Getting Started with MATLAB/SIMULINK

Course Overview

MATLAB is a high-level technical computing language and interactive environment for algorithm development, data visualisation, data analysis, and numeric computation. Using MATLAB, you can solve technical computing problems faster than with traditional programming languages, such as C, C++, and Fortran.

You can use MATLAB in a wide range of applications, including signal and image processing, communications, control design, test and measurement, financial modeling and analysis, and computational biology. Add-on toolboxes (collections of special-purpose MATLAB functions, available separately) extend the MATLAB environment to solve particular classes of problems in these application areas. MATLAB provides a number of features for documenting and sharing your work. You can integrate your MATLAB code with other languages and applications, and distribute your MATLAB algorithms and applications. MATLAB is an integrated technical computing environment that combines numeric computation, advanced graphics and visualisation, and a high-level programming language.

Whatever the objective - an algorithm, analysis, graph, report, or simulation - MATLAB gets you there. The flexible, interactive MATLAB language lets engineers, scientists and business analysts express their technical ideas simply. The extensive and powerful numeric computing methods and graphics allows testing and exploring alternative ideas easily, while the integrated development environment makes it easy to produce fast, practical results.

MATLAB is used in a wide variety of areas in industry including process industries, automotive, finance and economics, biotech/pharmaceutical and education. The open architecture makes it easy to use MATLAB and companion products to explore data and create custom tools that provide early insights and competitive advantages. MATLAB also features a family of application-specific solutions called toolboxes. Very important to most users of MATLAB, toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment in order to solve particular classes of problems. Researched and developed by experts in their fields, toolboxes let you learn, apply, and compare best-of-class techniques, allowing you to evaluate different approaches without writing the code. Probably the most important feature of MATLAB is its easy extensibility. You can also link to external software and data from MATLAB. MATLAB code and data formats are platform independent, so sharing your ideas and designs across PC, Unix and Macintosh platforms is seamless.

Simulink is a platform for multidomain simulation and Model-Based Design of dynamic systems. It provides an interactive graphical environment and a customisable set of block libraries that let you accurately design, simulate, implement, and test control, signal processing, communications, and other time-varying systems. Add-on products extend the Simulink environment with tools for specific modeling and design tasks and for code generation, algorithm implementation, test, and verification.

Simulink is integrated with MATLAB, providing immediate access to an extensive range of tools for algorithm development, data visualisation, data analysis and access, and numerical computation.

Simulink is a block-diagram modelling environment for simulating dynamic systems, evaluating performance, and refining control, DSP, and communications system designs. Simulink block diagrams

provide a highly interactive environment for nonlinear simulation. You can run simulations from both pull-down menus or in batch mode from the command line. Results are displayed "live" during simulations using scope and graph blocks.

Where MATLAB offers a familiar programming environment, Simulink and Stateflow provide a graphical design environment for modelling and simulating complex control, DSP and supervisory logic systems. Built on MATLAB, these products can call any MATLAB function including user-written routines, allowing you to combine the best of both approaches. Even toolbox functions can be embedded within Simulink block-diagram models.

Simulink Blocksets provide specialist block components for use in your Simulink Models. There are blocksets for DSP, Fixed Point, Power Systems, Dials and Gauages, Communications, CDMA reference, nonlinear control design, aerospace, mechanical systems, virtual reality and various DSP targets. A list of all Simulink related products can be found here.

Generate C and C++ code from Simulink and Stateflow models
Simulink Coder™ (formerly Real-Time Workshop®) generates and executes C and C++ code from Simulink® diagrams, Stateflow® charts, and MATLAB® functions. The generated source code can be used for real-time and nonreal-time applications, including simulation acceleration, rapid prototyping, and hardware-in-the-loop testing. You can tune and monitor the generated code using Simulink or run and interact with the code outside MATLAB and Simulink.

Generate C and C++ code optimized for embedded systems
Embedded Coder™ generates readable, compact, and fast C and C++ code for use on embedded processors, on-target rapid prototyping boards, and microprocessors used in mass production. Embedded Coder enables additional MATLAB Coder™ and Simulink Coder™ configuration options and advanced optimizations for fine-grain control of the generated code's functions, files, and data. These optimizations improve code efficiency and facilitate integration with legacy code, data types, and calibration parameters used in production. You can incorporate a third-party development environment into the build process to produce an executable for turnkey deployment on your embedded system.

