**Flags**

/DSTC=*solutionWave*

Specifies the output wave when all input waves are complex. If you omit /DSTC then the output wave is M_TDLinearSolution in the current data folder. /DSTC was added in Igor Pro 9.00.

/DSTR=*solutionWave*

Specifies the output wave when all input waves are real. If you omit /DSTR then the output wave is M_TDLinearSolution in the current data folder. /DSTR was added in Igor Pro 9.00.

/FREE Creates the solution wave as a free wave. /FREE is allowed only in functions and only if the solution wave, as specified by /DSTC or /DSTR, is a simple name or wave reference structure field. /DSTC was added in Igor Pro 9.00.

/Z No error reporting.

**Details**

The input waves can be single or double precision (real or complex). Results are returned in the wave M_TDLinearSolution in the current data folder. The wave *mainW* determines the size of the main diagonal (N). All other waves must match it in size with *upperW* and *mainW* containing one less point and *matrixB* consisting of N-by-NRHS elements of the same data type.

MatrixLinearSolveTD should be more efficient than MatrixLinearSolve with respect to storage requirements.

MatrixLinearSolveTD creates the variable V_flag, which is zero when it finishes successfully.

**See Also**

**Matrix Math Operations** on page III-138; the **MatrixLinearSolve** and **MatrixOp** operations.

# MatrixLLS

**MatrixLLS** [**/O/Z/M=*method***] ***matrixA matrixB***

The MatrixLLS operation solves overdetermined or underdetermined linear systems involving MxN *matrixA*, using either QR/LQ or SV decompositions. Both *matrixA* and *matrixB* must have the same number type. Supported types are real or complex single precision and double precision numbers.

**Flags**

/M=*method* Specifies the decomposition method.

- *method*=0: Decomposition is to QR or LQ (default). Creates the 2D wave M_A, which contains details of the QR/LQ factorization.
- *method*=1: Singular value decomposition. Creates the 2D wave M_A, which contains the right singular vectors stored row-wise in the first `min(m,n)` rows. Creates the 1D wave M_SV, which contains the singular values of *matrixA* arranged in decreasing order.

/O Overwrites *matrixA* with its decomposition and *matrixB* with the solution vectors. This requires less memory.

/Z No error reporting.

**Details**

When the /O flag is not specified, the solution vectors are stored in the wave M_B, otherwise the solution vectors are stored in *matrixB*. Let *matrixA* be *m* rows by *n* columns and *matrixB* be an *m* by NRHS (if NRHS=1 it can be omitted). If $m \geq n$, MatrixLLS solves the least squares solution to an overdetermined system:

$$Minimize\left|\left|matrixB - matrixA \times \mathbf{X}\right|\right|.$$

Here the first *n* rows of M_B contain the least squares solution vectors while the remaining rows can be squared and summed to obtain the residual sum of the squares. If you are not interested in the residual you can resize the wave using, for example:

```
Redimension/N=(n,NRHS) M_B
```

If *m*<*n*, MatrixLLS finds the minimum norm solution of the underdetermined system:

$$matrixA \times \mathbf{X} = matrixB.$$

In this case, the first *m* rows of M_B contain the minimum norm solution vectors while the remaining rows can be squared and summed to obtain the residual sum of the squares for the solution. If you are not interested in the residual you can resize the wave using, for example:

```
Redimension/N=(m,NRHS) M_B
```

**Note**: Here *matrixB* consists of one or more column vectors B corresponding to one or more solution vectors X that are computed simultaneously. If *matrixB* consists of a single column, M_B is a 2D matrix wave that contains a single solution column.

The variable V_flag is set to 0 when there is no error; otherwise it contains the LAPACK error code.

**Examples**

```
// Construct matrixA as a 4x4 matrix
Make/O/D/N=(4,4) matrixA
matrixA[0][0]= {-54,-27,-9,-38}
matrixA[0][1]= {13,60,-42,-26}
matrixA[0][2]= {-80,17,44,-35}
matrixA[0][3]= {98,-8,-72,7}

// Construct matrixB as a single column (1D wave)
Make/O/D/N=4 matrixB={-20,77,-79,-33}

// Solve for matrixA x vectorX = matrixB
MatrixLLS matrixA, matrixB// Output stored in M_B

// Verify the solution
MatrixOP/O/P=1 aa=sum(abs(matrixA x col(M_B,0) - matrixB)) aa={9.947598300641403e-14}

// Compute multiple solutions at once
// Construct matrixB as two columns (2D wave)
Redimension/N=(4,2) matrixB
matrixB[0][0]= {-20,77,-79,-33}
matrixB[0][1]= {-94,61,29,68}

// Perform the calculation
MatrixLLS matrixA, matrixB

// Verify the solutions
MatrixOP/O/P=1 aa=sum(abs(matrixA x col(M_B,0) - col(matrixB,0)))
    aa={9.947598300641403e-14}
MatrixOP/O/P=1 aa=sum(abs(matrixA x col(M_B,1) - col(matrixB,1)))
    aa={1.989519660128281e-13}

// See if we have an overdetermined system ...
Make/D/N=(5,4) matrixA
matrixA[0][0] = {-7,-62,-47,-54,-80}
matrixA[0][1] = {43,15,-9,-94,57}
matrixA[0][2] = {22,61,-95,-95,51}
matrixA[0][3] = {-54,68,81,-51,54}
Make/O/D/N=(5) matrixB
matrixB[0] = {-28,-75,-74,35,86}

// Perform the calculation
MatrixLLS matrixA, matrixB

// Remove the extra rows from M_B
Redimension/N=4 M_B

// M_B is only the least squares solution so there is no point in attempting
// to verify the solution as in the example above.
```