Using MatrixOp

When performing mathematical calculations your first choice may be to write standard wave assignments such as:

```
wave1 = wave2 + wave3
```

When the expression on the right-hand-side (RHS) is complicated or when the waves are large you can gain a significant performance improvement using MatrixOp (short for "matrix operation"). MatrixOp was originally conceived to perform calculations on 2D waves and was later extended to waves of any dimension.

To use the basic form of **MatrixOp** simply prepend its name to the assignment statement:

```
MatrixOp wave1 = wave2 + wave3
```

Although the two commands appear similar, having the general form:

```
destWave = expression
```

they are different in behavior.

A significant difference is that MatrixOp operates on pure array elements without regard to wave scaling and does not support indexing using x, y, z, t, p, q, r, or s on the RHS.

The wave assignment requires that the destination wave exist at the time of execution but MatrixOp creates its destination wave automatically. In these commands:

```
Make/O wave2 = 1/(p+1), wave3 = 1
MatrixOp/O wave1 = wave2 + wave3
```

MatrixOp creates a single precision (SP) wave named wave1 and stores the result in it.

When MatrixOp creates a destination wave, its data type and dimensions depend on the expression on the RHS. In the preceding example, MatrixOp created wave1 as SP because wave2 and wave3 are SP. The rules that MatrixOp uses to determine the data type and dimensions of the destination are discussed below under **MatrixOp Data Promotion Policy** on page III-145.

These rules leads to the following difference between MatrixOp and wave assignments:

```
Make/O wave1 = 42       // wave1 is 128 points wave, all points are set to 42

MatrixOp/O wave1 = 42   // wave1 is now 1 point wave set to 42
```

Assignment statements evaluate the RHS and store the result in a point-by-point manner. Because of this, you sometimes get unexpected results if the destination wave appears on the RHS. In this example, `WaveMin(wave2)` changes during the evaluation of the assignment statement:

```
wave2 = wave2 - WaveMin(wave2)
```

By contrast, MatrixOp evaluates the RHS for all points in the destination before storing anything in the destination wave. Consequently this command behaves as expected:

```
MatrixOp/O wave2 = wave2 - minVal(wave2)
```

Another important difference appears when the RHS involves complex numbers. If you try, for example,

```
Make/O/C cwave2
Make/O wave1, wave3
wave1 = cwave2 + wave3
```

you get an error, "Complex wave used in real expression".

By contrast, MatrixOp creates the destination as complex so this command works without error:

```
MatrixOp/O wave1 = cwave2 + wave3
```