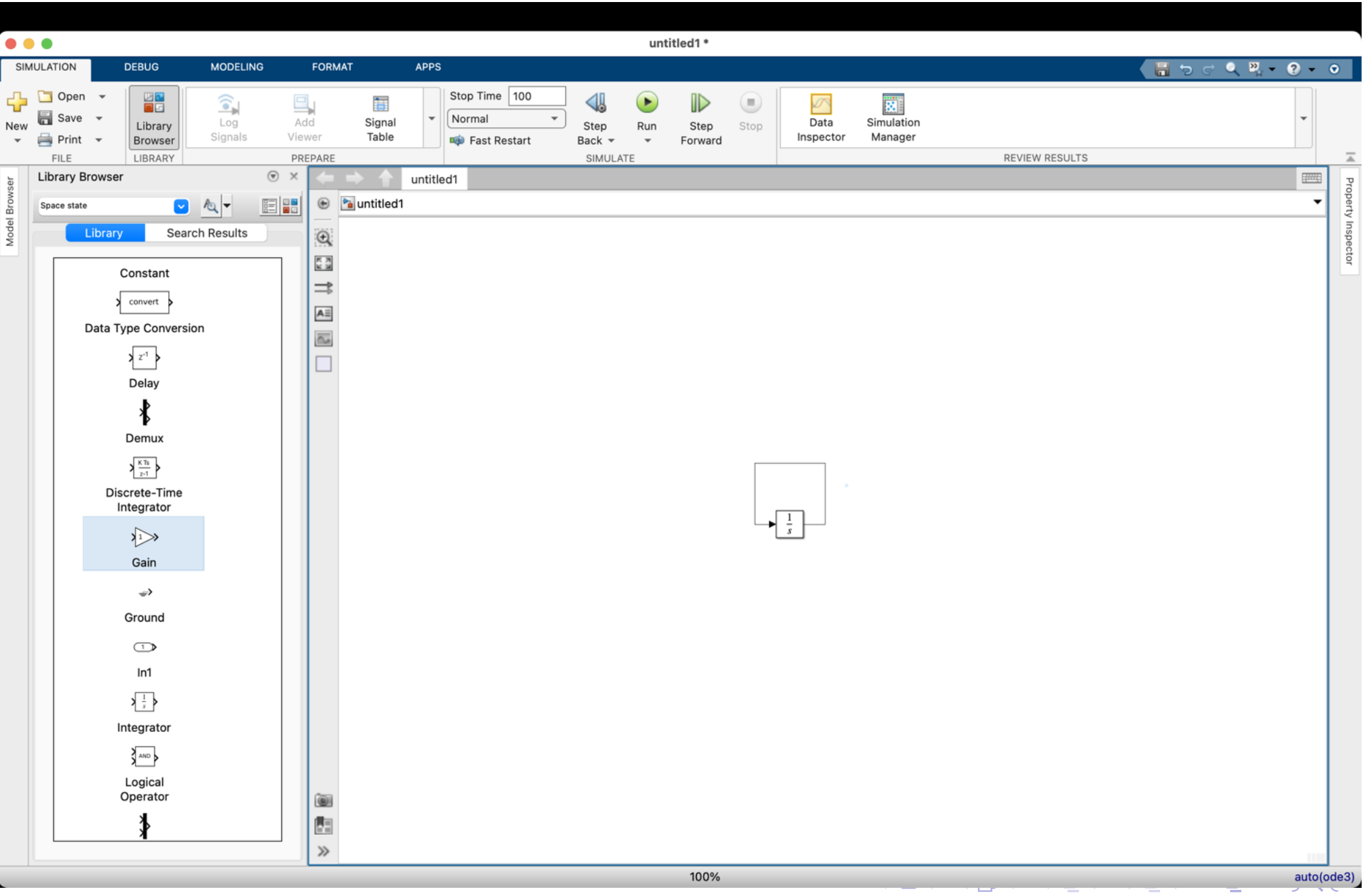
# The curve should look correctly now.



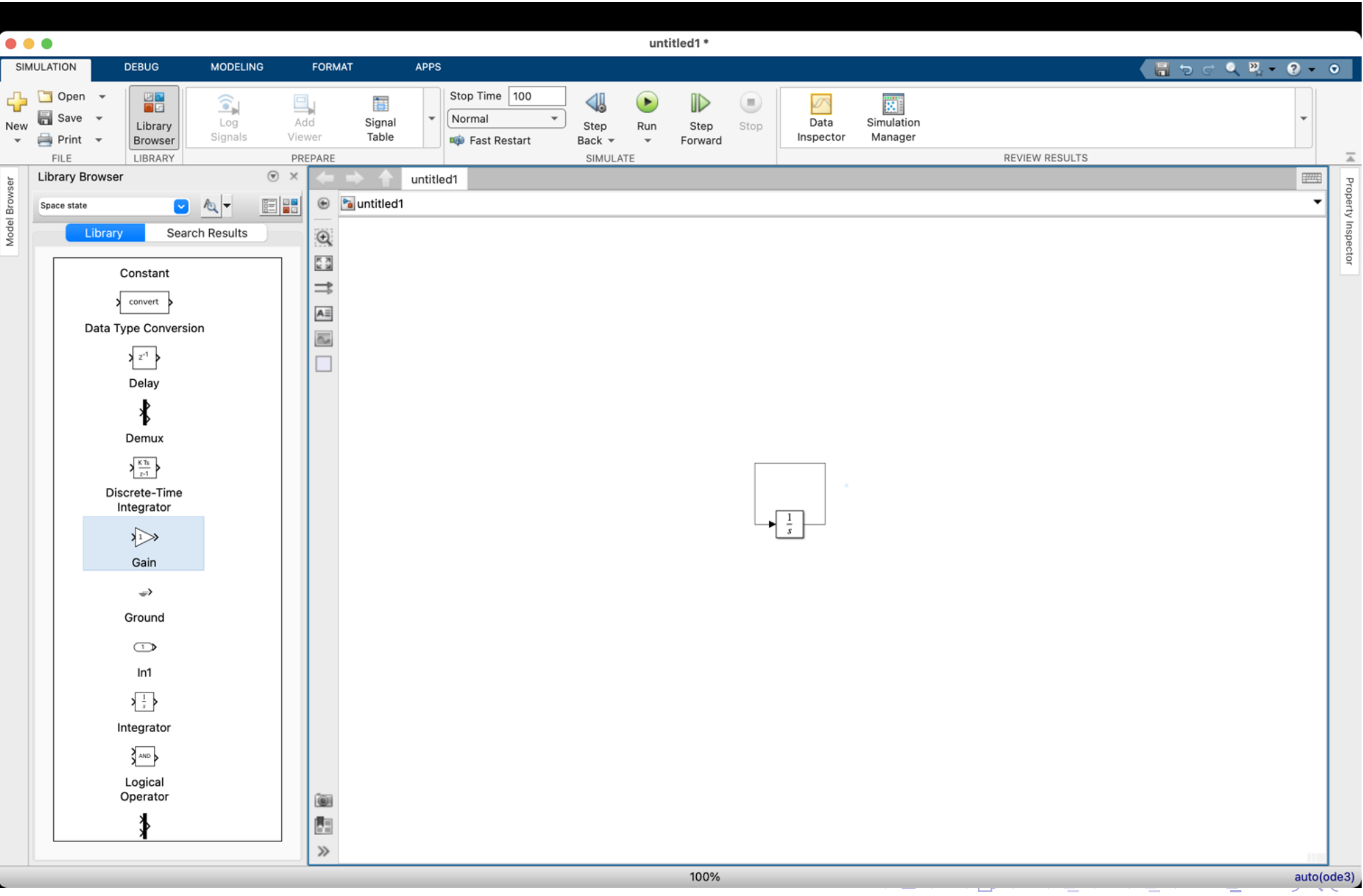
# Simulate for longer time

We will now build a Malthusian model.

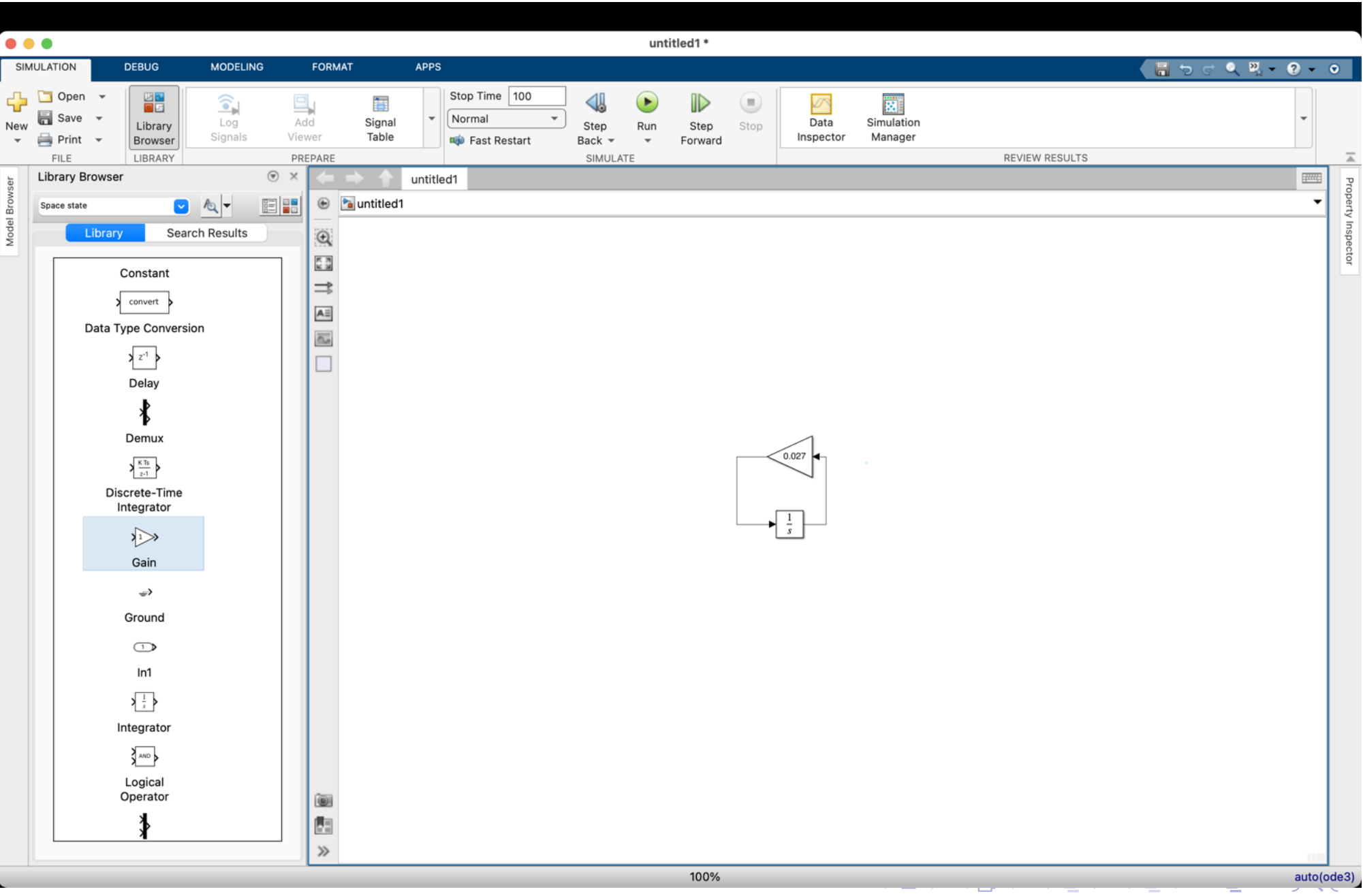
# The core of the model is an integrator