Course Objectives:

This course provides a comprehensive introduction to the MATLAB technical computing environment. This course is intended for beginning users and also for those looking for a review. No prior programming experience or knowledge of MATLAB is assumed, and the course is structured to allow thorough assimilation of ideas through hands-on examples and exercises. MATLAB competency is developed in a natural way, with an emphasis on practical application. Themes of data analysis, visualization, modeling, and programming are explored throughout the course.

This course also focuses on the details of data management and visualization techniques, from reading various formats of data files to producing customized publication-quality graphics. The course emphasizes creating scripts that extend the basic features provided by MATLAB. Hands-on examples explore features for efficiently organizing and presenting data, providing a practical set of tools for further data analysis.

It also provides hands-on experience using the features in the MATLAB language to write efficient, robust, and well-organized code. These concepts form the foundation for writing full applications,

developing algorithms, and extending built-in MATLAB capabilities. Details of performance optimization are covered throughout the one-day course, as well as tools for writing, debugging, and profiling code.

It provides details about how to build GUIs in MATLAB from the command line by using GUIDE, the GUI development environment. The course introduces concepts for designing and laying out GUIs. Demos show how you can link actions defined by code to a user interface object, such as a push button. You will also learn how to create custom menus for GUIs.

This course is for engineers who are new to system and algorithm modeling and design validation in Simulink. It demonstrates how to apply basic modeling techniques and tools to develop Simulink block diagrams. Topics include:

- Creating and modifying Simulink models and simulating system dynamics
- Modeling continuous-time, discrete-time, and hybrid systems
- Modifying solver settings for simulation accuracy and speed
- Building hierarchy into a Simulink model
- Creating reusable model components using subsystems, libraries, and model references

Intended Audience:

This training course in MATLAB is ideal for any person beginning to use MATLAB and/or would like to learn about finer points of the MATLAB environment. The course would be particularly beneficial for individuals working in a team with other MATLAB -based designers who are required to get "up to speed" quickly. The training course also provides an opportunity to evaluate the MATLAB product for future or more extensive use in your Organization or Institute.

Course Contents:

UNIT 1

Introduction

- What Is MATLAB?
- The MATLAB System
- Development Environment
- Starting and Quitting MATLAB
- MATLAB Desktop

Desktop Tools

UNIT 2

Manipulating Matrices

Matrices and Magic Squares

- Entering Matrices
- sum, transpose, and diag
- Subscripts
- The Colon Operator
- The magic Function

Expressions

- Variables
- Numbers
- Operators
- Functions

- The Empty Matrix
- Scalars
- Vectors

Full and Sparse Matrices

- Sparse Matrix Functions

Multidimensional Arrays

- Overview
- Creating Multidimensional Arrays
- Accessing Multidimensional Array Properties
- Indexing Multidimensional Arrays
- Reshaping Multidimensional Arrays
- Permuting Array Dimensions
- Computing with Multidimensional Arrays
- Organizing Data in Multidimensional Arrays
- Multidimensional Cell Arrays
- Multidimensional Structure Arrays

Summary of Matrix and Array Functions

Controlling Command Window Input and Output

- The format Command
- Suppressing Output
- Entering Long Command Lines
- Command Line Editing

UNIT 3

Data Types

Overview of MATLAB Data Types

Numeric Types

- Integers
- Floating-Point Numbers
- Complex Numbers
- Infinity and NaN
- Identifying Numeric Types
- Display Format for Numeric Values
- Function Summary

Logical Types

- Creating a Logical Array
- How Logical Arrays Are Used
- Identifying Logical Arrays

Characters and Strings

- Creating Character Arrays
- Cell Arrays of Strings
- String Comparisons
- Searching and Replacing
- Converting from Numeric to String
- Converting from String to Numeric
- Function Summary

Dates and Times

- Types of Date Formats

- Conversions Between Date Formats
- Date String Formats
- Output Formats
- Current Date and Time
- Function Summary

Structures

- Building Structure Arrays
- Accessing Data in Structure Arrays
- Using Dynamic Field Names
- Finding the Size of Structure Arrays
- Adding Fields to Structures
- Deleting Fields from Structures
- Applying Functions and Operators
- Writing Functions to Operate on Structures
- Organizing Data in Structure Arrays
- Nesting Structures
- Function Summary

Cell Arrays

- Creating Cell Arrays
- Obtaining Data from Cell Arrays
- Deleting Cells
- Reshaping Cell Arrays
- Replacing Lists of Variables with Cell Arrays
- Applying Functions and Operators
- Organizing Data in Cell Arrays
- Nesting Cell Arrays
- Converting Between Cell and Numeric Arrays
- Cell Arrays of Structures
- Function Summary

