**Introduction to MATLAB**

* MATLAB (an abbreviation of "MATrixLABoratory") is a proprietary multi-paradigm programming language and numeric computing environment developed by MathWorks.
* MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages.
* MathWorks is an American privately held corporation that specializes in mathematical computing software. Its major products include MATLAB and Simulink, which support data analysis and simulation.
* Simulink is a MATLAB-based graphical programming environment for modeling, simulating and analyzing multidomain dynamical systems.
* Its primary interface is a graphical block diagramming tool and a customizable set of block libraries.
* It offers tight integration with the rest of the MATLAB environment and can either drive MATLAB or be scripted from it. Simulink is widely used in automatic control and digital signal processing for multidomain simulation and model-based design.

**Solver Setting**

* To simulate a dynamic system, you compute its states at successive time steps over a specified time span.
* Time steps are time intervals when the computation happens.
* The size of this time interval is called step size.
* The process of computing the states of a model in this manner is known as solving the model.
* No single method of solving a model applies to all systems.
* Simulink® provides a set of programs called solvers. Each solver embodies a particular approach to solving a model.
* A solver applies a numerical method to solve the set of ordinary differential equations that represent the model.
* Through this computation, it determines the time of the next simulation step. In the process of solving this initial value problem, the solver also satisfies the accuracy requirements that you specify.

**NOTE - Problem** while solving electrical stimulation. AC signal is showing triangular wave – check solver and other settings.

**Questions –**

1. What is solver – In matlabthe set of programs which computes dynamic system over a specified step size.
2. What is step size – In matlab time step are time interval when computation happens and this size of this time intervals is called step size.
3. What is Matlab – It’s a high-performance language for technical computing. It integrates computation, visualization and programming in easy-to-use environment where the problem and solution are expressed in simple mathematical form (Modelling, simulation, prototyping, algorithm, programming)
4. Why we use fixed step solver for Code gen? – Variable solver – steps vary according to model (increases or decreases step size to meet the tolerance) so this step sizes cant be mapped with real time clock of target system hence we use fixed step size. (Software hardware interfacing)
5. What is your sample time why its 1 second?– Time taken by a block to update its internal state.Simulink determines the best sample time for the block based on the block context within the model.
6. What is simulation time – It is time defined by user for simulation of model in seconds.
7. What is start time and stop time – As per our requirement.
8. What are types of sample time? – Continuous, Discrete, triggered, variable, constant (S function)??

**Basic Blocks**

**Subsystems**

The various number of blocks are converted into a single block then this single block is called a Subsystem model.

Helps reduce the number of blocks displayed in your model window.

Reduce complexity and improve readability of model.

Main Types –

1. Virtual (Non atomic) - Execution is block by block –Code is generated only for individual block not for whole subsystem – Subsystem navigation for code is prohibited.
2. Non-Virtual (Atomic) - Execution as a single block - Code is generated for whole subsystem – Subsystem is navigated for code.

Sub Types –

1. Enabled (atomic)
2. Triggered (atomic)
3. Variant (non atomic)
4. Enabled Triggered
5. ***Switch Case Action Subsystem***
6. Enabled –

The control signal can be either a scalar or a vector.

If a scalar value is greater than zero, the subsystem executes.

If any one of the vector element values is greater than zero, the subsystem executes

(-ve 0 +ve +ve)

**+ve to +ve**

**-ve to +ve**

**zero to +ve**

**(NO -ve to zero)**

1. Triggered –

A triggered subsystem is a conditionally executed atomic subsystem that runs each time the control signal (trigger signal):

Either rises from a negative value to a positive value or zero, or rises from a zero value to a positive value.

Either falls from a positive value to a negative value or zero, or falls from a zero value to a negative value.

Rises or falls through or to a zero value.

**Set how the control signal triggers execution.**

**rising —** Trigger execution of the subsystem when the control signal rises from a negative or zero value to a positive value. (-ve 0 +ve )

**falling —** Trigger execution of the subsystem when the control signal falls from a positive or zero value to a negative value.(+ve 0 -ve )

**either —** Trigger execution of the subsystem with either a rising or falling control signal.

1. Variant –

The Variant Subsystem block enables you to include multiple implementations of a component in a separate hierarchy.

