**Lecture 9**

**Signal Basics**

This section provides an overview of Simulink signals and explains how to specify, display, and check the validity of signal connections.

**About Signals**

Simulink defines signals as the outputs of dynamic systems represented by blocks in a Simulink diagram and by the diagram itself. The lines in a block diagram represent mathematical relationships among the signals defined by the block diagram. For example, a line connecting the output of block A to the input of block B indicates that the signal output by B depends on the signal output by A.

**Creating Signals**

You can create signals by creating source blocks in your model. For example, you can create a signal that varies sinusoidally with time by dragging an instance of the Sine block from the Simulink Sources library into the model.

**Signal Labels**

A signal label is text that appears next to the line that represents a signal that has a name. The signal label displays the signal’s name. In addition, if the signal is a virtual signal and its show propagated signals property is on the label displays the names of the signals that make up the virtual signal.

Simulink creates a label for a signal when you assign it a name in the **Signal Properties** dialog box. You can change the signal’s name by editing its label on the block diagram. To edit

the label, left-click the label. Simulink replaces the label with an edit field. Edit the name in the edit field, the press **Enter** or click outside the label to confirm the change.

**Displaying Signal Values**

As with creating signals, you can use either blocks or the Signal & Scope Manager to display the values of signals during a simulation. For example, you can use either the Scope block or the Signal & Scope Manager to graph time-varying signals on an oscilloscope-like display during simulation.

**Signal Data Types**

Data type refers to the format used to represent signal values internally. The data type of Simulink signals is double by default. However, you can create signals of other data types. Simulink supports the same range of data types as MATLAB.

**Signal Dimensions**

Simulink blocks can output one- or two-dimensional signals. A one-dimensional (1-D) signal consists of a stream of one-dimensional arrays output at a frequency of one array (vector) per simulation time step. A two-dimensional (2-D) signal consists of a stream of two-dimensional arrays emitted at a frequency of one 2-D array (matrix) per block sample time. The Simulink user interface and documentation generally refer to 1-D signals as *vectors* and 2-D signals as *matrices*. A one-element array is frequently referred to as a *scalar*. A *row vector* is a 2-D array that has one row. A *column vector* is a 2-D array that has one column.

**Complex Signals**

The values of Simulink signals can be complex numbers. A signal whose values are complex numbers is called a complex signal. You can introduce a complex-valued signal into a model in the following ways:

**•**Load complex-valued signal data from the MATLAB workspace into the model via a root-level inport.

**•**Create a Constant block in your model and set its value to a complex number.

**•**Create real signals corresponding to the real and imaginary parts of a complex signal, then combine the parts into a complex signal, using the Real-Imag to Complex conversion block.

You can manipulate complex signals via blocks that accept them.

**Virtual Signals**

A *virtual signal* is a signal that represents another signal graphically. Some blocks, such as Bus Creator, Inport, and Outport blocks generate virtual signals either exclusively or optionally. Virtual signals are purely graphical entities. They have no mathematical or physical significance. Simulink ignores them when simulating a model. Whenever you run or update a model, Simulink determines the nonvirtual signal(s) represented by the model’s virtual signal(s), using a procedure known as *signal propagation*. When running the model, Simulink uses the corresponding nonvirtual signal(s), determined via signal propagation, to drive

the blocks to which the virtual signals are connected. Consider, for example, the following model.



The signals driving Gain blocks G1 and G2 are virtual signals corresponding to signals s2 and s1, respectively. Simulink determines this automatically whenever you update or simulate the model.

The **Show propagated signals** option displays the nonvirtual signals represented by virtual signals in the labels of the virtual signals.



**Control Signals**

A *control signal* is a signal used by one block to initiate execution of another block, e.g., a function-call or action subsystem. When you update or start simulation of a block diagram, Simulink uses a dash-dot pattern to redraw lines representing the diagram’s control signals as illustrated in the following example.



**Signal Buses**

A bus is a composite signal comprising a set of signals represented graphically by a bundle of lines. It is analogous to a bundle of wires held together by tie wraps. The components of a bus can have different data types and can themselves be composite signals (i.e., buses or muxed signals). You can use Bus Creator and Inport blocks to create signal buses and Bus Selector blocks to access a bus’s components.



Selecting a bus and then **Signal Dimensions** from the model editor’s **Format** menu displays the number of signal components carried by the bus.

**Virtual Versus Nonvirtual Buses**

Buses may be either virtual or nonvirtual. During simulation, blocks connected to a virtual bus read their inputs from memory allocated to the component signals, which may reside in noncontiguous areas of memory. By contrast, blocks connected to a nonvirtual bus read their inputs from a copy of the component signals maintained by Simulink in a contiguous area of memory allocated to the bus.

Some Simulink features, such as model referencing, require use of nonvirtual signals. Others require virtual buses. Nonvirtual buses also facilitate code generation by enabling buses to be

represented by data structures. On the other hand, nonvirtual buses can save memory where nonvirtual buses are not required.

The Bus Creator and Inport blocks output virtual buses by default. To cause them to output a nonvirtual bus, select the **Output as structure** option on their parameter dialog boxes. You can also use the Signal Conversion block to convert nonvirtual to virtual buses, and vice versa.

**Bus-Capable Blocks**

A *bus-capable block* is a block through which both virtual and nonvirtual buses can pass. All virtual blocks are bus capable. Further, the following nonvirtual blocks are also bus-capable:

**•**Memory

**•**Merge

**•**Switch

**•**Multiport Switch

**•**Rate Transition

**•**Unit Delay

**•**Zero-Order Hold

Some bus-capable blocks impose constraints on bus propagation through them.