Function Handles

- Constructing and Invoking a Function Handle
- Calling a Function Using Its Handle
- Simple Function Handle Example

UNIT 4

Programming with MATLAB

Basic Program Components

Variables

- Types of Variables
- Naming Variables
- Guidelines to Using Variables
- Scope of a Variable
- Lifetime of a Variable

Keywords

- Special Values

Operators

- Arithmetic Operators
- Relational Operators

Function Arguments

- Checking the Number of Input Arguments
- Passing Variable Numbers of Arguments
- Returning Output Arguments

Function Handles

- Constructing a Function Handle
- Calling a Function Using Its Handle
- Functions That Operate on Function Handles
- Additional Information on Function Handles

Calling Functions

- What Happens When You Call a Function
- Determining Which Function Is Called
- MATLAB Calling Syntax
- Passing Certain Argument Types
- Passing Arguments in Structures or Cell Arrays
- Calling External Functions

Types of Functions

Overview of MATLAB Function Types

Anonymous Functions

- Constructing an Anonymous Function
- Arrays of Anonymous Functions
- Outputs from Anonymous Functions
- Variables Used in the Expression
- Examples of Anonymous Functions

Primary M-File Functions

Nested Functions

- Writing Nested Functions
- Calling Nested Functions
- Variable Scope in Nested Functions
- Using Function Handles with Nested Functions
- Examples of Nested Functions

Subfunctions

- Calling Subfunctions
- Accessing Help for a Subfunction

UNIT 5

Graphics

Basic Plotting

- Creating a Plot
- Multiple Data Sets in One Graph
- Specifying Line Styles and Colors
- Plotting Lines and Markers
- Imaginary and Complex Data
- Adding Plots to an Existing Graph
- Figure Windows .
- Multiple Plots in One Figure
- Controlling the Axes
- Axis Labels and Titles

- Saving a Figure

Editing Plots

- Interactive Plot Editing
- Using Functions to Edit Graphs
- Using Plot Editing Mode
- Using the Property Editor

Mesh and Surface Plots

- Visualizing Functions of Two Variables

Images

Animations

- Erase Mode Method
- Creating Movies

UNIT 6

Data Import and Export

Overview

- Text Data
- Graphics Files
- Audio and Audio/Video Data
- Spreadsheets
- Scientific Formats
- The Internet
- Low-Level File I/O
- Large Data Sets
- Toolboxes for Importing Data

Using the Import Wizard

- Using the Import Wizard with Text Data
- Using the Import Wizard with Binary Data

Supported File Formats

Saving and Loading MAT-Files

- Exporting Data to MAT-Files
- Importing Data from MAT-Files

Importing Text Data

- The MATLAB Import Wizard
- Using Import Functions with Text Data
- Importing Numeric Text Data
- Importing Delimited ASCII Data Files
- Importing Numeric Data with Text Headers
- Importing Mixed Alphabetic and Numeric Data
- Importing from XML Documents

Exporting Text Data

- Exporting Delimited ASCII Data Files
- Using the diary Function to Export Data
- Exporting to XML Documents

Working with Graphics Files

- Getting Information About Graphics Files
- Importing Graphics Data
- Exporting Graphics Data

Working with Audio and Video Data

- Getting Information About Audio/Video Files
- Importing Audio/Video Data
- Exporting Audio/Video Data

Working with Spreadsheets

- Microsoft Excel Spreadsheets
- Lotus 123 Spreadsheets

Using Low-Level File I/O Functions

- Opening Files
- Reading Binary Data
- Writing Binary Data
- Controlling Position in a File
- Reading Strings Line by Line from Text Files
- Reading Formatted ASCII Data
- Writing Formatted Text Files
- Closing a File

UNIT 7

Improving Performance and Memory Usage

Analyzing Your Program's Performance

- The M-File Profiler Utility
- Stopwatch Timer Functions

Techniques for Improving Performance

- Vectorizing Loops
- Preallocating Arrays
- Coding Loops in a MEX-File
- Assigning to Variables
- Operating on Real Data
- Using Appropriate Logical Operators
- Overloading Built-In Functions
- Functions Are Generally Faster Than Scripts
- Load and Save Are Faster Than File I/O Functions
- Avoid Large Background Processes

Making Efficient Use of Memory

- Memory Management Functions
- Preallocating Arrays to Reduce Fragmentation
- Enlarging Arrays with *repmat*
- Working with Variables
- Converting Full Matrices into Sparse
- Structure of Arrays vs. Array of Structures
- Working with Large Amounts of Data

Resolving "Out of Memory" Errors

- General Suggestions For Reclaiming Memory
- Compressing Data in Memory
- Increasing System Swap Space
- Freeing Up System Resources on Windows Systems

UNIT 8

Programming Tips

Command and Function Syntax

- Syntax Help
- Command and Function Syntaxes
- Command Line Continuation
- Completing Commands Using the Tab Key
- Recalling Commands
- Clearing Commands
- Suppressing Output to the Screen

Help .

