Matrix Toolbox Manual

Chapter 2

Operator Descriptions

Chapter 2 - Operator Descriptions

A. Introduction

This chapter contains the man1 pages for all programs in the Matrix toolbox, organized alphabetically by object name. Any man1 page can be accessed by typing the following command.

```
% kman <program_name>
```

This information can also be accessed by selecting the *Help* button on the program's graphical user interface, which is displayed when running the program from within cantata, or when running it from the command line using the *-gui* option.

The keywords (-k) option in kman allows the user to provide a keyword string for searching. When invoked with the *-k* option, kman will perform the search and provide a list of the names and short descriptions of programs found by the search.

```
% kman -k "keyword string"
```

If you wish to display only the command line usage information for a program, enter the program name, followed by -U.

```
% <program_name> -U
```

B. Matrix Operators

B.1. madd — *Output = Matrix 1 + Matrix 2*

Object Information

Category:	*Matrix*	Subcategory:	*Arithmetic*
Operator:	*Matrix Add*	Object Type:	*pane*
In Cantata:	*Yes*		

Description

madd adds two matrices in a component by component basis. *madd* is really a pane object that uses the *karith2* operator in the DATAMANIP toolbox to actually perform the operation, so the man page for *karith2* should be consulted for additional details regarding the treatment of maps, masks, etc.

See Also

DATAMANIP:karith2(1)

Command Line Arguments

```
Usage for madd: Output = Matrix 1 + Matrix 2
```

```
% madd
  -i1        (infile)  Input matrix 1
  -i2        (infile)  Input Matrix 2
  -o         (outfile) output matrix
```

B.2. mconj — *Compute the Complex Conjugate Matrix*

Object Information

Category:	*Matrix*	Subcategory:	*Arithmetic*
Operator:	*Matrix Conjugate*	Object Type:	*pane*
In Cantata:	*Yes*		

Description

mconj is used to compute an output matrix where each component is the complex conjugate of the corresponding component in the input matrix. *mconj* is really a pane object that uses the *kcmplx* operator in the DATAMANIP toolbox to actually perform the calculations. See the man page for the *kcmplx* operator for details on the handling of masks, maps, etc.

See Also

DATAMANIP:kcmplx(1)

Command Line Arguments

```
Usage for mconj: Compute the Complex Conjugate Matrix
% mconj
  -i         (infile)  Input data object
  -o         (outfile) Resulting output data object
```

B.3. mcovar — *Compute Covariance Matrix*

Object Information

Category:	*Matrix*	Subcategory:	*Linear Operators*
Operator:	*Covariance Matrix*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

mcovar computes the covariance matrix of the set of data vectors supplied in the input data object. All W*H*D*T vectors are used collectively, forming an E*E covariance matrix.

The output will of of type DOUBLE if the input data is real, or of type DCOMPLEX if the input data is any complex data type.

Restrictions

A second version of mcovar should be written that uses the lmcovar() routine to produce a separate covariance matrix for each or various selected primitive and orientations.

Command Line Arguments

```
Usage for mcovar: Compute Covariance Matrix
% mcovar
  -i         (infile)  Input data object
  -o         (outfile) output matrix object
```

B.4. meigen — *Compute Eigenvectors and Eigenvalues of a Square Matrix*

Object Information

Category:	*Matrix*	Subcategory:	*EigenValues/Vectors*
Operator:	*Eigen*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

meigen computes the eigenvalues and right eigenvectors of a square matrix. The input matrix can be nonsymmetric. The input matrix is caste to KDCOMPLEX for reading, and all output matrices are of type KDCOMPLEX.

The eigenvectors are stored as columns of the eigenvector output matrix. The eigenvectors are all normalized to have unit 2-norm and a purely real largest component.

The eigenvalues are stored as a diagonal matrix with the i'th diagonal being the eigenvalue corresponding the the eigenvector in column i of the eigenvector output matrix.

meigen uses the LAPACK routine ZGEEV to perform the calculations.