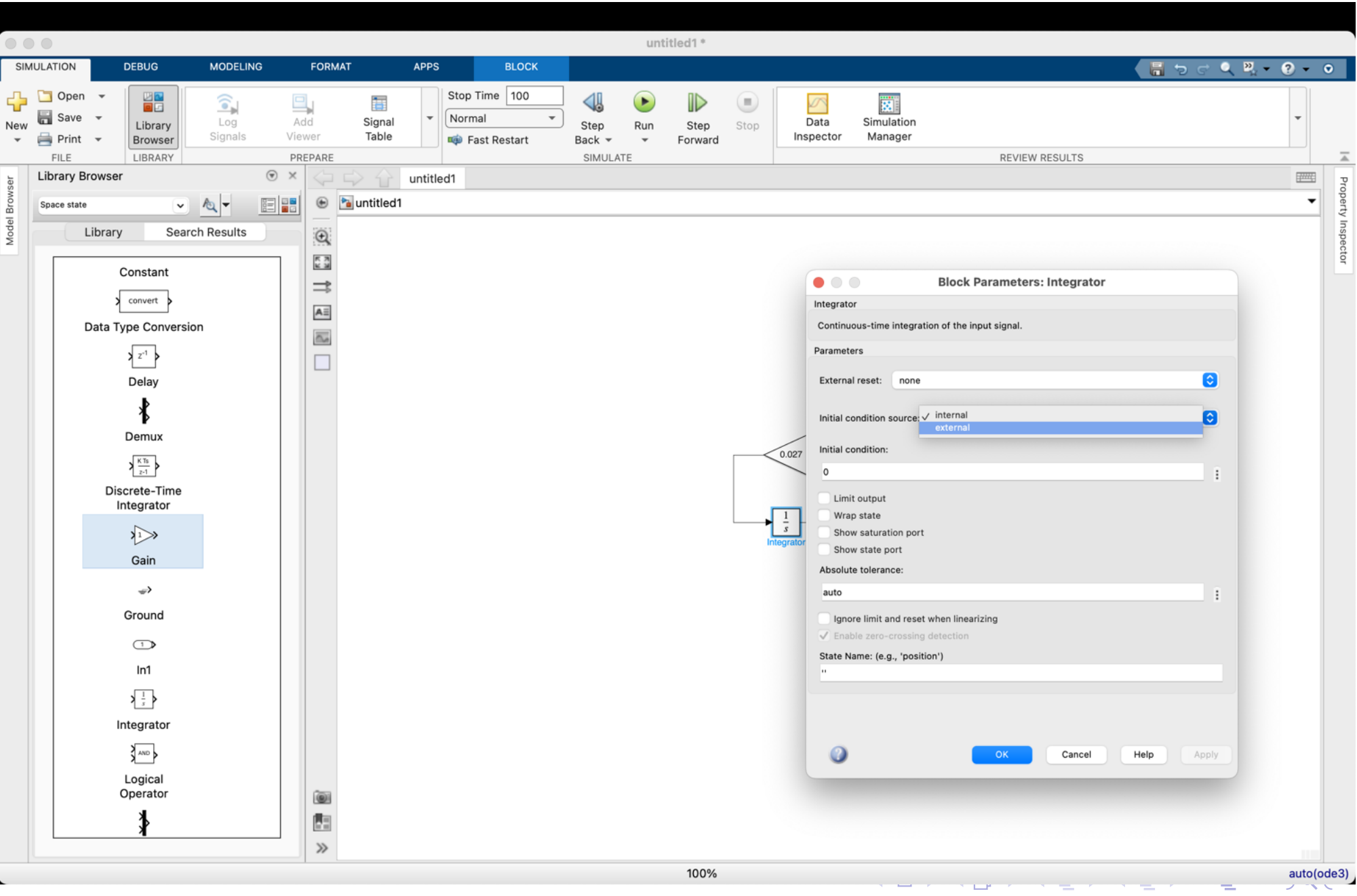
# The core of the model is an integrator



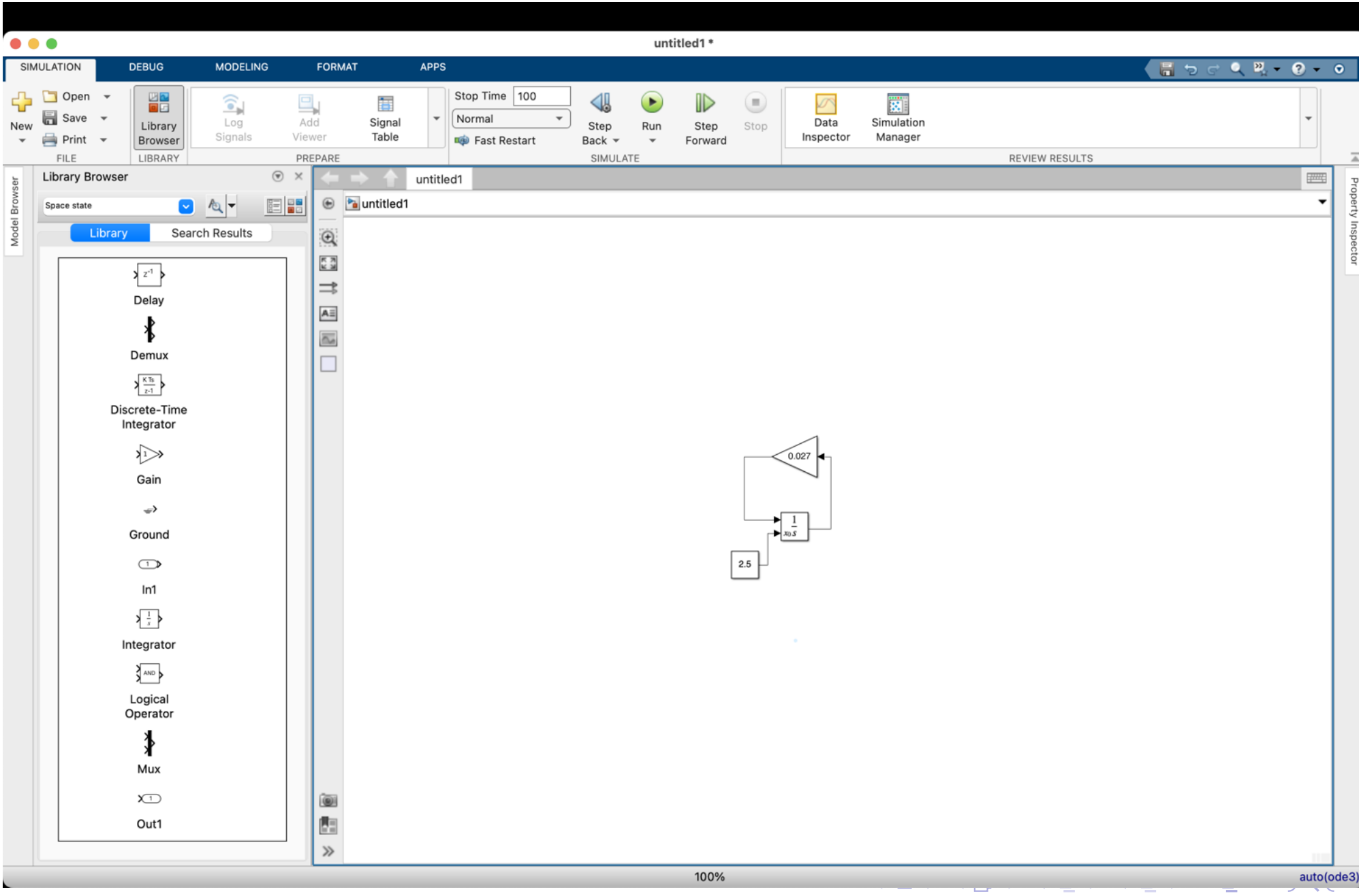
# Gain element adjust magnitude of the signal.



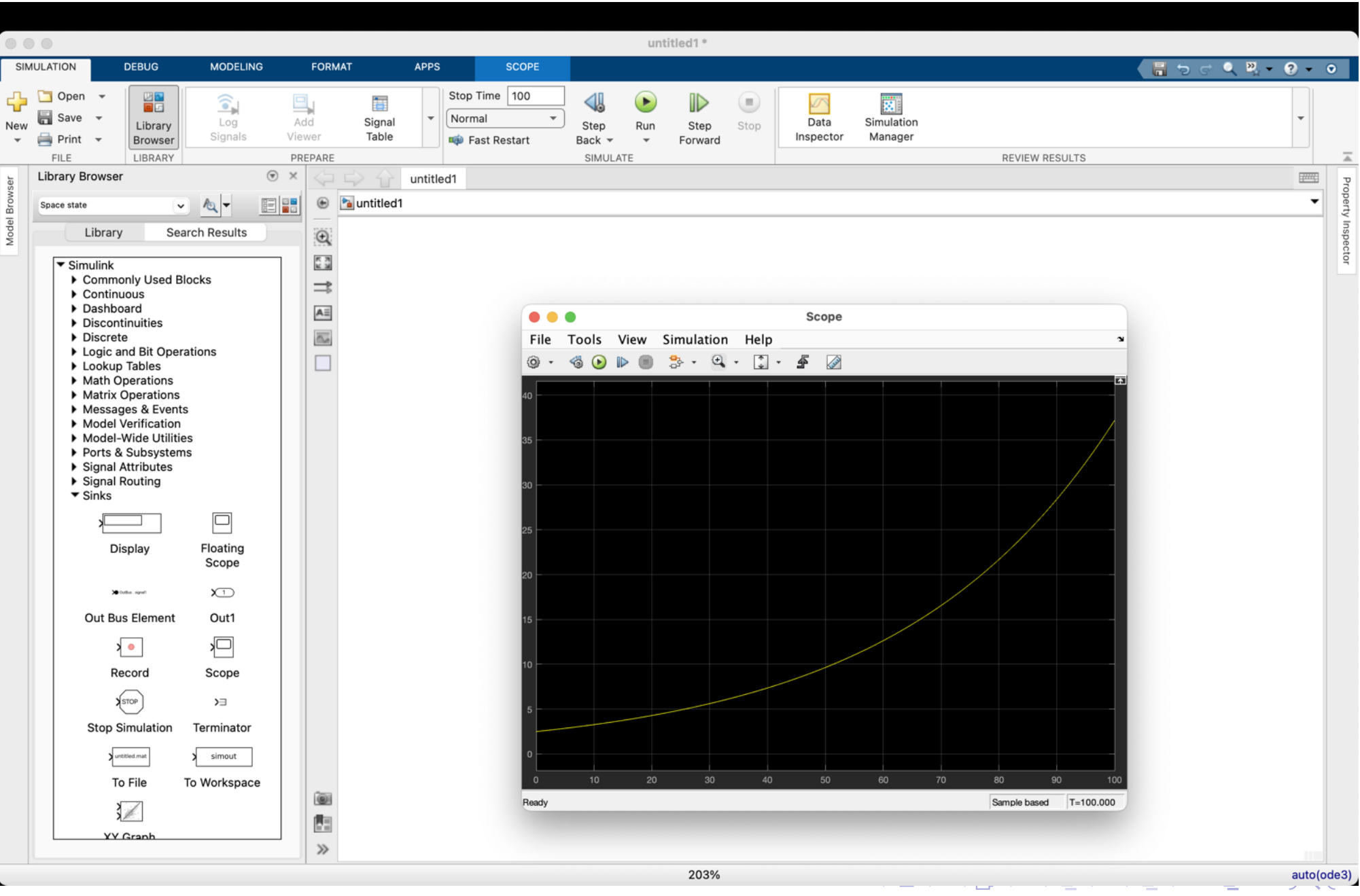
# Initial population can be a block parameter or external constant.



