

# Matlab Automated Test Tool (MATT) Documentation

Joel Henry, [Real-time System Lab of the University of Montana](#)

## What Is MATT

MATT is a testing tool for Matlab/Simulink models that can be used by both developers and testers to both simulate models and test automatically generated code. MATT makes generation of test data fast and easy, configuration and execution of simulations automatic, and detection of defects simple. Two unique features of MATT are: 1) it is integrated into the Matlab environment and 2) it reads and parses systems into subsystems making it easy for the user to navigate and test Simulink models.

## Specific Features

MATT provides an easy to learn and user-friendly interface and environment for testing both subsystem and system-level requirements using a blackbox testing approach. MATT works with Matlab but does not alter the Simulink models in any way. The intent of MATT is to provide developers and testers with tools they can easily use by leveraging their knowledge of the models and requirements to quickly create and execute tests, and then have defects automatically detected. MATT does not advance any specific testing approach or force the user to adhere to a theoretical model of test generation or execution. MATT developers believe that testing is most productive when a general tool is in the hands of users with specific project requirements and Simulink model knowledge.

MATT gathers information about a Simulink model through Matlab and then presents the user with an interface. The interface allows the user to choose the entire system or any subsystem within the model. Once the system or subsystem is chosen the interface populates a series of tab pages. The first tab page contains all inports to the system or subsystem, including their names, types, and range limitations (if specified within the model). The user can then specify the input values for each inport from a list of built-in functions, a Matlab function, an external file, or by using the graphical input specification tool (GIST). From this tab page, the user specifies the time step duration and simulation length.

Next, the user tabs to the output tab to specify the defect detection criteria. This can be done individually for each outport (for example, outport 3 values greater than 6400 are defects), or the user can configure defect detection based on a combination of outport values (for example, outport 3 less than 3200 and output 6 greater than 10.0). Defect detection criteria can also be specified for a range of time steps (for example, outport 3 less than 3200 and output 6 greater than 10.0 during time steps 250 – 3000).

A key feature of MATT is the ability to quickly and easily launch a simulation. Once the input values, time step, and simulation duration have been specified, the user simply chooses to simulate and MATT configures the simulation command, executes the simulation, and captures simulation results.

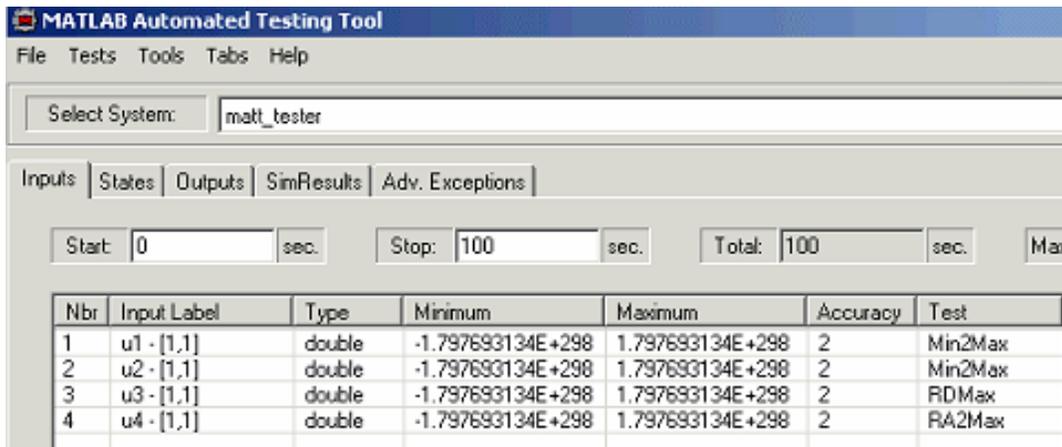
Once simulation has completed, the user can review output value ranges for each outport, quickly discovering where defects occurred. Using this information, the user can review the inport values and outport values for each simulation step, graph any combination of inports and outports, or revise defect criteria to narrow the investigation of simulation output.

Lastly, the user can generate and then test the automatically generated source code using the simulation input as test data. MATT automates this process and then compares simulation output to code execution output.

## Example of Using MATT

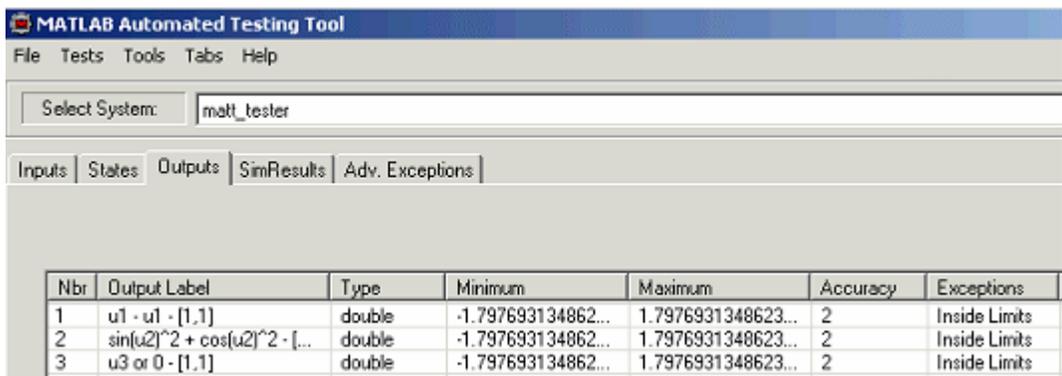
A Matlab/Simulink developer integrates MATT into his or her development process. Using a bottom up approach (meaning the subsystems are created and tested before being integrated into the system), the developer creates a subsystem model within Simulink.