Will take common input and output for all the subsystems created in variant subsystem

Ex. Two Models – add and sub

At a time only one is active

Initialize by right clicking on variant sub system by opening variant manager (expression add condition a==1 for addition and expression sub condition a==2 for sub)

Select the mode – command (add=Simulink.Variant ('a==1')) or expression or label

For expression call the subsystem from command window (a=1 or a=2)

For label select from right clicking on variant subsystems - RUN

1. Enabled Triggered

When both are on, we get the output.

**Masking-**

A mask is a custom user interface for a block. By masking a block you encapsulate the block diagram to have its own parameter dialog box with its own block description, parameter prompts, and help texts.

You can mask an independent custom block that you can reuse as unique blocks like those defined in Simulink.

Works like a Simulink.parameter

**Initialization –** If you initialize any variable in this pane (a=10;) then even if you prompt a from mask it will always take a as 10 and not the value entered in the prompt

**Parameters and dialogue box–** In this pane we can add the values or variables that we want to prompt when a mask is opened. Select and configure by going to type options.

**Merge Block –**

**When you use the Merge block, follow these guidelines:**

* Always use conditionally executed subsystems to drive Merge blocks.
* Ensure that at most one of the driving conditionally executed subsystems executes at any time step.
* Ensure that all input signals have the same sample time.
* Do not branch a signal that inputs to a Merge block, if you use the default setting of Classic for the Model Configuration Parameters > Diagnostics > Underspecified initialization detection parameter.
* For all conditionally executed subsystem Outport blocks that drive Merge blocks, set the Output when disabled parameter to held.
* If the output of a Model block is coming from a MATLAB Function block or a Stateflow® chart, do not connect that output port to the input port of the Merge block.

MUX –

Combines Many signals to One signal

Accepts any data type

Can convert two or more signals in to vectors

**(Must Be Same Data Type)**

DEMUX-

Split vector signals into scalars or smaller vectors. Check 'Bus Selection Mode' to split bus signals.

Bus Creator –

This block creates a bus signal from its inputs.

Any signal type can be connected.

**All data types are accepted.**

Bus selector –

If not selected as virtual bus (Virtual bus tab is disabled)

This block selects and shows all signals from its inputs in different or multiple way (output)

Any signal type can be connected.

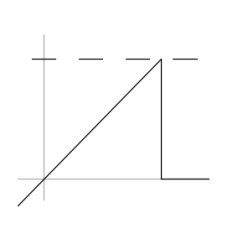
All data types are accepted.

If selected as virtual bus (Virtual bus tab is enabled)

This block selects and shows all signals from its inputs in same or one way (output)

**Data type must be same. Like MUX**

Wrap to Zero –



If the input is above the threshold, the output is zero, otherwise the output equals the input.

Detect Change –

3 Types - Increase, decrease, General

If the input does not equal its previous value, then output TRUE, otherwise output FALSE. The initial condition determines the initial value of the previous input U/z.

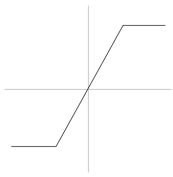
when U & U/Z same values that time out for all the detects is zero

initial conditions of all detects are zero

U - previous value

U/Z - current value

Saturation –



Limit input signal to the upper and lower saturation values.

If input is within lower limit or upper limit then output is equal to input, otherwise output is equal to lower or upper limit depending on input.

Dead zone –

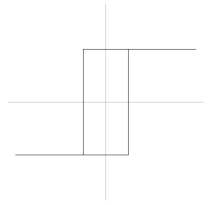


Output zero for inputs within the dead zone. Offset input signals by either the Start or End value when outside of the dead zone.

If input value is withinStart of dead zone and end of dead zone then output is zero, otherwise

1. If input is less than start of dead zone value then output is equal to input minus Start of dead zone value (lower)
2. If input is more than end of dead zone value then output is equal to input minus end of dead zone value (lower)

Relay –



Select and enter switch On and Switch Off points and also enter Output when on and Output when off.

