

# MODELING ORDINARY DIFFERENTIAL EQUATIONS IN MATLAB SIMULINK ®

Ravi Kiran Maddali  
Department of Mathematics, University of Petroleum and Energy Studies, Bidholi,  
Dehradun, Uttarakhand, India  
ravikiran@ddn.upes.ac.in

## Abstract

Ordinary differential equations (ODEs) play a vital role in engineering problems. They are used to model continuous dynamical systems as initial and boundary value problems. There are several analytical and numerical methods to solve ODEs. Various numerical methods such as Euler's method, Runge-Kutta method, etc are so popular in solving these ODEs. MATLAB, the language of technical computation developed by *mathworks*, is gaining importance both in academic and industry as powerful modeling software. SIMULINK®, is a tool in MATLAB for *simulating* both continuous and discrete dynamical systems. In SIMULINK®, we can simulate the behavior of a system by representing the system in terms of a block diagram with *interconnections* between the blocks and there by simulate its *behavior* over certain period of time. The study of ODEs has variety of applications in disciplines like aerospace, electronics, communication, medicine, finance, economics, and physiology. In this article, the technique of modeling and simulation of first order differential equations in SIMULINK, which can be further extended to higher order systems, is discussed.

**Keywords:** Dynamical Systems, Modeling and Simulation, MATLAB, Simulink, Ordinary Differential Equations.

## 1. Introduction

Simulink is a graphical extension to MATLAB for modeling and simulation of systems. In Simulink systems can be represented as block diagrams. A block will perform certain predefined operations on its inputs and produces an output signal which can be sent to another block by means of inter connections. Many blocks such as integrators, transfer functions, summing junctions etc as well as many virtual input and output devices like function generators and oscilloscopes are available in Simulink. The strong integration between MATLAB and Simulink facilitates one to transfer the data between these programs.

### 1.1 Starting Simulink

Simulink can be started from MATLAB command prompt by entering the following command.

```
>> simulink
```

We can also start the Simulink from the tool box of the MATLAB window as shown in the following figure 1.

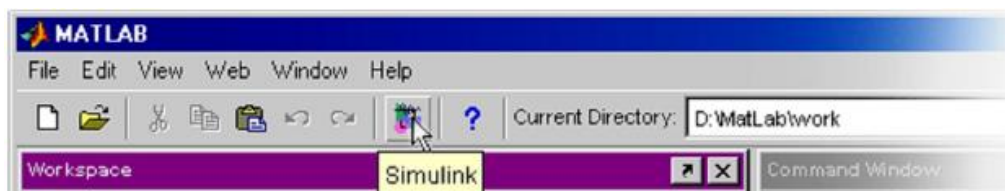


Figure 1 : Starting Simulink from MATLAB window

### 1.2 The Simulink Library browser

Once the Simulink is started, the Simulink library browser will be opened which is shown in the figure 2.

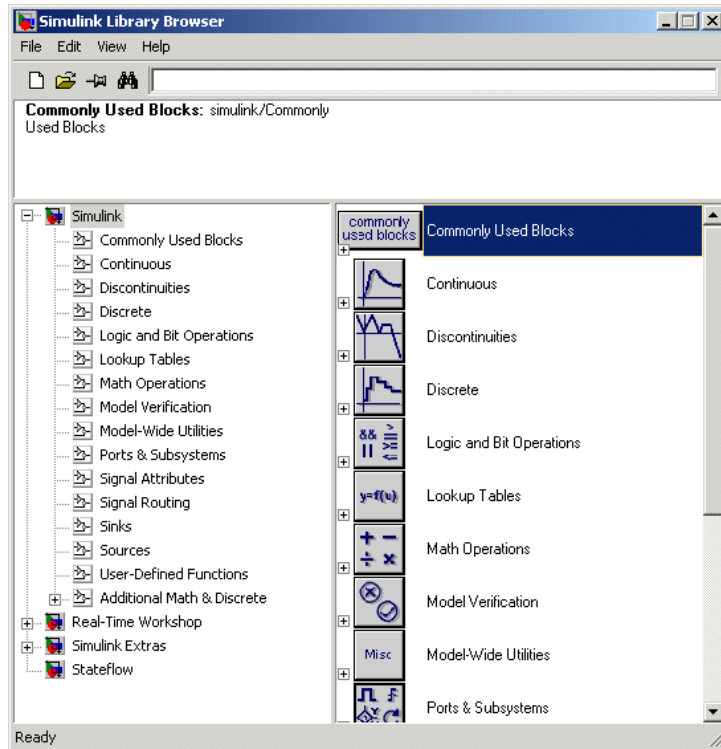


Figure 2: Simulink Block Library

As shown in the Figure 2, the block library consists of different blocks like continuous, discrete, math operations, signal routing, sources and sinks (for input and output purposes of simulations). You can also create your own functions by using user-defined functions block.