References

LAPACK Users' Guide, E. Anderson et. al., SIAM 1992. ISBN 0-89871-294-7

Command Line Arguments

```
Usage for meigen: Compute Eigenvectors and Eigenvalues of a Square Matrix
% meigen
  -i         (infile)  Input Matrix

  [-o1]      (outfile) Eigenvector output matrix
             (default = {none})
  [-o2]      (outfile) Eigenvalue output matrix
             (default = {none})
```

B.5. mexchg — *Exchange Rows or Columns of a Matrix*

Object Information

Category:	*Matrix*	Subcategory:	*Utilities*
Operator:	*Exchange row/col*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

mexchg is used to interchage a pair of columns and/or a pair of rows of a matrix. If only a row exchange is desired, set the column indices equal to each other. Likewise, if only a column exhange is desired, set the row indices equal to each other.

Command Line Arguments

```
Usage for mexchg: Exchange Rows or Columns of a Matrix
% mexchg
  -i         (infile)  Input matrix
  -o         (outfile) Output matrix


  [Optional Mutually Inclusive Group - Specify all or none of the 2 options:]
    [-r1] (integer) row #1
          (r1 >= 0, default = 1)
    [-r2] (integer) row #2
          (r2 >= 0, default = 1)

  [Optional Mutually Inclusive Group - Specify all or none of the 2 options:]
    [-c1] (integer) Column #1
          (c1 >= 0, default = 1)
    [-c2] (integer) Column #2
          (c2 >= 0, default = 1)
```

B.6. mextract_col — *Extract a Column from a Matrix*

Object Information

Category:	*Matrix*	Subcategory:	*Utilities*
Operator:	*Extract Col*	Object Type:	*pane*
In Cantata:	*Yes*		

Description

mextract_col is used to extract a specific column from a matrix into a column vector. *mextract_col* is a pane object pointing to *mextract*.

Command Line Arguments

```
Usage for mextract_col: Extract a Column from a Matrix
% mextract_col
  -i         (infile)  Input matrix
  -o         (outfile) Output column vector

  [-num]     (integer) Column number
             (num >= 0, default = 0)
```

B.7. mextract_diag — *Extract Matrix Diagonal, Row, or Column*

Object Information

Category:	*Matrix*	Subcategory:	*Utilities*
Operator:	*Extract diag/row/col*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

mextract_diag is used to extract the diagonal of a matrix into a row vector, or to extract a specific row or column from the matrix. The output will be either of type DOUBLE or DOUBLE COMPLEX according to the input object being real or complex respectively.

Command Line Arguments

```
Usage for mextract_diag: Extract Matrix Diagonal, Row, or Column
% mextract_diag
  -i         (infile)  Input matrix
  -o         (outfile) Output vector

  Required Mutually Exclusive Group - Specify exactly one of the 3 options:
    [-row]  (flag) Extract row
    [-col]  (flag) Extract column
    [-diag] (flag) Extract diagonal

  [-num]     (integer) Row or column number
             (num >= 0, default = 0)
```

B.8. mextract_row — *Extract a Row from a Matrix*

Object Information

Category:	*Matrix*	Subcategory:	*Utilities*
Operator:	*Extract Row*	Object Type:	*pane*
In Cantata:	*Yes*		

Description

mextract_row is used to extract a specific row from a matrix into a row vector. *mextract_row* is a pane object pointing to *mextract*.

Command Line Arguments

```
Usage for mextract_row: Extract a Row from a Matrix
% mextract_row
  -i         (infile)  Input matrix
  -o         (outfile) Output row vector

  [-num]     (integer) Row number
             (num >= 0, default = 0)
```

B.9. mgdiag — *Generate a Diagonal Matrix*

Object Information

Category:	*Matrix*	Subcategory:	*Generation*
Operator:	*Gen Diag Matrix*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

mgdiag is used to generate matrices with all elements zero except along the diagonal. There are several "modes" of behavior that can be obtained:

1. Constant diagonal values

   Here, the constant value is suppled by selecting and setting the "Constant" selections. The size of the matrix is controlled by selecting and setting the Rows and Columns as desired. If the matrix is not square, the diagonal will be carried as far as possible before hitting the boundary of the matrix. The matrix data type is controlled by the setting of the output data type list selector. If a complex data type is selected but the constant is underspecified (meaning that either the real or imagaring part was not specified) then the unspecified part is assumed to be zero. If a complex value is specified, but a real data type is selected for output, then the imaginary part is ignored.