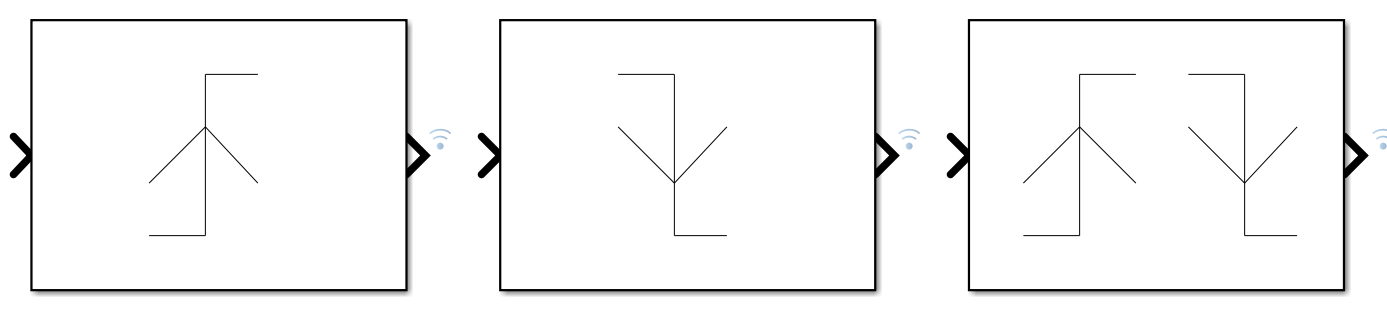
When the input value is more than switch On point then output is equal to **Output when on value.**

When the input value is less than switch off point then output is equal to **Output when off value.**

**The on/off state of the relay is not affected by input between the upper and lower limits.**

Output the specified 'on' or 'off' value by comparing the input to the specified thresholds.

Edge detector –



3 types – Rising, falling and either.

If input is greater than zero then output is - rising

If input is lesser than zero then output is – falling

Either change from zero then output is 1

**Look UP Table –**

It’s an array of data which maps input values to output values thereby approximating mathematical functions.

1D = Y = X2

Y = Table data

X = Breakpoints

This gives the relation and output between the values – Interpolation – Process of estimating values that lie between known values

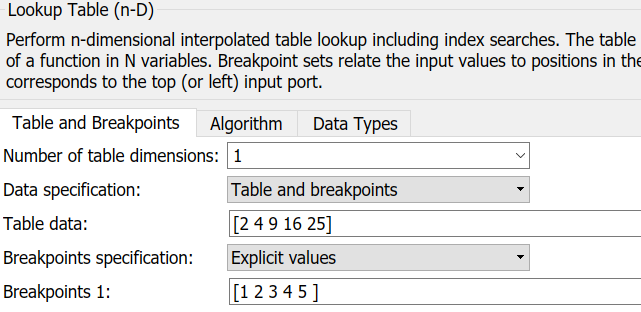
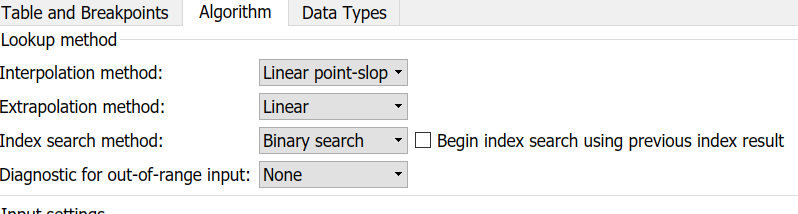
This gives the relation and output above the values – Extrapolation - Process of estimating values that are beyond known values

Types – 1D and 2D

Method of interpolation – Generally – Linear point slope Algorithm.

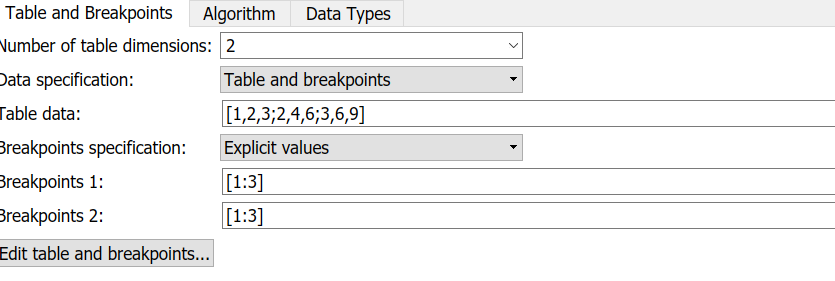
Method of extrapolation – Generally – Linear Algorithm.