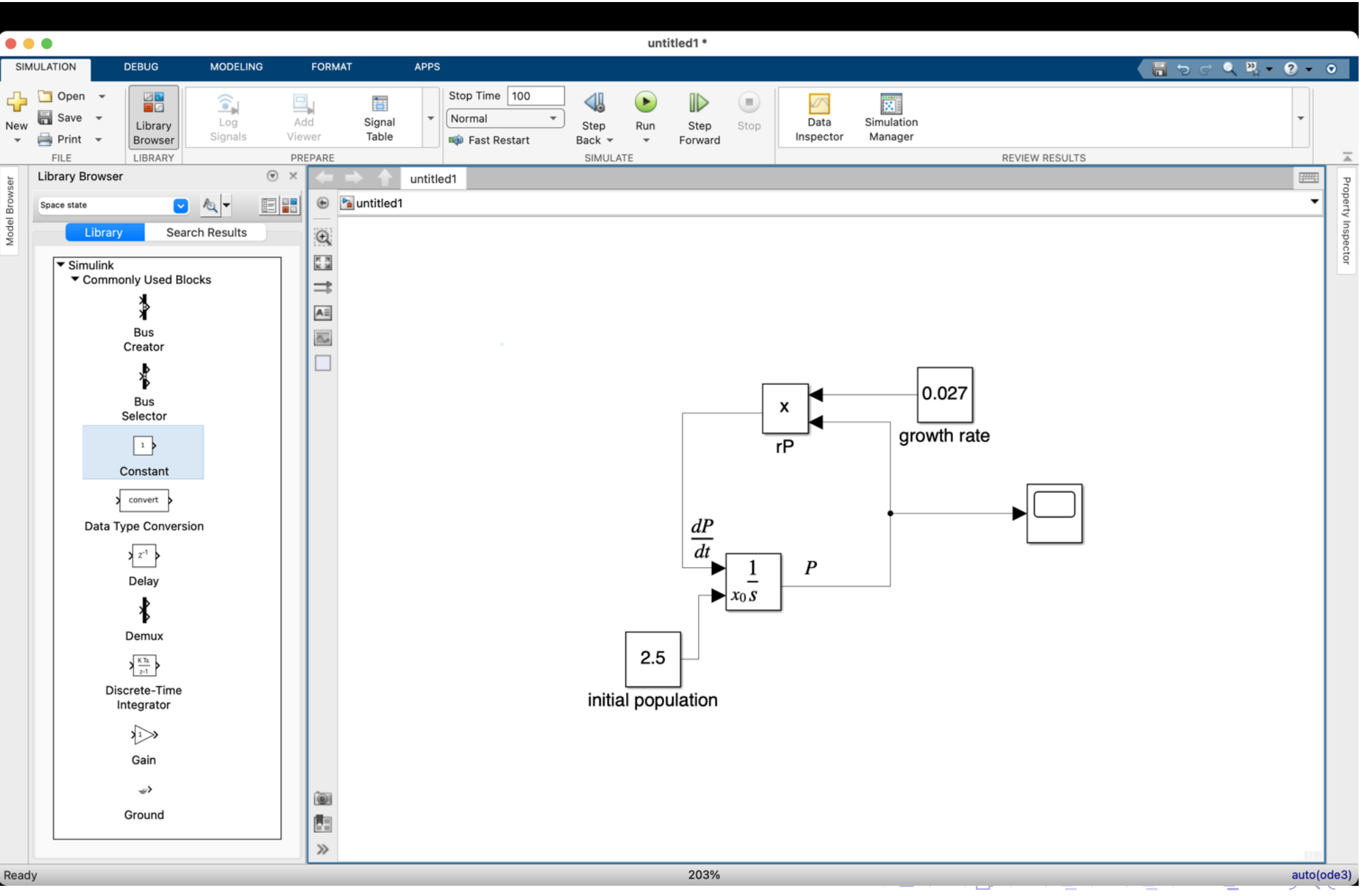
# Initial population can be a block parameter or external constant.