- Using the Help Browser
- Help on Functions from the Help Browser
- Help on Functions from the Command Window
- Topical Help
- Paged Output
- Writing Your Own Help
- Help for Subfunctions and Private Functions
- Help for Methods and Overloaded Functions

Development Environment

- Workspace Browser .
- Using the Find and Replace Utility
- Commenting Out a Block of Code
- Creating M-Files from Command History

M-File Functions

- M-File Structure
- Using Lowercase for Function Names
- Getting a Function's Name and Path
- What M-Files Does a Function Use?
- Dependent Functions, Built-Ins, Classes

Function Arguments

- Getting the Input and Output Arguments
- Variable Numbers of Arguments
- String or Numeric Arguments
- Passing Arguments in a Structure
- Passing Arguments in a Cell Array

Program Development

- Planning the Program
- Using Pseudo-Code
- Selecting the Right Data Structures
- General Coding Practices
- Naming a Function Uniquely
- The Importance of Comments
- Coding in Steps
- Making Modifications in Steps
- Functions with One Calling Function
- Testing the Final Program

Debugging

- The MATLAB Debug Functions

- More Debug Functions
- The MATLAB Graphical Debugger
- A Quick Way to Examine Variables
- Setting Breakpoints from the Command Line
- Finding Line Numbers to Set Breakpoints
- Stopping Execution on an Error or Warning
- Locating an Error from the Error Message
- Using Warnings to Help Debug
- Making Code Execution Visible
- Debugging Scripts

Variables

- Rules for Variable Names
- Making Sure Variable Names Are Valid
- Don't Use Function Names for Variables
- Checking for Reserved Keywords
- Avoid Using i and j for Variables
- Avoid Overwriting Variables in Scripts
- Persistent Variables
- Protecting Persistent Variables
- Global Variables

Strings

- Creating Strings with Concatenation
- Comparing Methods of Concatenation
- Store Arrays of Strings in a Cell Array
- Converting Between Strings and Cell Arrays
- Search and Replace Using Regular Expressions

Evaluating Expressions

- Find Alternatives to Using eval
- Assigning to a Series of Variables
- Short-Circuit Logical Operators
- Changing the Counter Variable within a for Loop

MATLAB Path

- Precedence Rules
- File Precedence
- Adding a Directory to the Search Path
- Handles to Functions Not on the Path
- Making Toolbox File Changes Visible to MATLAB
- Making Nontoolbox File Changes Visible to MATLAB
- Change Notification on Windows

Program Control

- Using break, continue, and return
- Using switch Versus if
- MATLAB case Evaluates Strings
- Multiple Conditions in a case Statement
- Implicit Break in switch-case
- Variable Scope in a switch
- Catching Errors with try-catch
- Nested try-catch Blocks

- Forcing an Early Return from a Function

Save and Load

- Saving Data from the Workspace
- Loading Data into the Workspace
- Viewing Variables in a MAT-File
- Appending to a MAT-File
- Save and Load on Startup or Quit
- Saving to an ASCII File

Files and Filenames

- Naming M-files
- Naming Other Files
- Passing Filenames as Arguments
- Passing Filenames to ASCII Files
- Determining Filenames at Run-Time
- Returning the Size of a File

Input/Output

- File I/O Function Overview
- Common I/O Functions
- Readable File Formats
- Using the Import Wizard
- Loading Mixed Format Data
- Reading Files with Different Formats
- Reading ASCII Data into a Cell Array
- Interactive Input into Your Program

Starting MATLAB

Getting MATLAB to Start Up Faster

Operating System Compatibility

- Executing O/S Commands from MATLAB
- Searching Text with grep
- Constructing Paths and Filenames
- Finding the MATLAB Root Directory
- Temporary Directories and Filenames