2. Non-constant diagonal values

   Here, the diagonal values are read from the given vector of values. If the matrix size is not specified explicitly, then it is set to a square matrix with dimensions equal to the length of the data vector. If the matrix size is explicitly specified, then the vector element values are inserted into the diagonal matrix until either the matrix boundary is hit or there are no more values in the vector (in this case the remaining diagonal values will be zero). If the vector is complex but the output type is real then the imaginary part is ignored. If the vector is real but the output type is complex, then the imaginary part will be assumed to be zero.

In all cases, the output data type is ultimately controlled by the Output Data Type selector list setting, and the matrix size by the Row and Column selections if set.

The generated matrix will contain a single plane of data, i.e. DxTxE=1.

See Also

mextract_diag(1)

Restrictions

This is a messy interface and could be improved, but it is also sufficiently general to let you make whatever kind of mess you want!

Command Line Arguments

```
Usage for mgdiag: Generate a Diagonal Matrix
% mgdiag
  -type       (cycle)   Output data type
          1  "Float",
          2  "Complex",
          3  "Double",
          4  "Double Complex "

  -o          (outfile) Resulting output data object

  [-cr]       (double)  Real part of constant diagonal value
              (unbounded, default = 0)
  [-ci]       (double)  Imaginary part of constant diagonal value
              (unbounded, default = 0)
  [-i]        (infile)  Vector of diagonal elements
              (default = {none})

  [Optional Mutually Inclusive Group - Specify all or none of the 2 options:]
    [-r] (integer) Number of rows
         (r > 0, default = 1)
    [-c] (integer) Number of columns
         (c > 0, default = 1)
```

B.10. mgident — *Generate Identity or Unit Matrix*

Object Information

Category:	*Matrix*	Subcategory:	*Generation*
Operator:	*Identity Matrix*	Object Type:	*pane*
In Cantata:	*Yes*		

Description

mgident is used to generate a matrix with all diagonal elements equal to one (unity). The output is of type DOUBLE.

mgident is a pane object pointing to *mgdiag* in the MATRIX toolbox.

See Also

mgdiag(1)

Command Line Arguments

```
Usage for mgident: Generate Identity or Unit Matrix
% mgident
  -o         (outfile) Output matrix

  [-r]       (integer) rows
             (r >= 0, default = 0)
  [-c]       (integer) cols
             (c >= 0, default = 0)
```

B.11. mgnull — *Generate Null or Zero Matrix*

Object Information

Category:	*Matrix*	Subcategory:	*Generation*
Operator:	*Null Matrix*	Object Type:	*pane*
In Cantata:	*Yes*		

Description

mgnull is used to generate a matrix with all elements equal to zero (null). The output is of type DOUBLE.

The optional trigger input is useful in cantata to cause a new output matrix to be generated.

mgnull is a pane object pointing to *kgconst* in the DATAMANIP toolbox.

Command Line Arguments

```
Usage for mgnull: Generate Null or Zero Matrix
% mgnull
  -o         (outfile) Output matrix

  [-hsize]   (integer) Height (rows)
             (hsize >= 0, default = 0)
  [-wsize]   (integer) Width (cols)
             (wsize >= 0, default = 0)
```

B.12. minvdiag — *Compute Inverse of Diagonal Matrix*

Object Information

Category:	*Matrix*	Subcategory:	*Arithmetic*

Operator:	*Invert Diagonal*	Object Type:	*pane*
In Cantata:	*Yes*		

Description

minvdiag computes the inverse of a diagonal matrix. The matrix can be non-square. Output is of type KDCOMPLEX if the input matrix is a complex type, otherwise the output is of type KDOUBLE.

No checking is done for a divide by zero condition.

Note that only the diagonal elements of the input matrix are modified.

minvdiag is a pane object pointing to *minvert*.