# We add display and labels



# Resulting population growth



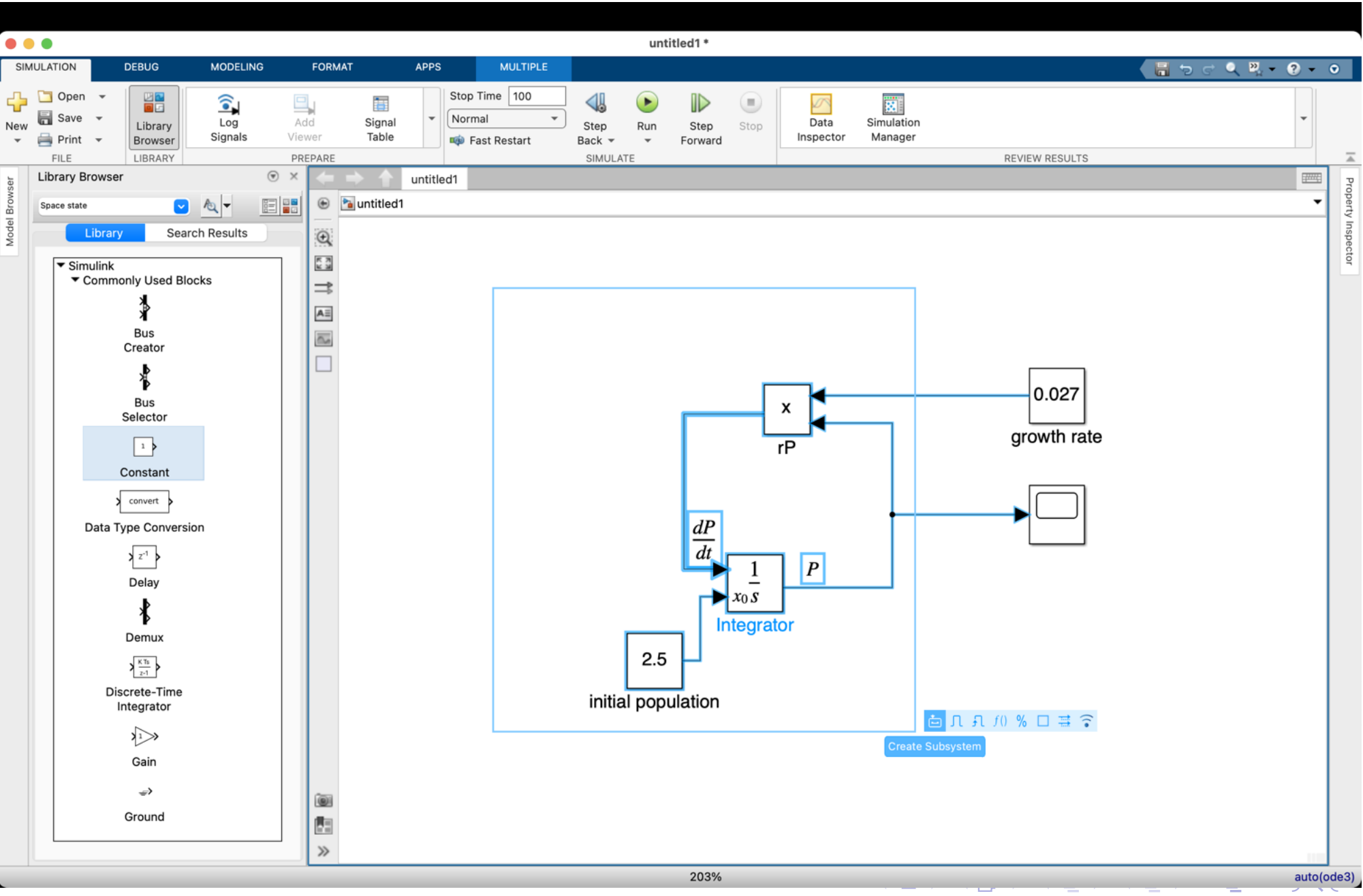
# Simulate for longer time

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

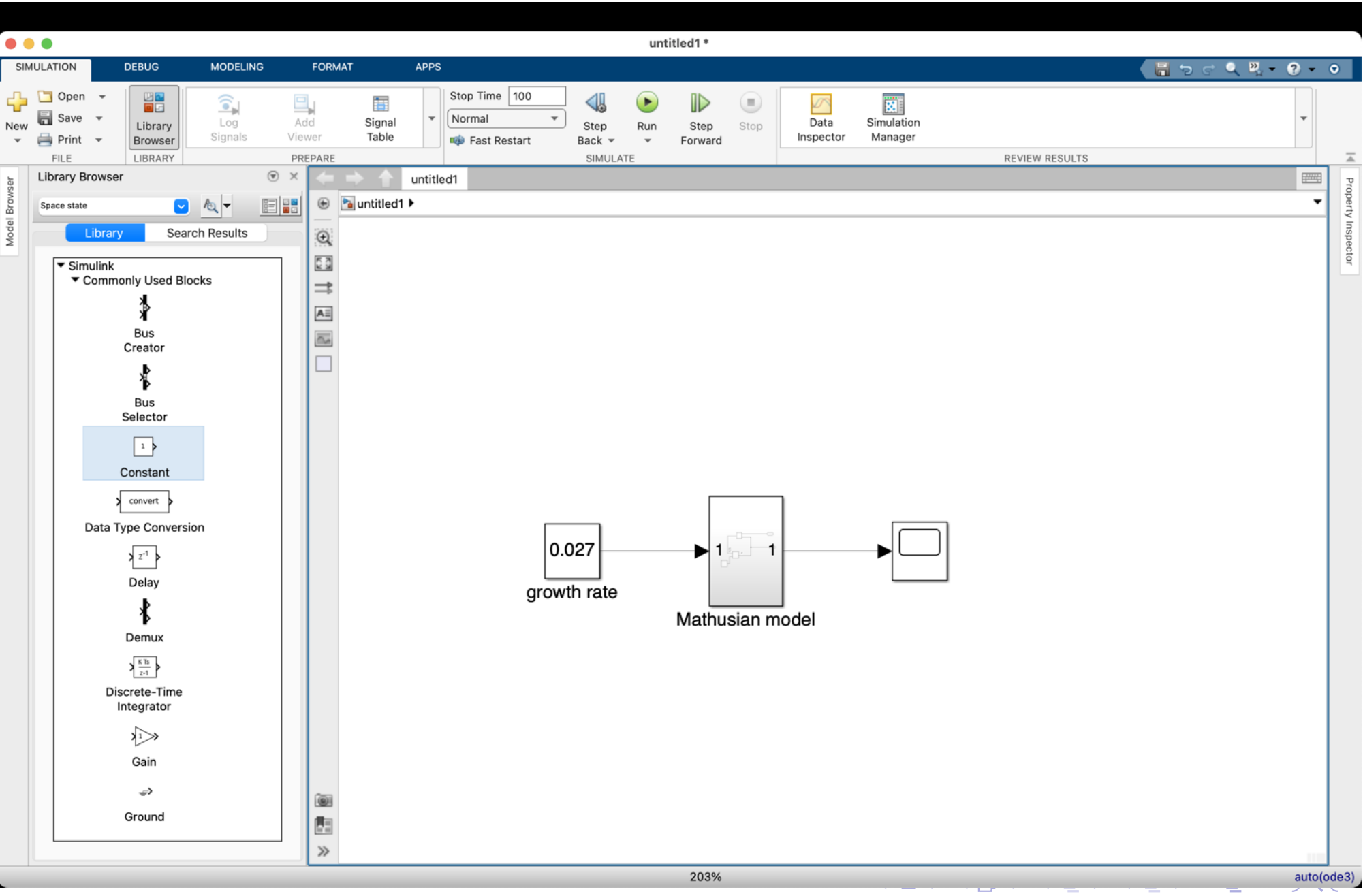
# Simulate for longer time

- 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

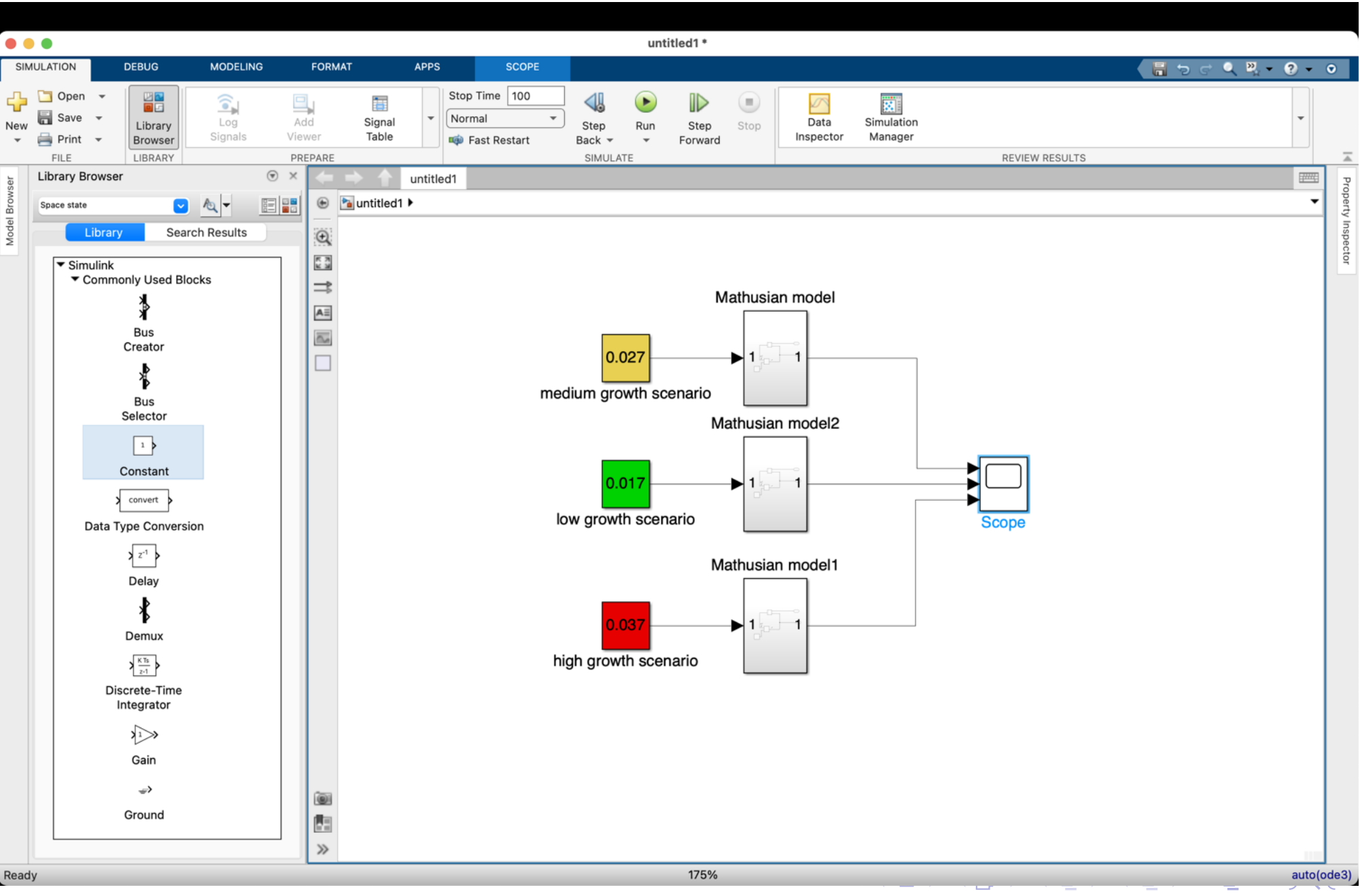
# We replace gain block and select relevant parts