UNIT 9

Mathematical Functions and Applications

- Trigonometry
- Complex Numbers
- Signal Representation, Processing, and Plotting
- Poly nomials
- Partial Fraction Expansion
- Functions of Two Variables
- User-Defined Functions
- Plotting Functions
- Data Analysis
- Maximum and Minimum
- Sums and Products
- Statistical Analy sis
- Random Number Generation

Solutions to Systems of Linear Equations

Curve Fitting and Interpolation

- MinimumMean-Square Error Curve Fitting
- Interpolation

Integration and Differentiation

- Numerical Integration .
- Numerical Differentiation

Strings, Time, Base Conversion and Bit Operations

- Character Strings
- Time Computations
- Base Conversions and Bit Operations

Symbolic Processing

- Symbolic Expressions and Algebra
- Manipulating Trigonometric Expressions
- Evaluating and Plotting Symbolic Expressions
- Solving Algebraic and Transcendental Equations

UNIT 10

ELEMENTS OF GUI DESIGN

WHAT IS A MATLAB GRAPHICAL USER INTERFACE?

THE THREE PHASES OF INTERFACE DESIGN

- Analysis
- Design
- User Considerations
- The Reason for the GUI
- Cognitive Considerations
- Physical Considerations
- Paper Prototyping
- Appearance
- Construction

UI CONTROL ELEMENTS

- The Styles
- Check Boxes
- Editable Text
- Frames
- Pop-Up Menus
- List Boxes
- Push Buttons
- Toggle Buttons
- Radio Buttons
- Sliders
- Static Text
- UI Control Properties
- Uicontrol BackgroundColor
- Uicontrol ButtonDownFcn
- Uicontrol CData
- Uicontrol CallBack
- Uicontrol Enable

- Uicontrol Extent
- Uicontrol ForegroundColor
- Uicontrol Font Angle, Name, Size, Units, and Weight
- Uicontrol HorizontalAlignment
- Uicontrol Min, Max, and Value
- Uicontrol SliderStep
- Uicontrol TooltipString
- Uicontrol Position
- Uicontrol String
- Style
- ListBoxTop
- Uicontrol Units
- Uicontrol Interruptible
- Uicontrol Tag
- Uicontrol UserData
- Uicontrol Visible
- Other UI Control Properties
- Creating Uicontrol Objects
- Uicontrol Object Layering
- Framing Objects
- A Stretchable GUI
- Predefined GUIs and Dialog Boxes

UIMENU ELEMENTS

- Uimenu Properties
- Uimenu Accelerator
- Uimenu CallBack
- Uimenu Checked
- Uimenu Children
- Uimenu Enable
- Uimenu ForegroundColor
- Uimenu Label
- Uimenu Position
- Uimenu Separator
- Uimenu Interruptible
- Uimenu Tag
- Uimenu UserData
- Uimenu Visible
- Other Uimenu Properties
- Creating Uimenus
- Top Level Uimenu
- Menu Items and Submenu Titles

HIGH-LEVEL GUI DEVELOPMENT – GUIDE

- The Layout Editor
- The Property Inspector
- The Object Browser
- The Menu Editor
- Saving the GUI
- The GUIDE Created FIG-File

- The GUIDE Created M-File
- Executing a GUI
- Editing a Previously Created GUI

COMMON PROGRAMMING DESIRES WITH UI OBJECTS

- Creating Exclusive Radio Buttons
- Linking Sliders and Editable Text Objects
- Editable Text and Pop-Up Menu
- Windowed Frame and Interruptions
- Toggling Menu Labels
- Customizing a Button with Graphics

UNIT 11:

SIMULINK – Getting Started

- Starting SIMULINK
- Basic Elements
- Building a System
- Running Simulations
- Start using SIMULINK
    - Block Libraries
    - Create a new Model
    - Wiring techniques
    - Help window
    - Configuration
    - Examples
- Useful Features
    - Comments/Labels
    - Align and Distribute Blocks
    - Flip Blocks
    - Hide Names
- Data-driven Modeling
    - Command window
    - m-file
    - Simulation Commands
- Hybrid Systems (continuous and discrete)
- Example: Mass-Spring-Damper System
    - Model
    - SIMULINK
    - Results
- Embedded Algorithms
- Subsystems
- Model Explorer
- Examples

UNIT 12:

SIMULINK Features

- Modeling, Simulation and Analysis with SIMULINK
    - Tool for Model Based Design
    - Tool for Simulation