Command Line Arguments

```
Usage for minvdiag: Compute Inverse of Diagonal Matrix
% minvdiag
  -i         (infile)  Input matrix
  -o         (outfile) Output matrix
```

B.13. minvert — *Compute Inverse Matrix*

Object Information

Category:	*Matrix*	Subcategory:	*Arithmetic*
Operator:	*Invert Matrix*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

minvert computes the inverse of a matrix. If the -diag flag *is* selected, then only the diagonal elements are inverted (this will work for any size of matrix). If the -diag flag is *not* selected, then the matrix must be square, and the inverse will be computed by backsolving an LU decomposition.

minvert uses the LAPACK routines DGETRF, DGETRI, ZGERF, and ZGTRI to actually perform the decomposition and compute the inverse.

The output data type will be KDCOMPLEX if the input matrix is either COMPLEX or KDCOMPLEX. Otherwise the output will be KDOUBLE.

After execution of the appropriate LAPACK routines, a diagnostic may be printed out indicating that the input matrix is singular and at what point this was detected. This will be printed only if KHOROS_NOTIFY is set to STANDARD or more verbosity. This message indicates a fatal error.

Restrictions

minvert does **not** check for a divide by zero condition when computing the inverse with the -diag flag selected.

References

LAPACK Users' Guide, E. Anderson et. al., SIAM 1992. ISBN 0-89871-294-7

Command Line Arguments

```
Usage for minvert: Compute Inverse Matrix
% minvert
  -i         (infile)  Input matrix
  -o         (outfile) Output matrix

  [-diag]    (flag)    Invert diagonal only
```

B.14. mlse — *Least Squares Solution to System of Linear Equations*

Object Information

Category:	*Matrix*	Subcategory:	*Linear Operators*
Operator:	*Simultaneous Eqns*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

mlse solves a system of linear simultaneous equations by either the QR or LQ decomposition. The input consists of a real or complex matrix A and a real or complex column vector b. If A is of size M rows by N columns, then for M>N *mlse* finds the least squares solution to the overdetermined system

$$\underset{x}{\text{minimize}} \| b - A*x \|_2$$

and for M<N *mlse* finds the minimum norm solution to the underdetermined system

$$A*x = b.$$

It is assumed that A has full rank, i.e. rank(A) = min(M,N). If A does not have full rank, then *mlse* will fail. In this case you may wish to use *msvd* to compute the solution.

For the overdetermined case, the QR decomposition is used, with the LQ decomposition used for the underdetermined situation.

Multiple right hand sides (b vectors) can be supplied as columns of a matrix B, and a solution will be computed for each b vector. These solutions will be stored in the columns of a solution matrix X. Note that the solution for each b vector is obtained independently; this is *not* the same as solving the system min || B - AX ||.

If either input (A matrix or b vector) is complex then the output will be of type KDCOMPLEX. Otherwise, the output will be of type KDOUBLE.

Command Line Arguments

```
Usage for mlse: Least Squares Solution to System of Linear Equations
% mlse
  -i1        (infile)  A matrix
  -i2        (infile)  b vector (or B matrix)
  -o         (outfile) solution vector (or matrix)
```

B.15. mlud — *Compute LU Decomposition of Matrix (Input = L*U)*

Object Information

Category:	*Matrix*	Subcategory:	*Decomposition*
Operator:	*LU Decomposition*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

mlud computes the LU decomposition of a matrix with partial pivoting and row interchanges. If A is the input matrix, *mlud* will solve the system P*L*U = A for the L and U matrices. The P matrix is represented on output by a pivot index vector PI, where row i has been interchanged with row PI[i]. U is an upper triangular matrix, and L lower triangular with unit diagonal. PI is of length min(rows,cols). The LAPACK routines DGETRF and ZGETRF are used to perform the actual decomposition.

After execution of the appropriate LAPACK routine, a diagnostic may be printed out indicating that the input matrix is singular and at what point this was detected. This will be printed only if KHOROS_NOTIFY is set to STANDARD or more verbosity. This message is NOT an error, just information, and *mlud* will run to completion anyway.

