```
MatrixOp/O wave1 = var1 - wave2  // Error: Can't subtract a matrix from a scalar
MatrixOp/O wave1 = var1 / wave2  // Error: Can't divide a scalar by a matrix
```

Division of a scalar by a matrix can be accomplished using the reciprocal function as in:

```
MatrixOp/O wave1 = scalar * rec(wave2)
```

The dot operator . is used to compute the generalized dot product:

```
MatrixOp/O wave1 = wave2 . wave3    // Spaces around '.' are optional
```

The x operator designates matrix-matrix multiplication.

```
MatrixOp/O wave1 = wave2 x wave3    // Spaces around 'x' are required!
```

The logical operators && and || are restricted to real-valued data tokens and produce unsigned byte numeric tokens with the value 0 or 1.

The two postfix operators ^t (matrix transpose) and ^h (Hermitian transpose) operate on the first token to their left:

```
MatrixOp/O wave1 = wave2^t + wave3
```

Tthe transpose operation has higher precedence and therefore executes before the addition.

The left token may be a compound token as in:

```
MatrixOp/O wave1 = sumCols(wave2)^t
```

MatrixOp supports the ^ character only in the two postfix operators ^t and ^h. For exponentiation use the *powR* and `powC` functions.

MatrixOp Multiplication and Scaling

MatrixOp provides a number of multiplication and scaling capabilities. They include matrix-scalar multiplication:

```
MatrixOp/O wave1 = wave2 * scalar   // Equivalent to a wave assignment
```

and wave-wave multiplication:

```
MatrixOp wave1 = wave2 * wave3      // Also includes layer-by-layer support
```

and matrix-matrix multiplication:

```
MatrixOp wave1 = wave2 x wave3
```

The latter is equivalent to the **MatrixMultiply** operation with the convenience of allowing you to write and execute complicated compound expressions in one line.

MatrixOp adds two specialized scaling functions that are frequently used. The `scaleCols` function multiplies each column by a different scalar and the `scale` function scales its input to a specified range.

MatrixOp Data Rearrangement and Extraction

MatrixOp is often used to rearrange data in a matrix or to extract a subset of the data for further calculation. You can extract an element or a layer from a wave using square-bracket indexing:

```
// Extract scalar from point a of the wave
MatrixOp destWave = wave1d[a]

// Extract scalar from element a,b of the wave
MatrixOp destWave = wave2d[a][b]

// Extract scalar from element a,b,c of the wave
MatrixOp destWave = wave3d[a][b][c]
```