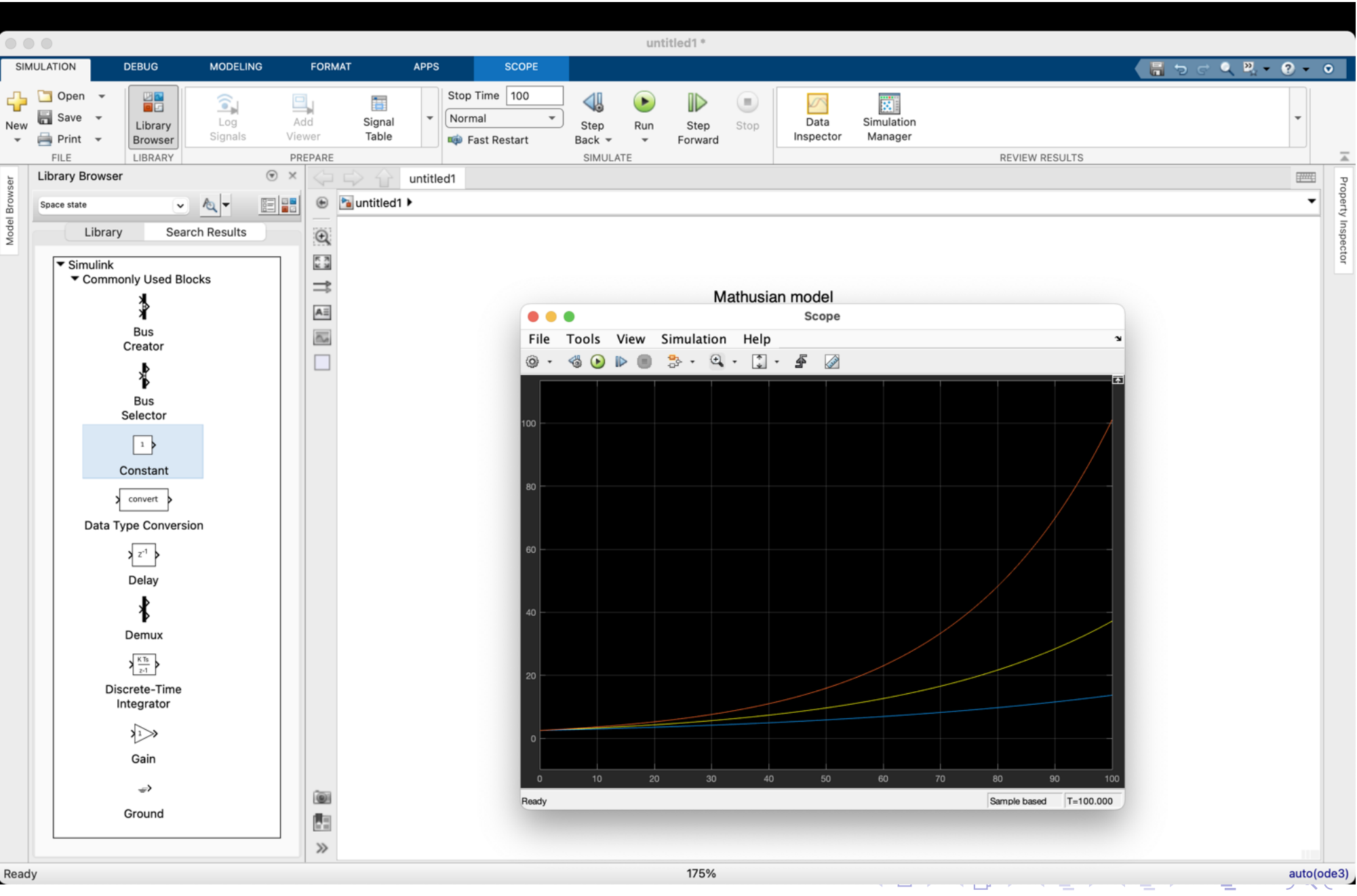
# Resulting subsystem



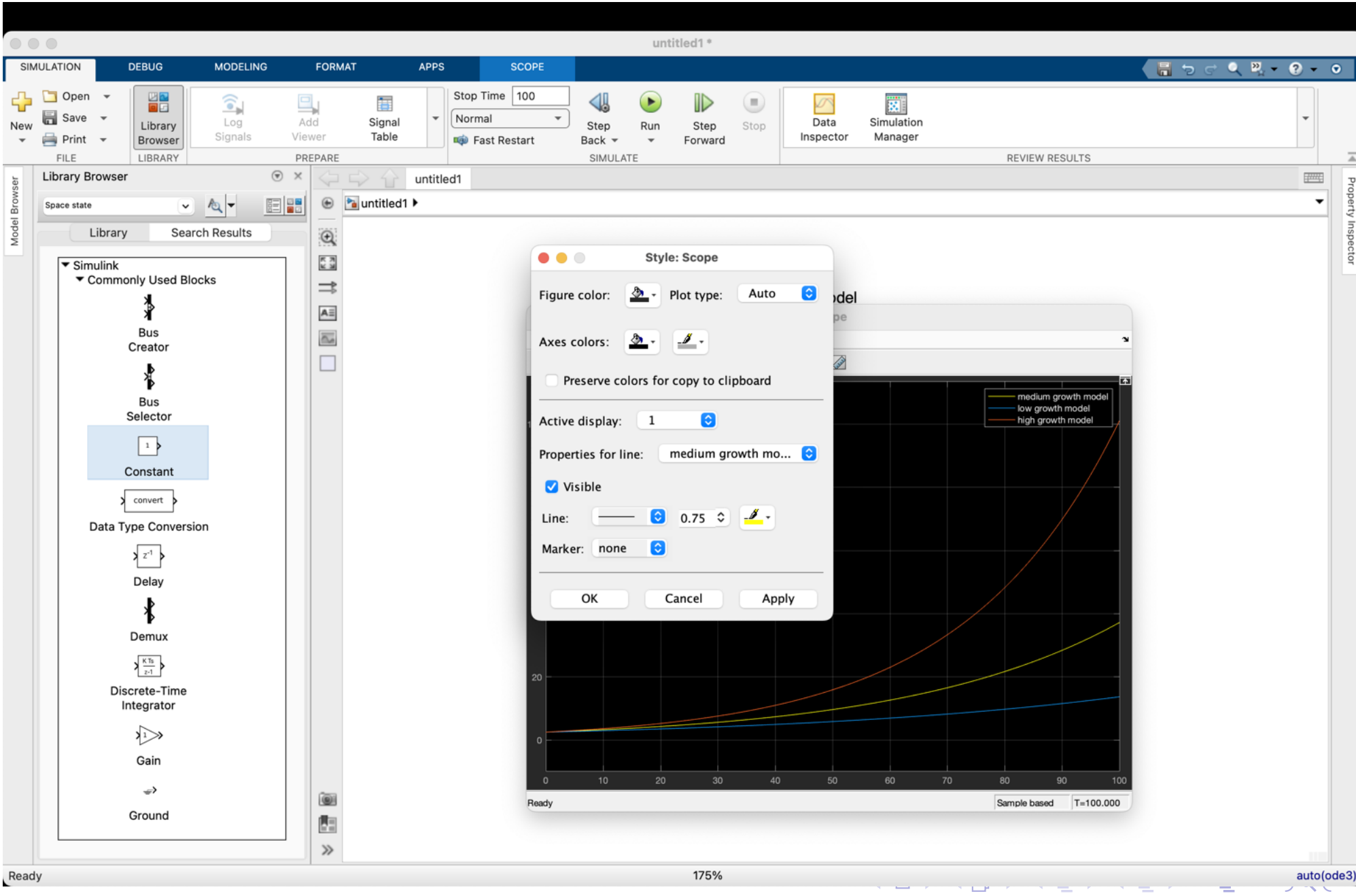
# We can duplicate subsystems for parallel simulations