References

LAPACK Users' Guide, E. Anderson et. al., SIAM 1992. ISBN 0-89871-294-7

Command Line Arguments

```
Usage for mlud: Compute LU Decomposition of Matrix  (Input = L*U)
% mlud
  -i         (infile)  Input matrix

  [-o1]      (outfile) Resulting lower triangular output matrix
             (default = {none})
  [-o2]      (outfile) Resulting upper triangular output  matrix
             (default = {none})
  [-o3]      (outfile) Pivot index vector
             (default = {none})
```

B.16. mmul — *Matrix Multiply (C=AB)*

Object Information

Category:	*Matrix*	Subcategory:	*Arithmetic*
Operator:	*Matrix Multiply*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

mmul forms the matrix product C=AB. The number of columns in A must match the number of rows in C.

Output will be either double or double complex, depending on the type of the input matrices. If either A or B is a complex type, the output will be double complex. Otherwise the output will be double.

Processing is done by independent planes in WxH.

References

See any book on linear algebra.

Command Line Arguments

```
Usage for mmul: Matrix Multiply (C=AB)
% mmul
  -i1        (infile)  Input matrix #1 (A)
  -i2        (infile)  Input matrix #2 (B)
  -o         (outfile) Output matrix (C)
```

B.17. mpow — *Compute the Positive Integral Power of a Square Matrix*

Object Information

Category:	*Matrix*	Subcategory:	*Arithmetic*
Operator:	*Matrix Power*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

mpow raises a square matrix to an integer, non-negative power. The output will be DCOMPLEX if the input matrix is either COMPLEX or DCOMPLEX. Otherwise the output type will be DOUBLE.

Command Line Arguments

```
Usage for mpow: Compute the Positive Integral Power of a Square Matrix
```

```
% mpow
  -i         (infile)  Input matrix object
  -pow       (integer) power to which the matrix should be raised (>=0)
              pow >= 0
  -o         (outfile) Output matrix object
```

B.18. mrcsum — *Compute Row or Column Sums for a Matrix*

Object Information

Category:	*Matrix*	Subcategory:	*Utilities*
Operator:	*Row/Col Sums*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

mrcsum computes row and column sum vectors for a matrix object. The output objects will be either of type DOUBLE or DOUBLE COMPLEX according to the input object being real or complex respectively.

The output vectors can be scaled if desired by N or N-1 where N is the number of colums (for a row sum) or the number of rows (for a column sum). This feature facilitates the computation of a mean vector, and other statistical measures.

Command Line Arguments

```
Usage for mrcsum: Compute Row or Column Sums for a Matrix
% mrcsum
  -i         (infile)  Input matrix object

  [-scale]   (flag)    scale by 1, 1/N or 1/(N-1)
          1  (Do not scale the output data),
          2  (Scale output data by 1/N),
          or 3  (Scale output data by 1/(N-1))
[default: 1]
  [-or]      (outfile) Row sum output object
             (default = {none})
  [-oc]      (outfile) Column sum output object
             (default = {none})
```

B.19. mreplicate — *Replicate a Given Matrix as Submatrices of a Larger Matrix*

Object Information

Category:	*Matrix*	Subcategory:	*Utilities*
Operator:	*Replicate Submatrix*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

mreplicate replicates the input matrix as a rectangular array of submatrices to form a larger matrix. The input maybe of any dimension NxM as long as N>1 and M>1. The output matrix will be (R*N)x(C*M) where R is the number of times the input matrix is to be replicated along the row dimension and C is the number of times the input matrix is to be replicated along the column dimension.

The output data type is the same as the input data type.

Examples

% mreplicate -i image:moon -o blah -r 2 -c 3

Produces an image of the moon replicated 3 times in the horizontal direction and 2 times in the vertical direction.

Command Line Arguments

```
Usage for mreplicate: Replicate a Given Matrix as Submatrices of a Larger Matrix
% mreplicate
  -i         (infile)  Input mmatrix data object
  -o         (outfile) Resulting matrix output data object

  [-r]       (integer) replica count along rows
             (r > 0, default = 1)
  [-c]       (integer) replica count along columns
             (c > 0, default = 1)
```

B.20. mscalar_abs — *Output = Absolute Value of Input (componentwise)*

Object Information

Category:	*Matrix*	Subcategory:	*Arithmetic*
Operator:	*Scalar ABS*	Object Type:	*pane*
In Cantata:	*Yes*		

Description

mscalar_abs computes the absolute value of the input matrix on a component by component basis. *mscalar_abs* is really a pane object that uses the *karith1* operator in the DATAMANIP toolbox to actually perform the operation, so the man page for *karith1* should be consulted for additional details regarding the treatment of maps, masks, etc.

