You can use any combination of data types for operands. In particular, you can mix real and complex types in *expression*. MatrixOp determines data types of inputs and the appropriate output data type at runtime without regard to any type declaration such as `Wave/C`.

MatrixOp functions can create certain types of 2D data tokens. Such tokens are used on the RHS of the assignment. The `const` and `zeroMat` functions create matrices of specified dimensions containing fixed values. The `identity` function creates identity matrixes and `triDiag` creates a tridiagonal matrix from 1D input waves. The **rowRepeat** and `colRepeat` functions also construct matrices from 1D waves. The `setRow`, `setCol` and `setOffDiag` functions transfer data from 1D waves to elements of 2D waves.

MatrixOp and Wave Dimensions

MatrixOp was designed to optimize operations on 2D arrays (matrices) but it also supports other wave dimensions in various expressions. However it does not support zero point waves.

A 1D wave is treated as a matrix with one column. 3D and 4D waves are treated as arrays of layers so their assignments are processed on a layer-by-layer basis. For example:

```
Make/O/N=(4,5) wave2 = q
Make/O/N=(4,5,6) wave3 = r
MatrixOp/O wave1 = wave2 * wave3       // 2D multiplies 3D one layer at a time
```

wave1 winds up as a 3D wave of dimensions (4,5,6). Each layer of wave1 contains the contents of the corresponding layer of wave3 multiplied on an element-by-element basis by the contents of wave2.

Exceptions for layer-by-layer processing are special MatrixOp functions such as `beam` and `transposeVol`.

Some binary operators have dimensional restrictions for their wave operands. For example, matrix multiplication (wave2 x wave3) requires that the number of columns in wave2 be equal to the number of rows in wave3. Regular multiplication (wave2*wave3) requires that the number of rows and the columns of the two be equal. For example,

```
Make/O/N=(4,6) wave2
Make/O/N=(3,8) wave3
MatrixOp/O wave1 = wave2 * wave3       // Error: "Matrix Dimensions Mismatch"
```

The exception to this rule is when the righthand operand is a single wave element:

```
MatrixOp/O wave1 = wave2 * wave3[2][3] // Equivalent to scalar multiplication
```

Some MatrixOp functions operate on each column of the matrix and result in a 1 row by N column wave which may be confusing when used elsewhere in Igor. If what you need is an N row 1D wave you can redimension the wave or simply include a transpose operator in the MatrixOp command:

```
Make/O/N=(4,6) wave2=gnoise(4)
MatrixOp/O wave1=sumCols(wave2)^t      // wave1 is a 1D wave with 6 rows
```

MatrixOp Operators

MatrixOp defines 8 binary operators and two postfix operators which are available for use in the RHS expression. The operators are listed in the Operators section of the reference documentation for **MatrixOp** on page V-550.

MatrixOp does not support operator combinations such as `+=`.

The basic operators, +, -, *, and /, operate in much the same way as they would in a regular wave assignment with one exception: when the - or / operation involves a matrix and a scalar, it is only supported if the matrix is the first operand and the scalar is the second. For example:

```
Make/O wave2 = 1
Variable var1 = 7
MatrixOp/O wave1 = wave2 - var1  // OK: Subtract var1 to each point in wave2
```