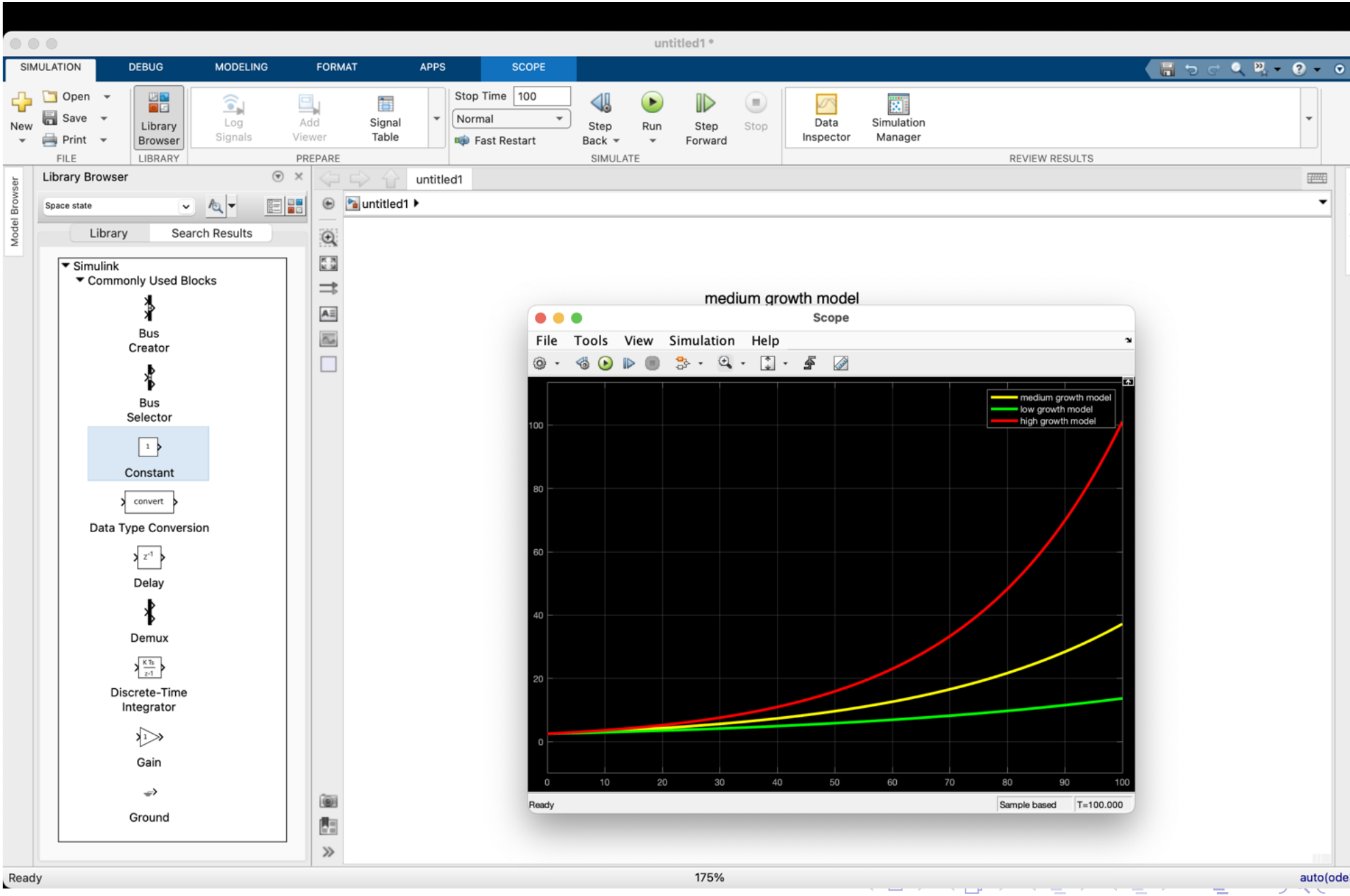
# Plots can be easily adjusted



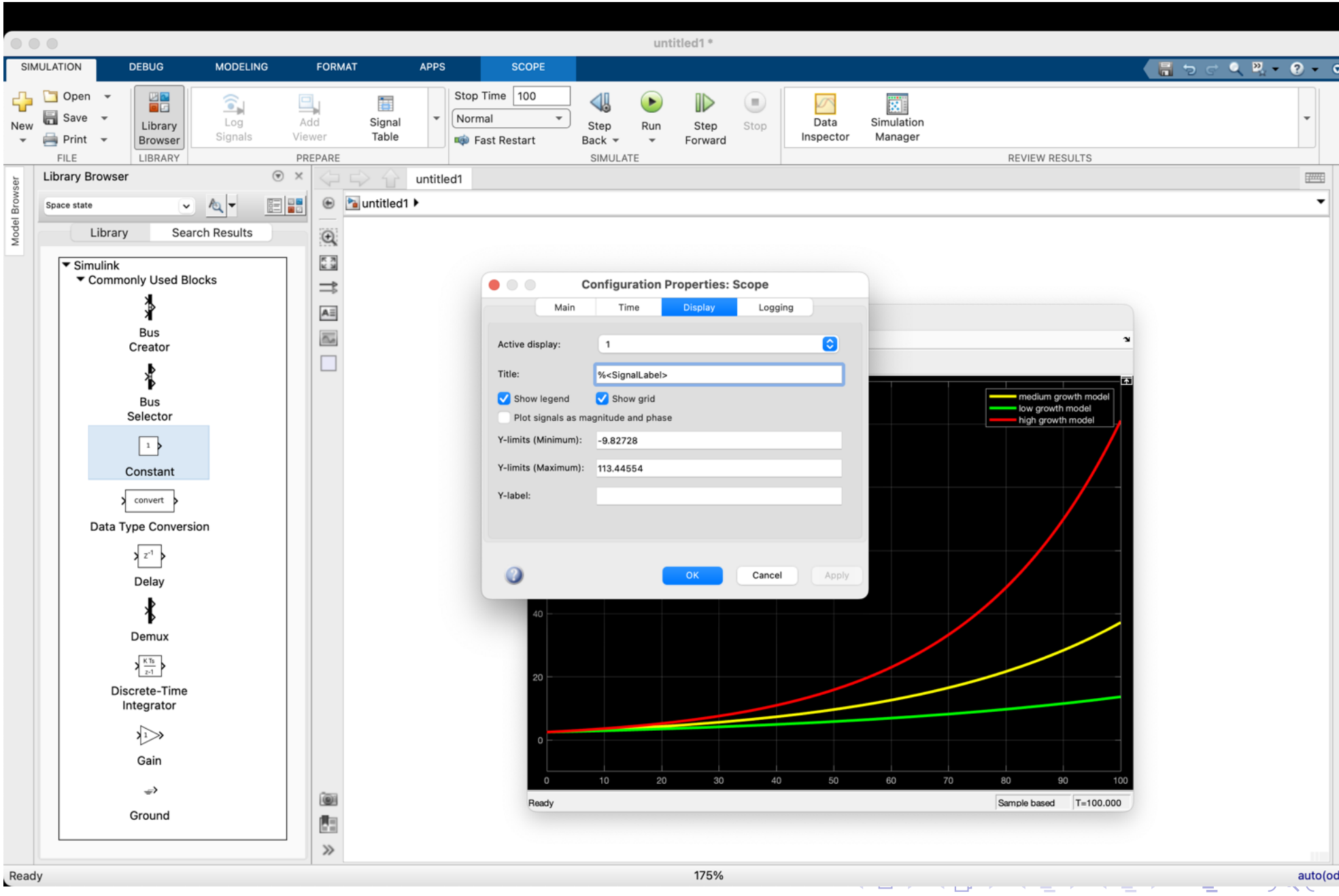
# Plots can be easily adjusted



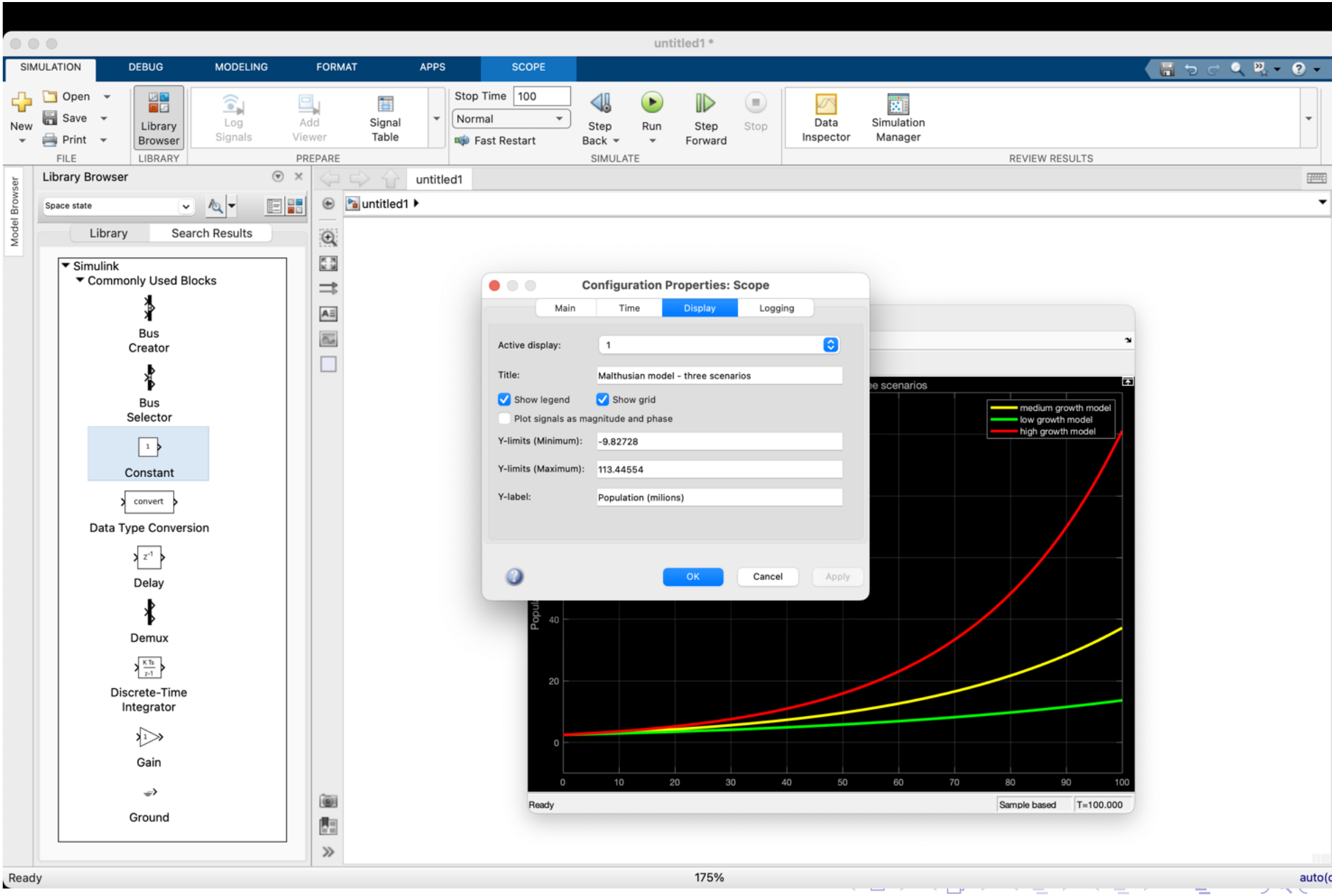
# Plots can be easily adjusted



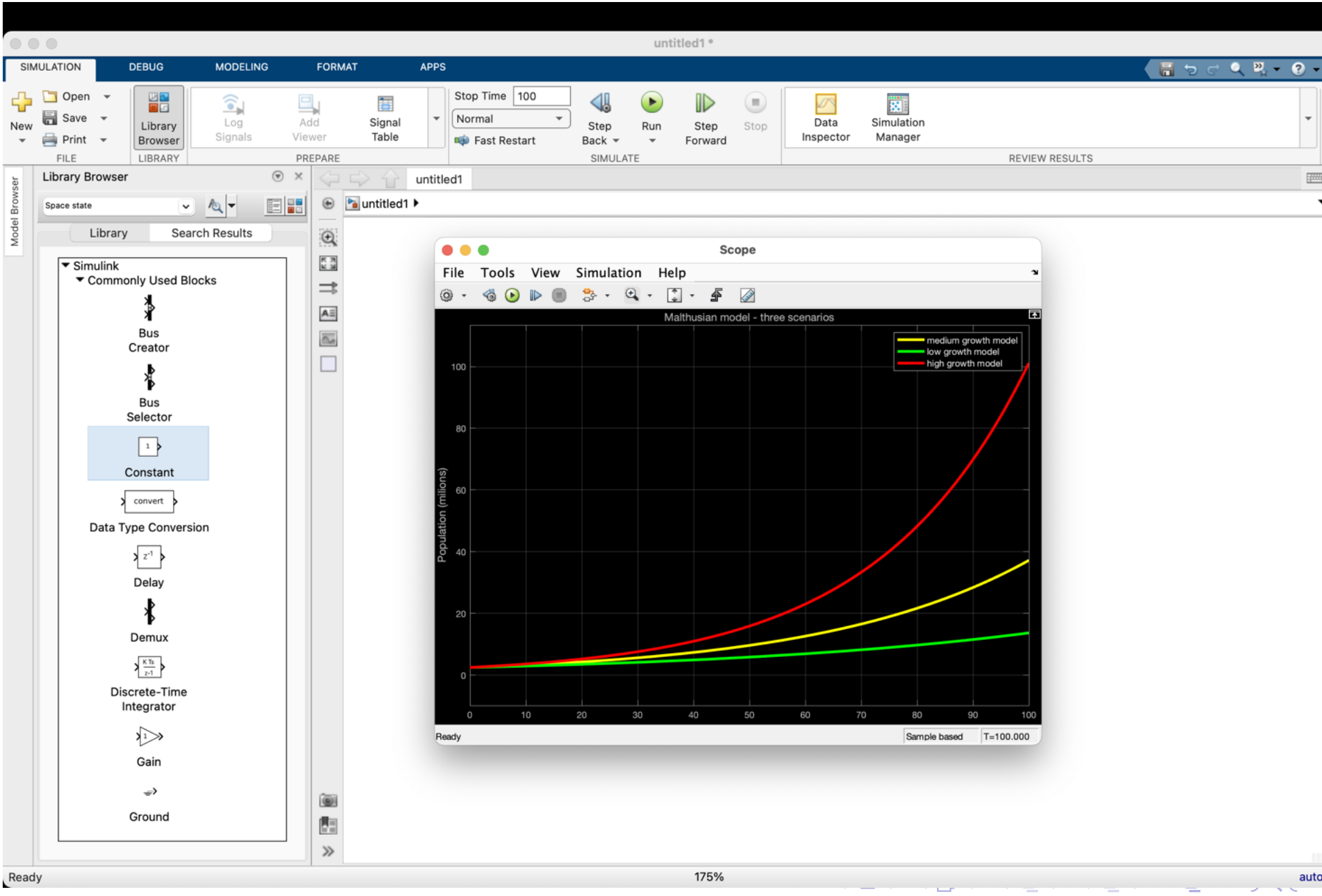
# Plots can be easily adjusted



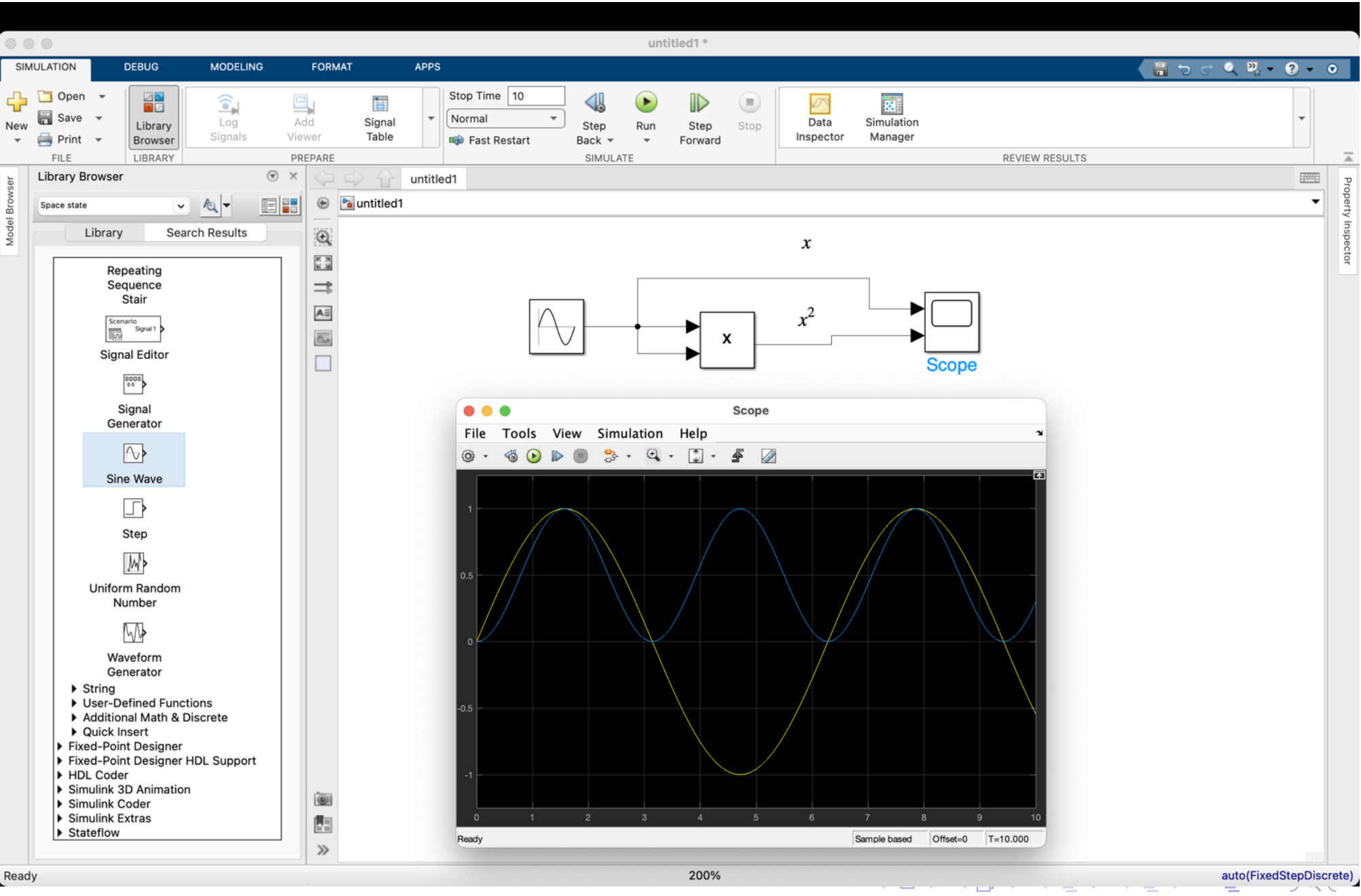
# Plots can be easily adjusted



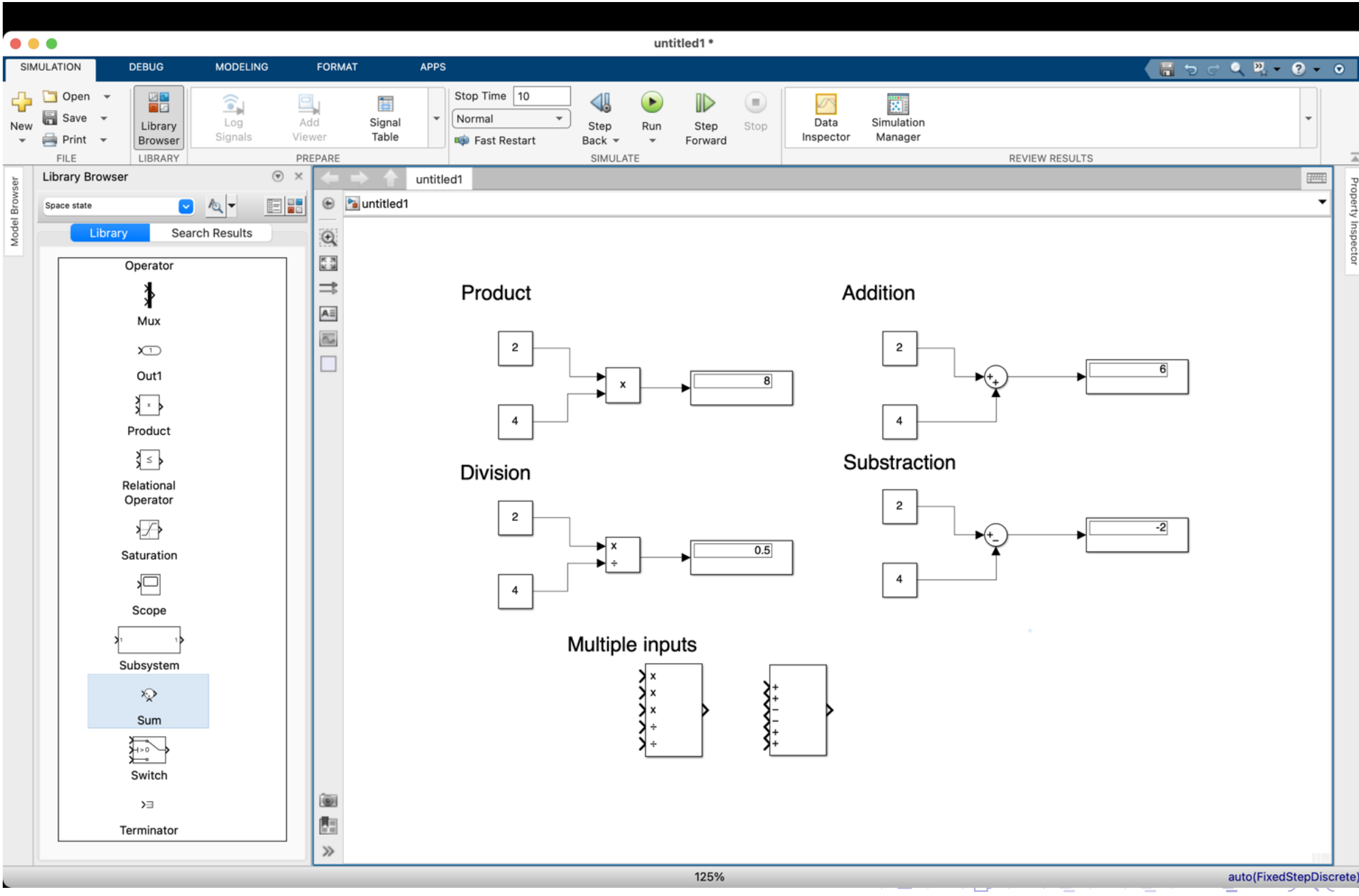
# Plots can be easily adjusted



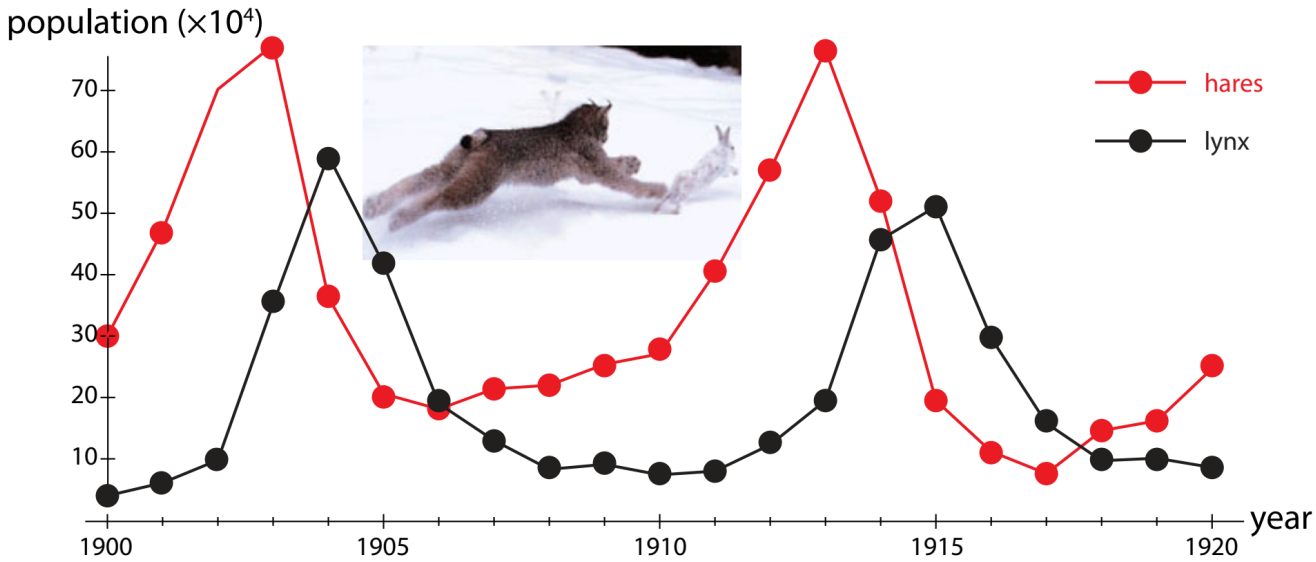
# Experiments with blocks and settings



# Experiments with blocks and settings



# Predator-Prey


population (×10⁴)
70
60
50
40
30
20
10
hares
lynx
year
1900
1905
1910
1915
1920

# 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:

$$\frac{dN}{dt} = rN - \beta NP \tag{1}$$

$$\frac{dP}{dt} = m\beta NP - dP \tag{2}$$

# The Mathematical Model

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

$$\frac{dN}{dt} = rN - \beta NP \tag{1}$$

$$\frac{dP}{dt} = m\beta NP - dP \tag{2}$$

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:

$$\frac{dN}{dt} = rN - \beta NP \tag{1}$$

$$\frac{dP}{dt} = m\beta NP - dP \tag{2}$$

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:

$$\frac{dN}{dt} = rN - \beta NP \tag{1}$$

$$\frac{dP}{dt} = m\beta NP - dP \tag{2}$$

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:

$$\frac{dN}{dt} = rN - \beta NP \tag{1}$$

$$\frac{dP}{dt} = m\beta NP - dP \tag{2}$$