See Also

DATAMANIP:karith1(1)

Command Line Arguments

```
Usage for mscalar_abs: Output = Absolute Value of Input (componentwise)
% mscalar_abs
  -i         (infile)  Input matrix
  -o         (outfile) Output matrix
```

B.21. mscalar_add — *Output = Input Matrix + Constant*

Object Information

Category:	*Matrix*	Subcategory:	*Arithmetic*
Operator:	*Scalar Add*	Object Type:	*pane*
In Cantata:	*Yes*		

Description

mscalar_add adds a specified constant to each component in a matrix. *mscalar_add* is really a pane object that uses the *kadd* operator in the DATAMANIP toolbox to actually perform the operation, so the man page for *kadd* should be consulted for additional details regarding the treatment of maps, masks, etc.

See Also

DATAMANIP:karith2(1)

Command Line Arguments

```
Usage for mscalar_add: Output = Input Matrix + Constant
% mscalar_add
  -i1        (infile)  Input matrix
  -o         (outfile) output matrix

  [-real]    (double)  Real constant value
             (unbounded, default = 0)
  [-image]   (double)  Imaginary constant value
             (unbounded, default = 0)
```

B.22. mscalar_div — *Output = Input Matrix / Constant*

Object Information

Category:	*Matrix*	Subcategory:	*Arithmetic*
Operator:	*Scalar Divide*	Object Type:	*pane*
In Cantata:	*Yes*		

Description

mscalar_div divides each component in a matrix by a specified constant. *mscalar_div* is really a pane object that uses the *kdiv* operator in the DATAMANIP toolbox to actually perform the operation, so the man page for *kdiv* should be consulted for additional details regarding the treatment of maps, masks, etc.

See Also

DATAMANIP:karith2(1)

Command Line Arguments

```
Usage for mscalar_div: Output = Input Matrix / Constant
% mscalar_div
  -i1         (infile)  Input matrix 1
  -o          (outfile) output matrix

  [-real]     (double)  Real constant value
              (unbounded, default = 1)
  [-imag]     (double)  Imaginary constant value
              (unbounded, default = 1)
```

B.23. mscalar_mul — *Output = Input Matrix * Constant*

Object Information

Category:	*Matrix*	Subcategory:	*Arithmetic*
Operator:	*Scalar Multiply*	Object Type:	*pane*
In Cantata:	*Yes*		

Description

mscalar_mul multiplies each component in a matrix by a specified constant. *mscalar_mul* is really a pane object that uses the *kmul* operator in the DATAMANIP toolbox to actually perform the operation, so the man page for *kmul* should be consulted for additional details regarding the treatment of maps, masks, etc.

See Also

DATAMANIP:karith2(1)

Command Line Arguments

```
Usage for mscalar_mul: Output = Input Matrix * Constant
% mscalar_mul
  -i1         (infile)  Input matrix 1
  -o          (outfile) output matrix

  [-real]     (double)  Real constant value
              (unbounded, default = 1)
  [-imag]     (double)  Imaginary constant value
              (unbounded, default = 1)
```

B.24. mscalar_pow — *Output = Input Raised to a Power (componentwise)*

Object Information

Category:	*Matrix*	Subcategory:	*Arithmetic*
Operator:	*Scalar Power*	Object Type:	*pane*
In Cantata:	*Yes*		

Description

mscalar_pow raises each component in a matrix to a specified constant power. *mscalar_pow* is really a pane object that uses the *karith2* operator in the DATAMANIP toolbox to actually perform the operation, so the man page for *karith2* should be consulted for additional details regarding the treatment of maps, masks, etc.