    - Tool for Analysis
    - Interaction with MATLAB Environment
- Model Based Design
    - Model Based Design Process
    - Defining the System
    - Identifying System Components
    - Modeling the System with Equations
    - Building the SIMULINK Block Diagram
    - Running the Simulation
    - Validating the Simulation Results

SIMULINK Software basics

- Start the SIMULINK Software
    - Open SIMULINK Library Browser
    - Create New SIMULINK Model
    - Open an Existing Model
- SIMULINK User Interface
    - SIMULINK Library Browser
    - SIMULINK Editor

Create a SIMULINK Model

- Creating a Simple Model
    - Overview of the Simple Model
    - Create the Simple Model
    - Connect Blocks in the Simple Model
    - Simulate the Simple Model

Modeling a Dynamic Control System

- Dynamic Control System Model
- Anatomy of the Control System Model
    - Open the Example Model
    - Overview of the Example Model
    - Subsystems in the Example Model
    - Subsystems and Masks
    - Creating a Subsystem
    - Creating a Subsystem Mask
- Simulate the Control System Model
    - Run the Simulation
    - Change the Model Settings
- Move Data Between MATLAB and SIMULINK Model
    - Import Data from MATLAB Workspace
    - Export Simulation Data to MATLAB Workspace

Basic Simulation Workflow

- Simulation with Data Import and Signal Inspection

UNIT 13:

Using SIMULINK as Design Tool

- How SIMULINK Works
- Working with Sample Times
- Creating a Model
- Creating Conditional Subsystems

- Referencing a Model
- Working with Blocks
- Working with Block Libraries
- Working with Block Masks
- Working with Signals
- Working with Variable Sized Signals
- Using Composite Signals
- Working with Data
- Using Enumerated Data
- Importing and Exporting Data
- Working with Data Stores
- Working with Lookup Tables
- Modeling with SIMULINK
- Exploring, Searching and Browsing Models
- Managing Configuration Sets
- Running Simulations
- Running a Simulation Programmatically
- Improving Simulation Performance and Accuracy
- Visualization Simulation Results
- Analyzing Simulation Results
- SIMULINK Debugger
- Accelerating Models
- Customizing SIMULINK User Interface
- Creating Custom Blocks
- Using the Embedded Function Block
- Print Frame Editor

UNIT 14:

Advanced Usage of SIMULINK

- Building a Model
- Setting Simulink Preferences
- Modeling Dynamic Systems
- Simulating Dynamic Systems
- Modeling and Simulating Discrete Systems
- Opening Models
- Entering SIMULINK Commands
- SIMULINK Windows
- Saving a Model
- Printing a Block Diagram
- Getting a Model Report

Creating a Model

- Selecting Objects
- Specifying Block Diagram Colors
- Connecting Blocks
- Annotating Diagrams
- Creating Subsystems
- Creating Conditionally Executed Subsystems
- Modeling with Control Flow Blocks

- Referencing Models
- Model Discretizer
- Using Callback Routines
- Working with Model Workspaces
- Managing Model Versions

Working with Blocks

- About Blocks
- Editing Blocks
- Setting Block Parameters
- Changing a Block's Appearance
- Displaying Block's Outputs
- Controlling and Displaying the Sorted Order
- Lookup Table Editor
- Working with Block Libraries
- Accessing Block Data During Simulation

Working with Signals

- Signal Basics
- Determining Output Signal dimensions
- The Signal & Scope Manager
- The Signal Selector
- Logging Signals
- Signal Properties Dialog Box
- Working with Test Points
- Displaying Signal Properties
- Working with Signal Groups
- Bus Editor

Working with Data

- Working with Data Types
- Working with Data Objects
- Sub classification of SIMULNK Data Classes
- Associating User Data with Blocks
- Modeling Equations
- Avoiding Invalid Loops
- Tips for Building Models

Exploring, Searching and Browsing Models

- The Model Explorer
- The Finder
- The Model Browser

Running Simulations

- Simulations Basics
- Specifying a Simulation Start Time and Stop Time
- Choosing a Solver
- Importing and Exporting Simulation Data
- Configuration Sets
- The Configuration Parameters Dialog Box
- Diagnosing Simulation Errors
- Improving Simulation Performance and Accuracy
- Running a Simulation Programmatically

Analyzing Simulation Results

- Viewing Output Trajectories

Creating Masked Subsystems

- About Masks
- Masked Subsystem Example
- Masking a Subsystem
- The Mask Editor
- Linking Mask Parameters to Block Parameters
- Creating Dynamic Dialogs for Masked Blocks