See Also

The **MatrixOp** operation for more efficient matrix operations.

Matrix Math Operations on page III-138 for more about Igor's matrix routines.

MatrixDot

MatrixDot(*waveA, waveB*)

The MatrixDot function calculates the inner (scalar) product for two 1D waves. A 1D wave **A** represents a vector in the sense:

$$\mathbf{A} = \sum \alpha_i \hat{e}_i, \qquad \hat{e}_i \text{ is a unit vector.}$$

Given two such waves **A** and **B**, the inner product is defined as

$$ip = \sum \alpha_i \beta_i .$$

When both *waveA* and *waveB* are complex and the result is assigned to a complex-valued number MatrixDot returns:

$$ipc = \sum \alpha_i^* \beta_i .$$

If you prefer the definition where the second factor is the one that is conjugated, you can simply reverse the order of *waveA* and *waveB* in the function call.

If the result is assigned to a real number, MatrixDot returns:

$$ip = \left| \sum \alpha_i^* \beta_i \right| .$$

If either *waveA* or *waveB* is complex and the result is assigned to a real-valued number, MatrixDot returns:

$$ip = \left| \sum \alpha_i \beta_i \right| .$$

When the result is assigned to a complex-valued number MatrixDot returns:

$$ipc = \sum \alpha_i \beta_i .$$

See Also

The **MatrixOp** operation for more efficient matrix operations.

Matrix Math Operations on page III-138 for more about Igor's matrix routines.

MatrixEigenV

MatrixEigenV [***flags***] ***matrixA*** **[,** ***matrixB*****]**

MatrixEigenV computes the eigenvalues and eigenvectors of a square matrix using LAPACK routines.

Flags for General Matrices

/C
: Creates complex valued output for real valued input waves. This is equivalent to converting the input to complex, with zero imaginary component, prior to computing the eigenvalues and eigenvectors. The main benefit of this format is that the output has simple packing of eigenvalues and eigenvectors. See the example in the **Details for General Matrices** section below.

  The /C flag was added in Igor Pro 7.00.

/B=*balance* Determines how the input matrix should be scaled and or permuted to improve the conditioning of the eigenvalues.

balance=0 (default), 1, 2, or 3, corresponding respectively to N, P, S, or B in the LAPACK routines. Applicable only with the /X flag.

0: Do not scale or permute.

1: Permute.

2: Do diagonal scaling.

3: Scale and permute.

/L Calculates for left eigenvectors.

/O Overwrites *matrixWave*, requiring less memory.

/R Calculates for right eigenvectors.

/S=*sense* Determines which reciprocal condition numbers are calculated.

sense=0 (default), 1, 2, or 3, corresponding respectively to N, E, V, or B in the LAPACK routines. Applicable only with the /X flag.

0: None.

1: Eigenvalues only.

2: Right eigenvectors.

3: Eigenvalues and right eigenvectors.

If sense is 1 or 3 you must compute both left and right eigenvectors.

NOTE: /S is applicable only with the /X flag or for the generalized eigenvalue problem.

/X Uses LAPACK expert routines, which require additional parameters (see /B and /S flags). The operation creates additional waves:

The W_MatrixOpInfo wave contains in element 0 the ILO, in element 1 the IHI, and in element 2 the ABNRM from the LAPACK routines.

The wave W_MatrixRCONDE contains the reciprocal condition numbers for the eigenvalues.

The wave W_MatrixRCONDV contains the reciprocal condition number for the eigenvectors.

Flags for Symmetric Matrices

/SYM Computes the eigenvalues of an NxN symmetric matrix and stores them in the wave W_eigenValues. You must specify this flag if you want to use the special routines for symmetric matrices. The number of eigenvalues is stored in the variable V_npnts. Because W_eigenValues has N points, only the first V_npnts will contain relevant eigenvalues.

When using this flag with complex input the matrix is assumed to be Hermitian.

/EVEC Computes eigenvectors in addition to eigenvalues. Eigenvectors will be stored in the wave M_eigenVectors, which is of dimension NxN. The first V_npnts columns of the wave will contain the V_npnts eigenvectors corresponding to the eigenvalues in W_eigenValues. /EVEC must be preceded by /SYM.

/RNG={*method*,*low*,*high*}

Determines what is computed:

method=0:	Computes all the eigenvalues or eigenvectors (default).
method=1:	Computes eigenvalues for *low* and *high* double precision range.
method=2:	Computes eigenvalues for *low* and *high* integer indices (1 based). For example, to compute the first 3 eigenvalues use: `/RNG={2,1,3}`.

/RNG must be preceded by /SYM.

Flags for Generalized Eigenvalue Problem

These are the same flags as for general (non-symmetric) matrices above except for /O and /X which are not supported for the generalized eigenvalue solution.

Common Flags

/Z	No error reporting (except for setting V_flag).

Details

There are three mutually exclusive branches for the operation. The first is designed for a square matrix input *matrixA*. The operation computes the solution to the problem

$$\mathbf{Ax} = \lambda\mathbf{x},$$

where A is the input matrix, x is an eigenvector and λ is an eigenvalue.