Once the user has the subsystem in a state that permits a simulation to be performed (all blocks connected, inports and outports specified, etc.), MATT can be launched from the Matlab command line. MATT gathers information about the subsystem, including inport and outport specifications, as shown below.



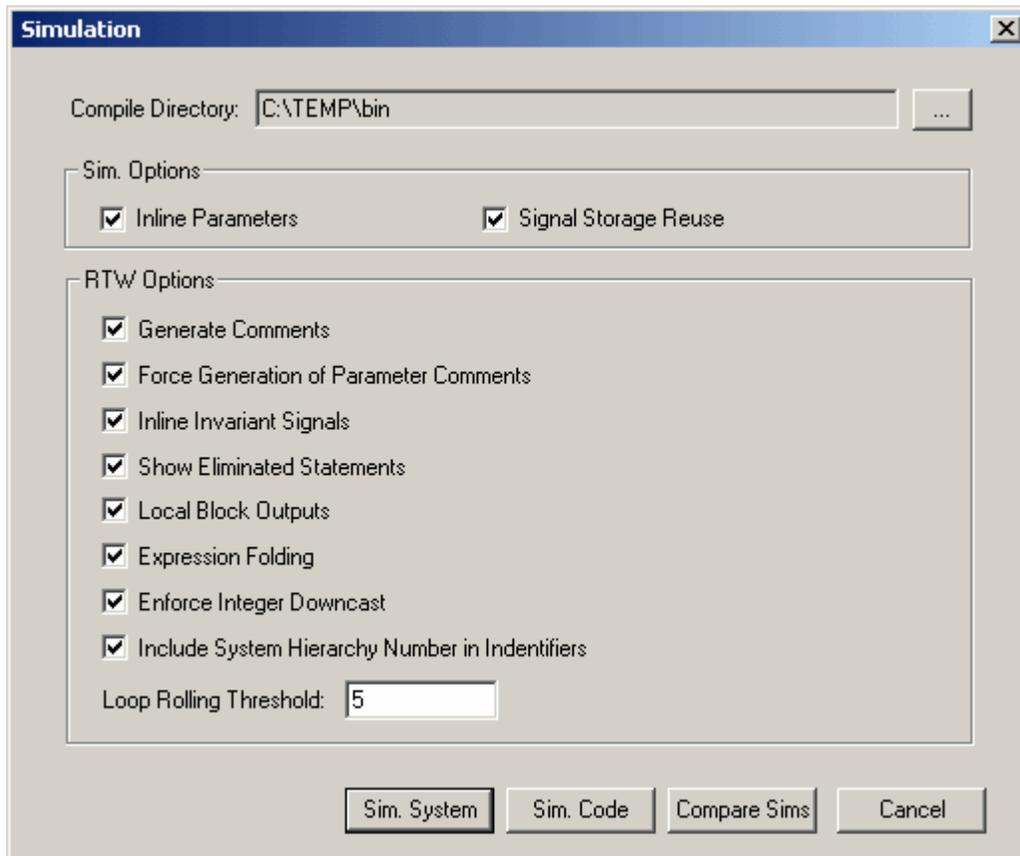
The user then specifies the time step duration and simulation length. Focus shifts to the inport simulation test values. The user can specify the accuracy (number of digits to the right of the decimal point for floating point inport types) and the input values for each inport. The user might choose to use built-in functions, such sine or linear ascending, constant values, or user specified values from the GIST tool to generate input values for the inports.

Once functions and values have been chosen for inports, the user can specify defect criteria or simulate the subsystem. Defect criteria are applied to output values following simulation so they can be specified and revised any time after a simulation has been performed, again as shown below.



The user specifies defect criteria for each output by setting the acceptable range of output values (which can be a single value if desired), and then chooses whether the test function should be above, below, within, or outside the specified range. Defect criteria can be applied to a subset of time steps and can be specified as a conjunction of output values (output 3 and output 6 criteria).

Simulation can be launched by a simple drop-down menu command. MATT allows complete configuration of the simulation. MATT captures output values for each time step and saves these values so that defect criteria can be applied repeatedly if desired.



Once a simulation has been completed, the user can review the defects detected by viewing specific values for each time step. MATT uses Matlab graphing functions to permit the user to view any combination of inport and outport values over the simulation time period.

Lastly, but importantly, MATT can save the input values or input and output values for comparison and use with subsequent versions of the subsystem. This permits the user to quickly test, revise, and retest the subsystem using saved test data. Multiple tests can be saved for a subsystem so that a complete set of unit tests can be used throughout development and maintenance of a system.