Each block contains several subblocks belonging to that category.

## 2.0 Modeling a first order differential equation

Let us understand how to simulate an ordinary differential equation (continuous time system) in Simulink through the following example from chemical engineering:

“A mass balance for a chemical in a completely mixed reactor can be mathematically modeled as the differential equation  $V \frac{dc}{dt} = F - Qc - kVc^2$ , where  $V$  is the volume ( $10 \text{ m}^3$ ),  $c$  is the concentration ( $\text{g}/\text{m}^3$ ),  $F$  is the feed rate ( $200 \text{ g}/\text{min}$ ),  $Q$  is the flow rate ( $1 \text{ m}^3/\text{min}$ ), and  $k$  is a second order reaction rate ( $0.1 \text{ m}^3/\text{g}/\text{min}$ ). If  $c(0)$  is 0, simulate the behavior of the system for  $t \in [0 \ 5]$ ”.

### 2.1 Model Building in Simulink:

The above equation is a first order differential equation. After substituting the known values of  $V$ ,  $c$ ,  $F$ ,  $Q$ , and  $k$  in the above equation, the equation can be represented as following initial value problem.

$$\frac{dc}{dt} = 20 - (0.1)(c + c^2), \quad c(0) = 0 \quad (1)$$

We can solve the above initial value problem numerically by the methods like Euler’s method, Runge-Kutta method etc. The MATLAB Simulink will do the same for solving this equation. But for this, we have to model the equation (1) in Simulink by using blocks available under different categories of Simulink block library. The different blocks used and the corresponding block categories are listed in the following Table 1:

Block Name	Block Category
Integrator	Continuous
Math function	Math operations
Add block (two in number)	Math Operations
Gain block	Math Operations
Constant	Sources
Scope	Sinks
Simout to workspace	Sinks

Table 1: Blocks used in Simulink model

The purpose of the above mentioned blocks can be describes as follows:

- The *integrator* block integrates the input and is used with continuous time signals.
- The *math function* block can be used to generate the corresponding function value of its input.
- The *add* block is used to add its two inputs.
- The *constant* block is from the *sources* category used to store any constant value and pass it as input to any other block.
- The *scope* block is used for graphical display of solution as a function of simulation time.
- The *simout to workspace* block is used to write its inputs to the MATLAB workspace.

To create a new Simulink model press FILE → NEW → MODEL from the Simulink Library browser. A new model window will be opened in which we will have to transfer the required blocks and make proper connections among them to represent the equation of our interest. Let us save this file by name *timeconc*. The Simulink extension for model files is **.mdl**.

To transfer the blocks (for example *integrator block*) from the Simulink library, click on the *continuous* category and then position the mouse on the *integrator* and drag it into the model window and release. Similarly drag all the other blocks into the model window. After this step, your model window will be as shown in the figure 3:

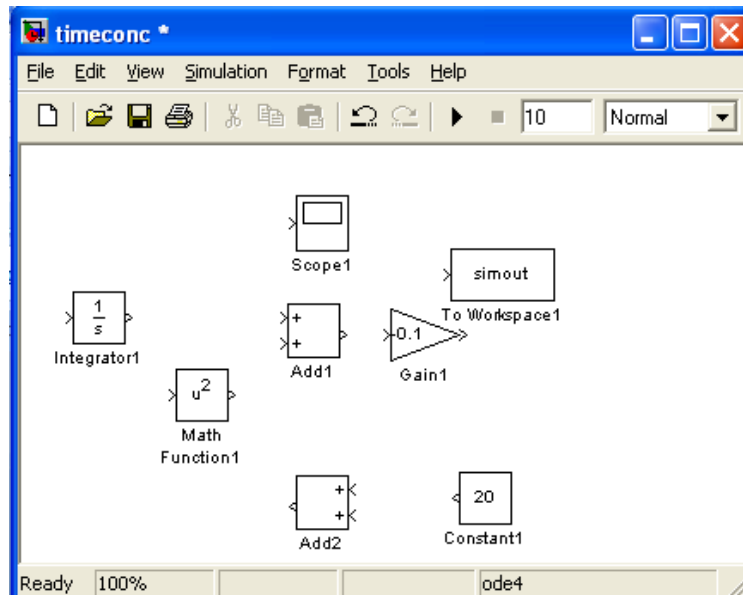


Figure 3: Blocks without connections in model window.