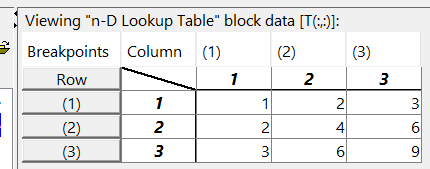
1D Lookup



2D Lookup

In 2d output is a function of two inputs (x and y)

Z= X(row)\*Y(column)



STATE FLOW-

**BASICS**

**State Action Types**

Entry Action: − Entry actions are executed first when a state becomes active.

Transition between states – Generally given as conditions (i==5)

During Action: −It checks the transition action and then executes within state (i=i+2)

Exit Action: −condition when during action is completed (i=10)

EX

en:i=1;

du:i=i+2;

ex:i=10;

Transition [i==5]

**State Decomposition**

Every state has decomposition that decideshowvarious sub states will execute.

○ Exclusive (OR) State Decomposition :−

Substates with solid borders indicate exclusive (OR) state decomposition. Use this decomposition to describe operating modes that are mutually exclusive. When a state has exclusive (OR) decomposition, only one substate can be active at a time.

○ Parallel (AND) State Decomposition :−

Substates with dashed borders indicate parallel (AND) decomposition. Use this decomposition to describesimultaneous operating modes. When a state has parallel (AND) decomposition, all substates are active at the same time.

**Temperal Logic –**

Temporal logic controls the execution of a chart in terms of time OR ticks (clicks).

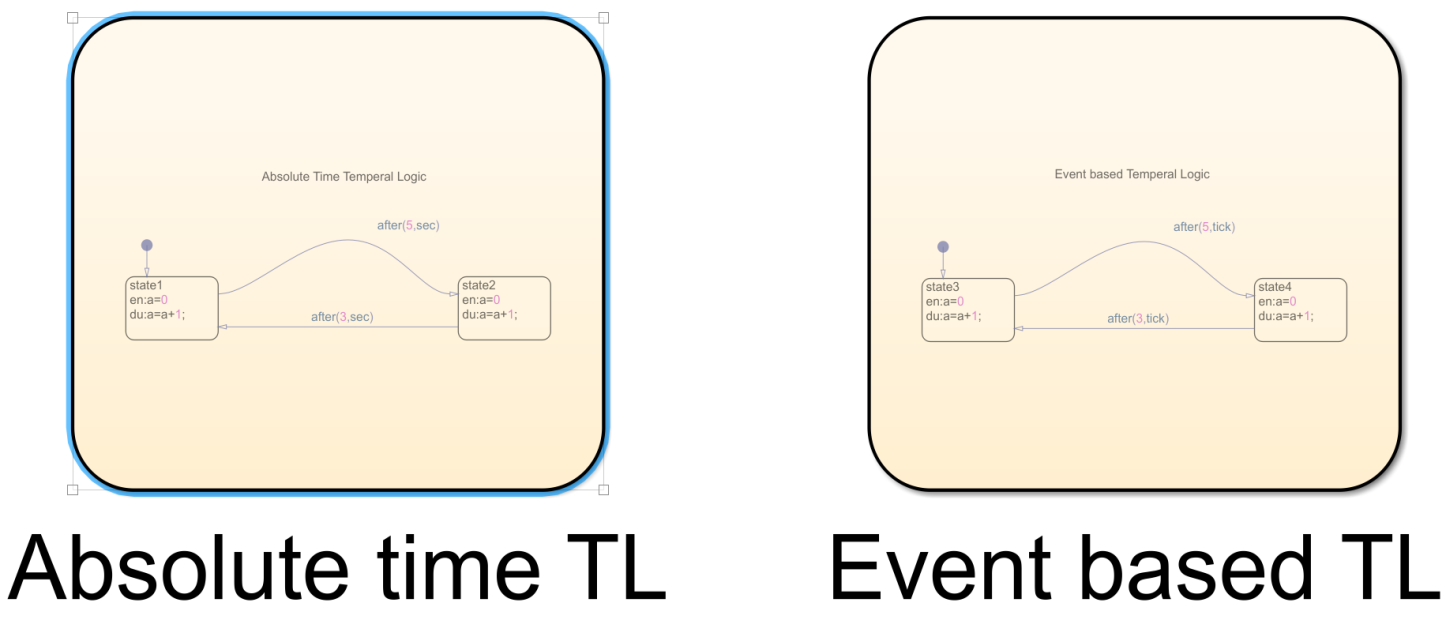
In state actions and transitions, you can use two types of temporal logic:

▪ Absolute time temporal logic

Absolute−time temporal logic tracks the time (sample time from solver) since a state became active.