The second branch is designed for symmetric matrices A, i.e., when

$$\mathbf{A} = \mathbf{A}^{\mathbf{T}},$$

where the superscript T denotes a transpose.

The third branch of the operation is designed to solve the generalized eigenvalue problem,

$$\mathbf{Ax} = \lambda\mathbf{Bx},$$

where A and B are square matrices, x is an eigenvector and λ is an eigenvalue.

Each branch of the operation supports its own set of flags as shown above. All branches support input of single and double precision in real or complex waves. If you specify both *matrixA* and *matrixB* then they must have the same number type.

Details for General Matrices

The eigenvalues are returned in the 1D complex wave W_eigenValues. The eigenvectors are returned in the 2D wave M_R_eigenVectors or M_L_eigenVectors.

The calculated eigenvectors are normalized to unit length.

Complex conjugate pairs of eigenvalues appear consecutively with the eigenvalue having the positive imaginary part first.

If the *jth* eigenvalue is real, then the corresponding eigenvector *u*(*j*)=*M*[][*j*] is the *jth* column of M_L_eigenVectors or M_R_eigenVectors. If the *jth* and (*j*+1)*th* eigenvalues form a complex conjugate pair, then *u*(*j*) = *M*[][*j*] + *i***M*[][*j*+1] and *u*(*j*+1) = *M*[][*j*] - *i***M*[][*j*+1].

Example

```
Function TestMatrixEigenVReal()
   Make/D/N=(2,2)/O eee={{0,1},{-1,0}}
   MatrixEigenV/R eee
   Wave W_eigenvalues, M_R_eigenVectors
   MatrixOP/O firstEV=cmplx(col(M_R_eigenVectors,0),col(M_R_eigenVectors,1))
   MatrixOP/O aa=eee x firstEV - W_eigenvalues[0]*firstEV
   Print aa
End

Function TestMatrixEigenVComplex()
   Make/D/N=(2,2)/O eee={{0,1},{-1,0}}
   MatrixEigenV/R/C eee
```

```
    Wave W_eigenvalues, M_R_eigenVectors
    MatrixOP/O aa=eee x col(M_R_eigenVectors,0) - W_eigenValues[0]*col(M_R_eigenVectors,0)
    Print aa
End
```

Details for Symmetric Matrices

The LAPACK routines that compute the eigenvalues and eigenvectors of symmetric matrices claim to use the Relatively Robust Representation whenever possible. If your matrix is symmetric you should use this branch of the operation (/SYM) for improved accuracy.

Details for Generalized Eigenvalue Problem

Here the right eigenvectors (/R) are solutions to the equation

$$\mathbf{Ax} = \lambda\mathbf{Bx},$$

and the left eigenvectors (/L) are solutions to

$$\mathbf{x^H A} = \lambda\mathbf{x^H B},$$

where the superscript H denotes the conjugate transpose.

When both *matrixA* and *matrixB* are real valued, the operation creates the following waves in the current data folder:

W_alphaValues	Contains the complex alpha values.
W_betaValues	Contains the real-valued denominator such that the eigenvalues are given by $\lambda_j = \frac{\alpha_j}{\beta_j}$.
M_leftEigenVectors and M_rightEigenVectors	Real valued waves where columns correspond to eigenvectors of the equation.

Every point in the wave W_alphaValues corresponds to an eigenvalue. When the imaginary part of W_alphaValues[j] is zero, the eigenvalue is real and the corresponding eigenvector is also real e.g., M_rightEigenVectors[][j]. When the imaginary part is positive then there are two eigenvalues that are complex-conjugates of each other with corresponding complex eigenvectors given by

```
cmplx(M_rightEigenVectors[][j],M_rightEigenVectors[][j+1])
```

and

```
cmplx(M_rightEigenVectors[][j],-M_rightEigenVectors[][j+1])
```

When both *matrixA* and *matrixB* are complex the operation creates the complex waves:

W_alphaValues, W_betaValues, M_leftEigenVectors, M_rightEigenVectors

with the ratio W_alphaValues[j]/W_betaValues[j] expressing the generalized eigenvalue. The corresponding M_leftEigenVectors[][j] and M_rightEigenVectors[][j] are the respective left and right eigenvectors. The simplicity of the complex case suggests that when *matrixA* and *matrixB* are real it is best to convert them to complex waves prior to executing matrixEigenV.

Depending on the choice of /S the operation also calculates the reciprocal condition number for the eigenvalues (stored in W_reciprocalConditionE) and the reciprocal condition number of the eigenvectors (stored in W_reciprocalConditionV). Note that a zero entry in W_reciprocalConditionV implies that the eigenvalues could not be ordered.

See Also

Matrix Math Operations on page III-138 for more about Igor's matrix routines and for background references with details about the LAPACK libraries.

Symmetric matrices can also be decomposed using the **MatrixSchur** operation and using **MatrixOp** chol.