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:

$$\frac{dN}{dt} = rN - \beta NP \tag{1}$$

$$\frac{dP}{dt} = m\beta NP - dP \tag{2}$$

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
- $\beta$ attack efficiency of the predator

# Prey equations

The prey population is given by:

$$\frac{dN}{dt} = rN - \beta NP \tag{3}$$

# Prey equations

The prey population is given by:

$$\frac{dN}{dt} = rN - \beta NP \tag{3}$$

In the absence of predators, the equation becomes:

$$\frac{dN}{dt} = rN \tag{4}$$

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

# Prey equations

The prey population is given by:

$$\frac{dN}{dt} = rN - \beta NP \tag{3}$$

In the absence of predators, the equation becomes:

$$\frac{dN}{dt} = rN \tag{4}$$

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

1em The reduction in prey population due to predators is captured by the expression:

$$-\beta NP \tag{5}$$

# Prey equations

The prey population is given by:

$$\frac{dN}{dt} = rN - \beta NP \tag{3}$$

In the absence of predators, the equation becomes:

$$\frac{dN}{dt} = rN \tag{4}$$

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

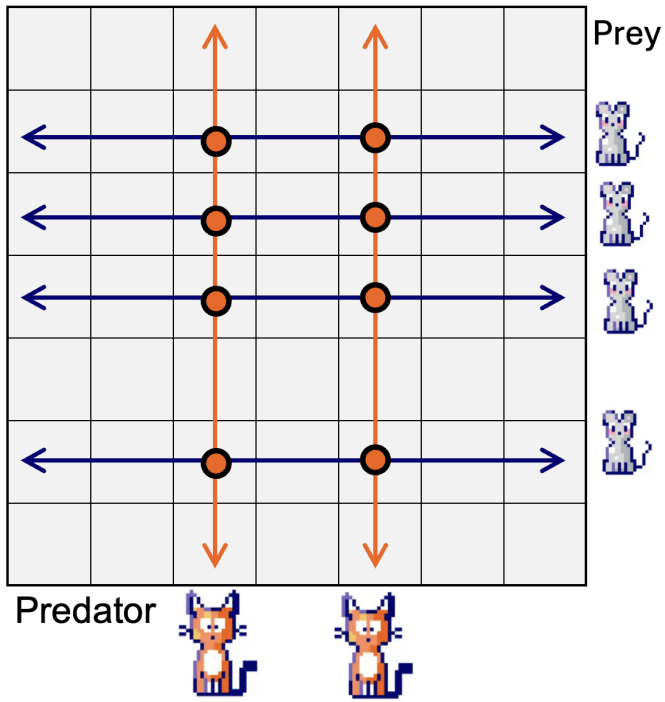
1em The reduction in prey population due to predators is captured by the expression:

$$-\beta NP \tag{5}$$

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. $\beta$ is attack efficiency of the predator. Higher $\beta$ means higher probability of prey being killed.