▪ Event base temporal logic

Event−based temporal logic tracks the ticks (clicks) of events.



**Truth table –**

It represents logical decision-making behavior with condition decision and action.

Truth tables implement combinatorial logic design in a short, tabular format.

1. First give the default transition
2. Specify the same name on truth table and default transition panel (else shows error)

Truth table output can also be an enum – for that you have to specify file name same as that of in script in truth tanbles action column like - output = filename.enum

**History Junction –**

A history junction records the activity of substates inside superstates.

Use a history junction in a chart or superstate to indicate that its last active substate becomes active when the chart or superstate becomes active.

Placing a history junction in a superstate indicate that historical state activity information is used to determine the next state to become active.

**Super State and Sub State -**

A state within the boundaries of another state indicate that inner state is a substate or child of the super state

The outer state is a parent of inner state or we can say it a supper state.

**State flow pattern –**

The pattern wizard is a utility that generate common flow chart patterns for use in graphical function and charts.

This saves lot of time compared to adding transition manually and also reduces the risks of error while designing any flow charts.

State flow pattern generally,

Generates common logic and iterative loop patterns.

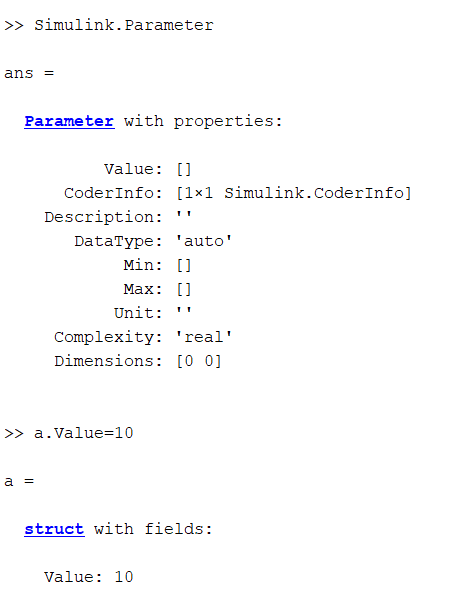
Promotes consistency in geometry and layout across patterns.

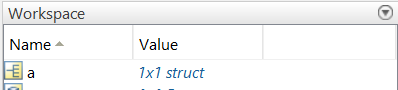
Facilitates storing and reusing patterns from a central location.

Allows inserting patterns in an existing flow chart.

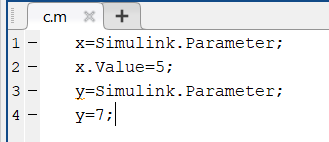
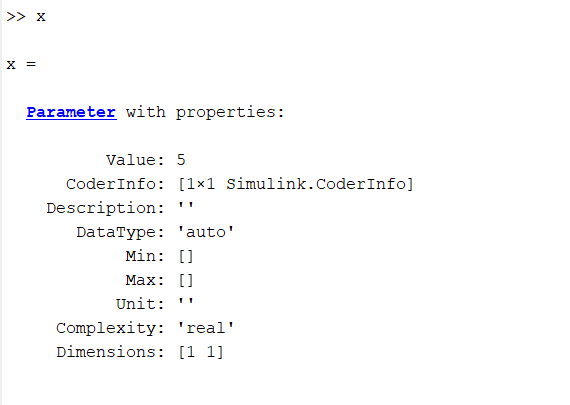
**Simulink.Parameter–**

