Since wave3D has three dimensions, we do not, and must not, specify a chunk index.

There is one important difference between wave access using 1D waves versus multidimensional waves. For 1D waves alone, Igor performs linear interpolation when the specified index value, whether scaled or unscaled, falls between two points. For multidimensional waves, Igor returns the value of the element whose indices are closest to the specified indices.

When a multidimensional wave is the destination of a wave assignment statement, you can specify a subrange for each dimension. You can specify an entire dimension by using []. For example:

```
wave3D[2][][1,2] = 3
```

This sets row 2 of all columns and layers 1 and 2 to the value 3.

Note that indexing of the form [] (entire dimension) or [1,2] (range of a dimension) can be used on the left-hand side only. This is because the indexing on the left side determines which elements of the destination are to be set whereas indexing on the right side identifies a particular element in the source which is to contribute to a particular value in the destination.

Multidimensional Wave Assignment

As with one-dimensional waves, you can assign a value to a multidimensional wave using a wave assignment statement. For example:

```
Make/O/N=(3,3) wave0_2D, wave1_2D, wave2_2D
wave1_2D = 1.0; wave2_2D = 2.0
wave0_2D = wave1_2D / wave2_2D
```

The last command sets all elements of wave0_2D equal to the quotient of the corresponding elements of wave1_2D and wave2_2D.

Important: Wave assignments as shown in the above example where waves on the right-hand side do not include explicit indexing are defined only when all waves involved have the same dimensionality. The result of the following assignment is undefined and may produce surprising results.

```
Make/O/N=(3,3) wave33
Make/O/N=(2,2) wave22
wave33 = wave22
```

Whenever waves of mismatched dimensionality are used you should specify explicit indexing as described next.

In a wave assignment, Igor evaluates the right-hand side one time for each element specified by the left-hand side. During this evaluation, the symbols p, q, r and s take on the value of the row, column, layer and chunk, respectively, of the element in the destination for which a value is being calculated. For example:

```
Make/O/N=(5,4,3) wave3D = 0
Make/O/N=(5,4) wave2D = 999
wave3D[][][0] = wave2D[p][q]
```

This stores the contents of wave2D in layer 0 of wave3D. In this case, the destination (wave3D) has three dimensions, so p, q and r are defined and s is undefined. The following discussion explains this assignment and presents a way of thinking about wave assignments in general.

The left-hand side of the assignment specifies that Igor is to store a value into all rows (the first []) and all columns (the second []) of layer zero (the [0]) of wave3D. For each of these elements, Igor will evaluate the right-hand side. During the evaluation, the symbol p will return the row number of the element in wave3D that Igor is about to set and the symbol q will return the column number. The symbol r will have the value 0 during the entire process. Thus, the expression wave2D[p][q] will return a value from wave2D at the corresponding row and column in wave3D.

As the preceding example shows, wave assignments provide a way of transferring data between waves. With the proper indexing, you can build a 2D wave from multiple 1D waves or a 3D wave from multiple

2D waves. Conversely, you can extract a layer of a 3D wave into a 2D wave or extract a column from a 2D wave into a 1D wave. Here are some examples that illustrate these operations.

```
// Build a 2D wave from multiple 1D waves (waveforms)
Make/O/N=5, wave0=p, wave1=p+1, wave2=p+2    // 1D waveforms
Make/O/N=(5,3) wave0_2D
wave0_2D[][0] = wave0[p]      // Store into all rows, column 0
wave0_2D[][1] = wave1[p]      // Store into all rows, column 1
wave0_2D[][2] = wave2[p]      // Store into all rows, column 2

// Build a 3D wave from multiple 2D waves
Duplicate/O wave0_2D, wave1_2D; wave1_2D *= -1
Make/O/N=(5,3,2) wave0_3D
wave0_3D[][][0]= wave0_2D[p][q]  // Store into all rows/cols, layer 0
wave0_3D[][][1]= wave1_2D[p][q]  // Store into all rows/cols, layer 1

// Extract a layer of a 3D wave into a 2D wave
wave0_2D = wave0_3D[p][q][0]     // Extract layer 0 into 2D wave

// Extract a column of a 2D wave into a 1D wave
wave0 = wave0_2D[p][0]           // Extract column 0 into 1D wave
```

To understand assignments like these, first figure out, by looking at the indexing on the left-hand side, which elements of the destination wave are going to be set. (If there is no indexing on the left then all elements are going to be set.) Then think about the range of values that p, q, r and s will take on as Igor evaluates the right-hand side to get a value for each destination element. Finally, think about how these values, used as indices on the right-hand side, select the desired source element.

To create such an assignment, first determine the indexing needed on the left-hand side to set the elements of the destination that you want to set. Then think about the values that p, q, r and s will take on. Then use p, q, r and s as indices to select a source element to be used when computing a particular destination element.

Here are some more examples:

```
// Extract a row of a 2D wave into a 1D wave
Make/O/N=3 row1
row1 = wave0_2D[1][p]         // Extract row 1 of the 2D wave
```

In this example, the *row* index (p) for the destination is used to select the source *column* while the source row is always 1.

```
// Extract a horizontal slice of a 3D wave into a 2D wave
Make/O/N=(2,3) slice_R2       // Slice consisting of all of row 2
slice_R2 = wave0_3D[2][q][p]  // Extract row 2, all columns/layers
```

In this example, the row data for slice_R2 comes from the layers of wave0_3D because the p symbol (row index) is used to select the layer in the source. The column data for slice_R2 comes from the columns of wave0_3D because the q symbol (column index) is used to select the column in the source. All data comes from row 2 in the source because the row index is fixed at 2.

You can store into a range of elements in a particular dimension by using a range index on the left-hand side. As an example, here are some commands that shift the horizontal slices of wave0_3D.

```
Duplicate/O wave0_3D, tmp_wave0_3D
wave0_3D[0][][] = tmp_wave0_3D[4][q][r]
wave0_3D[1,4][][] = tmp_wave0_3D[p-1][q][r]
KillWaves tmp_wave0_3D
```

The first assignment transfers the slice consisting of all elements in row 4 to row zero. The second assignment transfers slice n-1 to slice n. To understand this, realize that as p goes from 1 to 4, p-1 indexes into the preceding row of the source.