From the Figure 3, we can observe that every block is having ports on either of its sides (except for blocks from sources and sinks which have only input ports). The port(s) pointing towards the block are input port(s) and the port directing away from the block is its output port. We can establish connections between two blocks by

pressing the output port of a block with left mouse button and releasing it after dragging it to the input port of the other block. So we will adopt this method to make necessary connections among blocks.

From the equation (1), simple integration on both sides and the substitution of the initial condition, yields us the solution  $c(t)$ . So, the input to our *integrator* block will be the right hand side of the equation (1). Therefore first make the connections to represent the right hand side of the equation and then input it to the *integrator* block. This can be achieved by the following steps:

- Double click on the constant block. A *source block parameters* window will be opened. Set the *constant value* to 20.
- As the term  $(c+c^2)$  is to be multiplied with the value  $-0.1$ , double click on the *Gain* block and set the *Gain value* as  $-0.1$  in the *Function block parameters window*.
- Since the output of the integrator block is  $c$ , and as we need to perform the operation of addition  $(c+c^2)$ , join the output port integrator to the first of the input ports of *Add* block (Add1). Then the *Add* block receives  $c$  as an input. Then from the branch joining integrator and add block (Add1), press control and left mouse button and drag it to the input port of math function block.
- Double click math function block and set the function as *square* from the available popup values from the function block parameters window. Also join its output port to the second input port of Add1. By this time, the add block (Add1) has two inputs  $c$  and  $c^2$  to perform the operation of addition.
- Pass the output of the add block to the input of the *Gain* block so that the operation  $-0.1(c + c^2)$  is performed.
- Join the output of *Gain* block to the first input of the second add block (Add2) and also join the constant block to the second input of the second add block (Add2) (which is flipped by right clicking on the block and setting the format).
- At this stage, the right hand side of the equation is modeled and so complete the process of modeling the equation by connecting the output of the add block to the input of the integrator block. Also double click the integrator block and set the initial condition to '0' since  $c(0)=0$ .
- Join the branch joining integrator and the first add block to the scope block for getting the graph of the solution.
- Also connect the output branch from integrator to scope as input to the *simout to workspace* block for representing the output as MATLAB variables which will be stored in MATLAB workspace for further analysis.

After making the necessary connections, the Simulink model for the equation (1) will be as in the Figure 4.

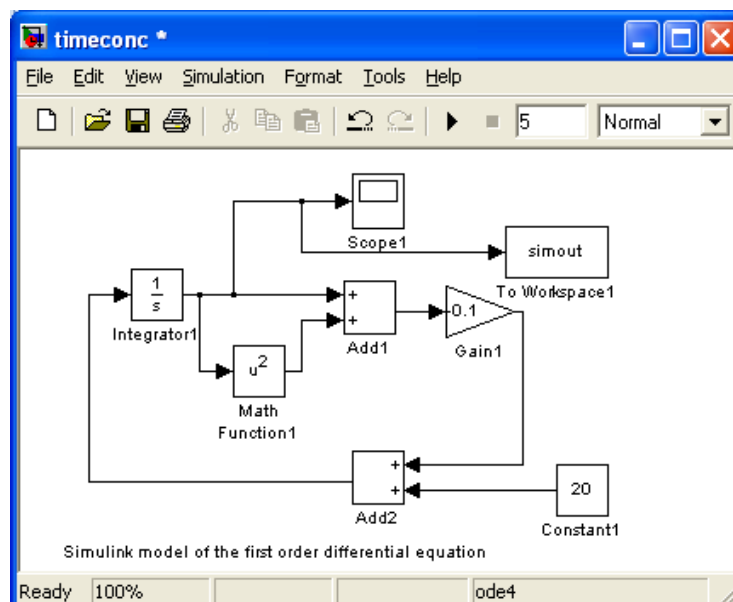


Figure 4: Simulink model of the equation (1)

## 2.2 Running the simulation:

As we need to simulate the system from  $t=0$  to  $t=5$ , set the stop time of the simulation to 5 from the model window. Set the necessary solver options by clicking on the SIMULATION → CONFIGURATION

PARAMETERS. Let us simulate the system by setting the solver as ODE4 (Runge-Kutta), using a fixed step size as  $0.1$ . We can also change the start and stop time from the same window which is shown in the Figure 5.