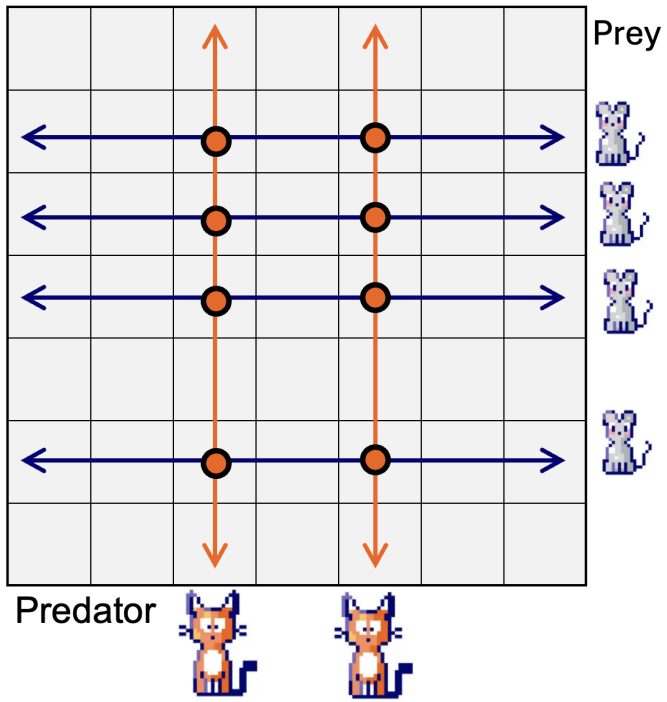
# Predator-prey encounters

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:

$$\frac{dP}{dt} = m\beta NP - dP \tag{6}$$

# Predator equations

The predator population is given by:

$$\frac{dP}{dt} = m\beta NP - dP \tag{6}$$

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

$$\frac{dN}{dt} = -dN \tag{7}$$

which represents dying of prey at exponential rate $d$.

# Predator equations

The predator population is given by:

$$\frac{dP}{dt} = m\beta NP - dP \tag{6}$$

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

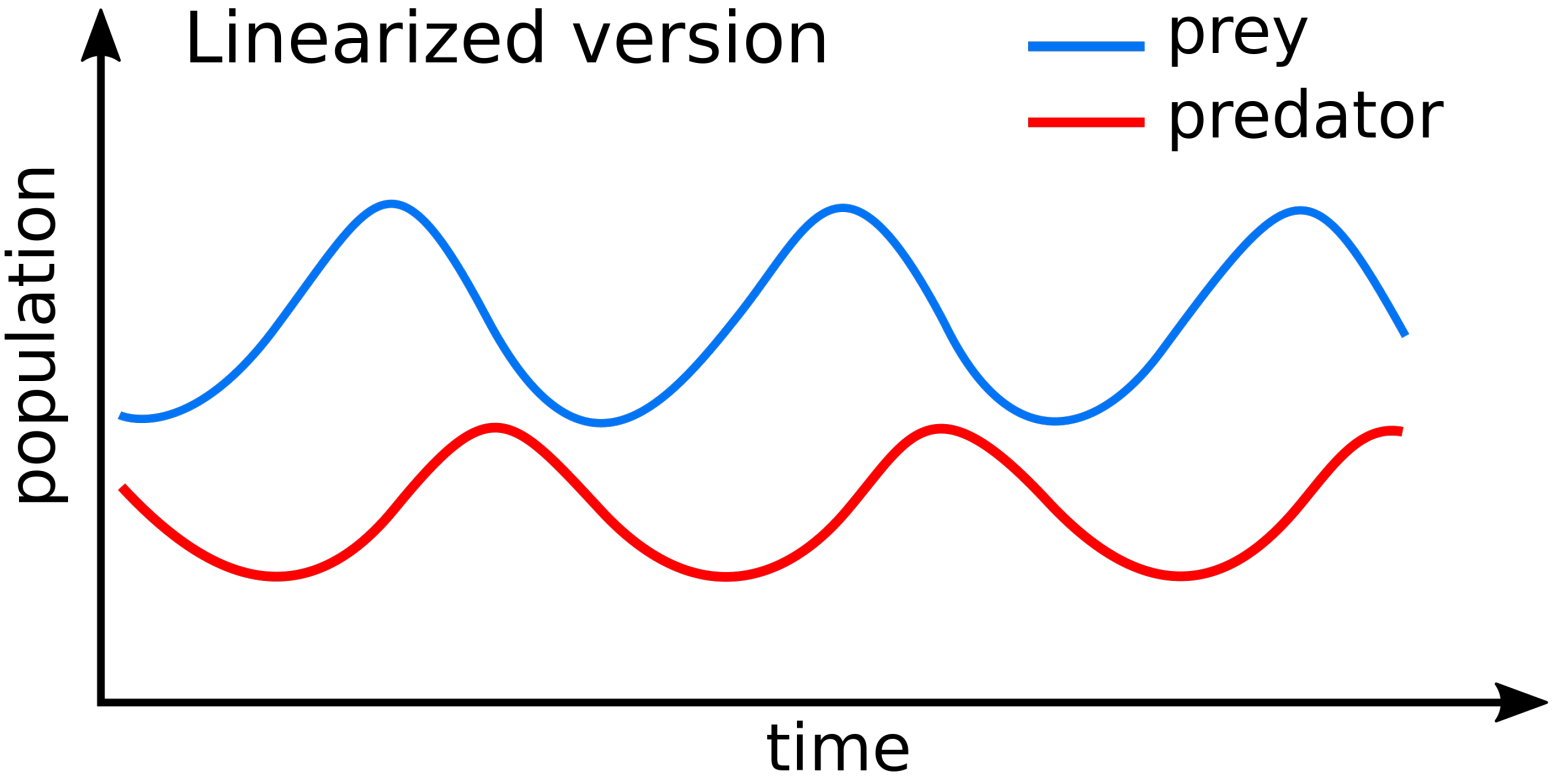
$$\frac{dN}{dt} = -dN \tag{7}$$

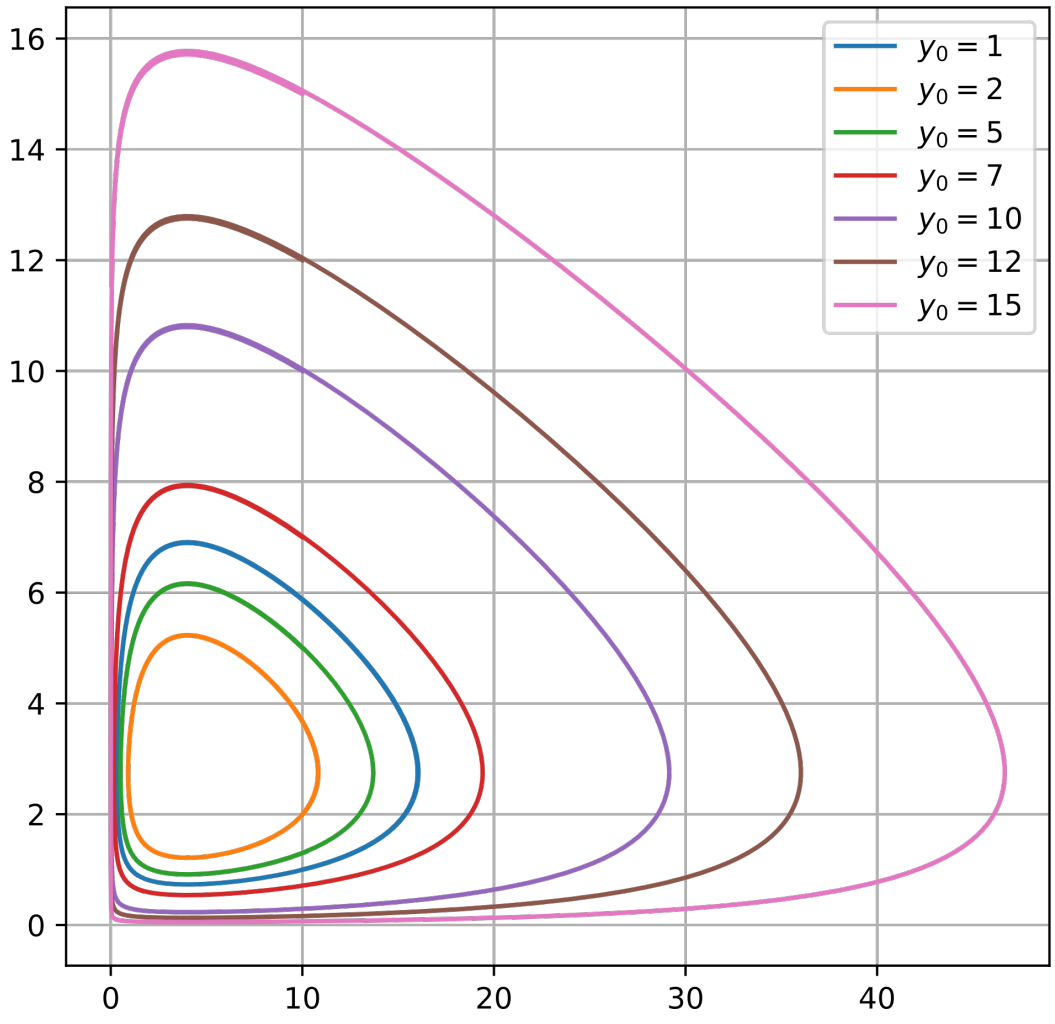
which represents dying of prey at exponential rate $d$.

The growth of predator population growth by eating of prey

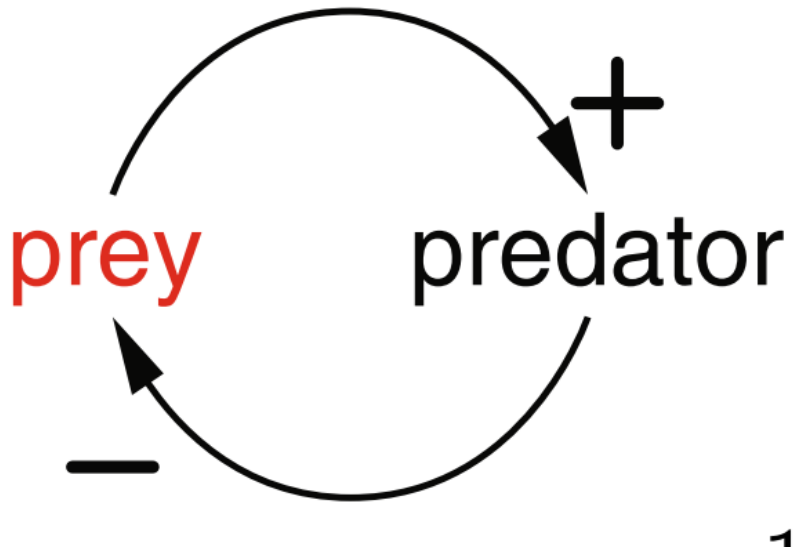
$$+m\beta NP \tag{8}$$

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


Linearized version
prey
predator
population
time


16
14
12
10
8
6
4
2
0
0
10
20
30
40
$y_0 = 1$
$y_0 = 2$
$y_0 = 5$
$y_0 = 7$
$y_0 = 10$
$y_0 = 12$
$y_0 = 15$

# 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 $\beta$ 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.