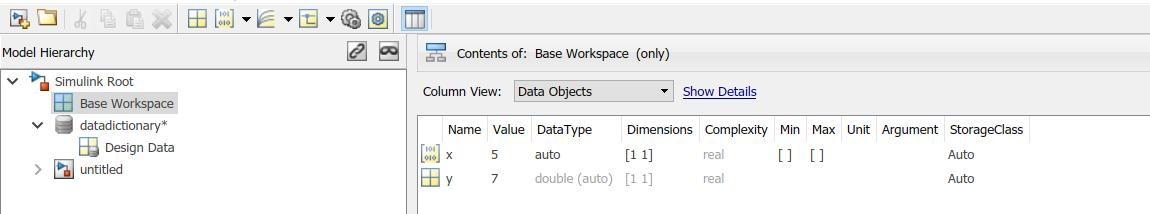
1. **From Command prompt**





1. **From Scripts –**

 **3. From base workspace –**

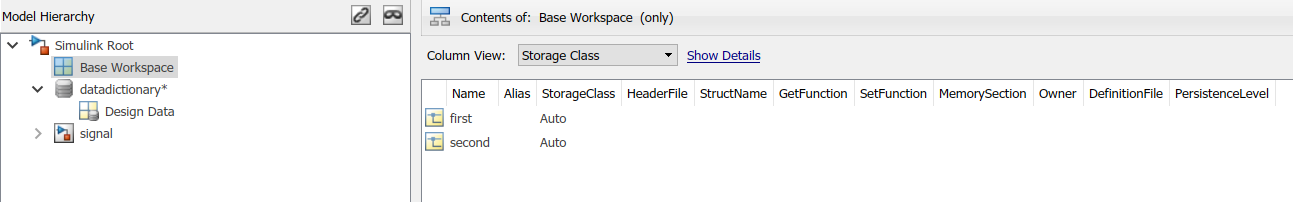


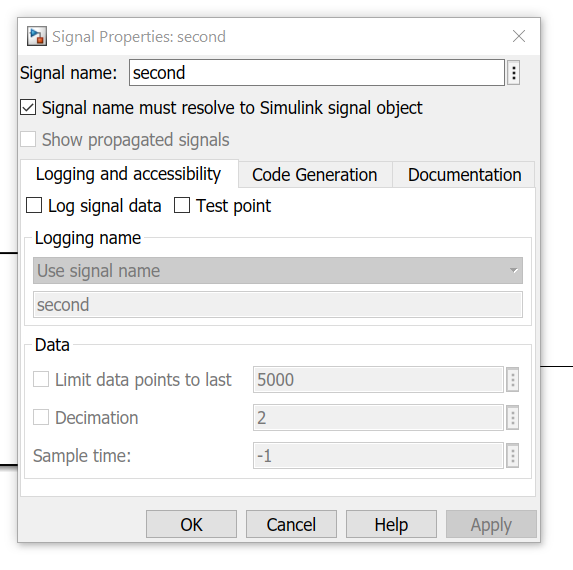
**Simulink.Signal–**

Specify instance-specific properties of signal or discrete state

A Simulink.Signal object enables you to assign or validate the attributes of a signal or discrete state, such as its data type, numeric type, dimensions.

**Successful validation guarantees that the signal has the attributes that you intended it to have. (no loss of data)**





**Data Dictionary –**

A data dictionary is a source of data that are relevant to your model.

You can also use the base workspace to store design data that are used by your model during simulation. However, a data dictionary provides more capabilities.

One can load variables in base workspace but when model is reopened, we need to add variables again this disadvantage is overcome by just loading data dictionary containing all the variables used in the model. This saves lot of time and one can also use this dictionary for various computer systems containing same model. Main use is that one can use a single variable at any point in model in data dictionary.

The dictionary stores design data, which define parameters and signals, and include data that define the behavior of the model.

**The dictionary does not store simulation data, which are inputs or outputs of model simulation that enter and exit Inport and Outport blocks.**

**MIL Testing (Model in loop) –**

Difference between slx and slxs?

Error - Unrecognized property 'deta' for class 'timeseries'

Solution – spelling was wrong – correction made (deta to data) data is right.

Error - Invalid expression. When calling a function or indexing a variable, use parentheses. Otherwise, check formismatched delimiters

Solution – There should not be equal to sign after xlsread and xlswrite command. (directly use bracket after command)

Error - Undefined function or variable 'xlxwrite'.

Solution – Check command neatly (xls not slx or xlx)

Error - dot indexing not supported for this type of variable

Solution – Always start time from **0**

Error – test3.xlsx(file\_name) is not writable. It might be locked by another process.

Solution – Xcel sheet open (must close the xcel sheet before running scripts)

Error – file name format does not comply with signal builder required format

Solution – check in xcel sheet – Only one worksheet must be there inside our xcel sheet

**Error -**Invalid structure-format variable specified as workspace input in 'test1/From Workspace'. If the input signal is a bus signal, the variable must be a structure of MATLAB timeseries objects. Otherwise, the variable must include 'time' and 'signals' fields, and the 'signals' field must be a structure with a 'values' field.

**Solution –**

1. While doing MIL for a model, If you are using from workspace block you must use timeseries command in script.

**Names in xcel sheet and script**