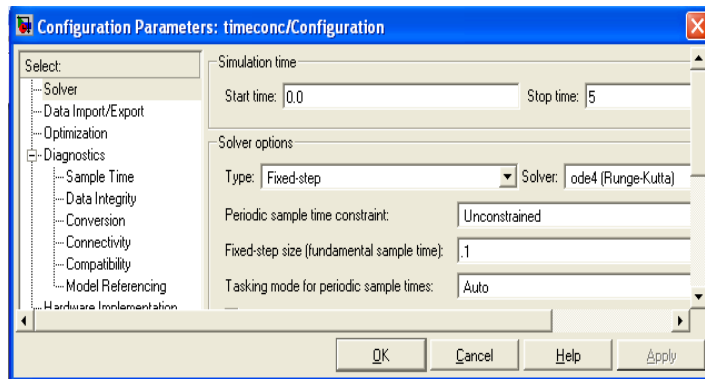



Figure 5: Configuration Parameters Dialogue Box.

Run the Simulation by clicking the Start Simulation button  from the model window toolbox. After running the simulation, the graphical output of the solution (concentration) can be obtained by clicking on the scope block which is showed in the Figure 6.

Also two variables *tout* and *simout* will be created in MATLAB workspace which can be used for further analysis. The variable *tout* will contains values from 0 to 5 with an increment of step size  $0.1$ . The variable *simout* contains the concentration  $c(t)$  values corresponding values of *tout*.

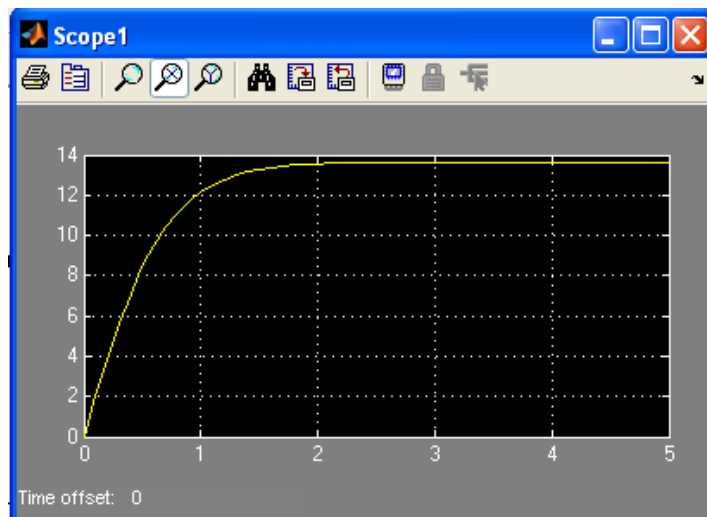


Figure 6: Simulation Results.

### 3.0 Conclusion

The task of modeling differential equations is very interesting and easy. The Simulink uses numerical solvers to solve ODEs and there are several numerical solvers available. A user can set the solver options from the 'configuration parameters' from the *simulation* menu, depending upon the requirement. Different block parameters can be changed within no time and the behavior of the system can be analyzed i.e. studying the behavior of the system by changing the block parameters, becomes more easy. Since the labor of finding solution is reduced by using software, the user can concentrate on the conceptualization rather than on mere calculations. The same modeling technique can be extended for higher order systems without any efforts.

### 4.0 References

- [1] Chapra, Steven C., Applied Numerical Methods with MATLAB for Engineers and Scientists (2012), McGraw Hill Publications.
- [2] Karris, Steven T., Introduction to Simulink with Engineering Applications (2006), Orchard Publications.
- [3] Klee Harold, and Allen Randal, Simulation of Dynamical Systems with MATLAB and Simulink(2011), second edition, CRC Press.
- [4] Palm William J III, Introduction to MATLAB for Engineers (2010), 3<sup>rd</sup> edition, McGraw Hill Higher Education.
- [5] Mathworks homepage for MATLAB ([www.mathworks.in](http://www.mathworks.in)).