See Also

DATAMANIP:karith2(1)

Command Line Arguments

```
Usage for mscalar_pow: Output = Input Raised to a Power (componentwise)
% mscalar_pow
  -i1        (infile)  Input matrix 1
  -o         (outfile) output matrix

  [-real]    (double)  Real constant value
             (unbounded, default = 0)
  [-imag]    (double)  Imaginary constant value
             (unbounded, default = 0)
```

B.25. mscalar_recip — *Output = Reciprocal of Input (componentwise)*

Object Information

Category:	*Matrix*	Subcategory:	*Arithmetic*
Operator:	*Scalar Recip*	Object Type:	*pane*
In Cantata:	*Yes*		

Description

mscalar_recip computes the reciprocal of the input matrix on a component by component basis. This is not the same as a *matrix* reciprocal (inverse). *mscalar_recip* is really a pane object that uses the *karith1* operator in the DATAMANIP toolbox to actually perform the operation, so the man page for *karith1* should be consulted for additional details regarding the treatment of maps, masks, etc.

See Also

DATAMANIP:karith1(1)

Command Line Arguments

```
Usage for mscalar_recip: Output = Reciprocal of Input (componentwise)
% mscalar_recip
  -i         (infile)  Input matrix
  -o         (outfile) Output matrix
```

B.26. mscalar_sqrt — *Output = Sqrt of Input (componentwise)*

Object Information

Category:	*Matrix*	Subcategory:	*Arithmetic*
Operator:	*Scalar Sqrt*	Object Type:	*pane*
In Cantata:	*Yes*		

Description

mscalar_sqrt computes the square root of the input matrix on a component by component basis. This is not the same as a *matrix* square root. *mscalar_sqrt* is really a pane object that uses the *karith1* operator in the DATAMANIP toolbox to actually perform the operation, so the man page for *karith1* should be consulted for additional details regarding the treatment of maps, masks, etc.

See Also

DATAMANIP:karith1(1)

Command Line Arguments

```
Usage for mscalar_sqrt: Output = Sqrt of Input (componentwise)
% mscalar_sqrt
  -i         (infile)  Input matrix
  -o         (outfile) Output matrix
```

B.27. mscalar_sub — *Output = Input Matrix - Constant*

Object Information

Category:	*Matrix*	Subcategory:	*Arithmetic*
Operator:	*Scalar Subtract*	Object Type:	*pane*
In Cantata:	*Yes*		

Description

mscalar_sub subtracts a specified constant from each component in a matrix. *mscalar_sub* is really a pane object that uses the *ksub* operator in the DATAMANIP toolbox to actually perform the operation, so the man page for *ksub* should be consulted for additional details regarding the treatment of maps, masks, etc.

See Also

DATAMANIP:karith2(1)

Command Line Arguments

```
Usage for mscalar_sub: Output = Input Matrix - Constant
% mscalar_sub
  -i1        (infile)  Input matrix
  -o         (outfile) output matrix

  [-real]    (double)  Real constant value
             (unbounded, default = 0)
  [-imag]    (double)  Imaginary constant value
             (unbounded, default = 0)
```

B.28. msub — *Output = Matrix 1 - Matrix 2*

Object Information

Category:	*Matrix*	Subcategory:	*Arithmetic*
Operator:	*Matrix Subtract*	Object Type:	*pane*
In Cantata:	*Yes*		

Description

msub subtracts one matrix from another on a component by component basis. *msub* is really a pane object that uses the *karith2* operator in the DATAMANIP toolbox to actually perform the operation, so the man page for *karith2* should be consulted for additional details regarding the treatment of maps, masks, etc.

See Also

DATAMANIP:karith2(1)

Command Line Arguments

```
Usage for msub: Output = Matrix 1 - Matrix 2
% msub
  -i1        (infile)  Input matrix 1
  -i2        (infile)  Input Matrix 2
  -o         (outfile) Output file
```

B.29. msvd — *Compute Singular Value Decomposition (SVD)*

Object Information

Category:	*Matrix*	Subcategory:	*Decomposition*
Operator:	*SVD*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

msvd computes the Singular Value Decomposition of a matrix.

The output matrices will be of type DOUBLE unless the input matrix was a complex type. In that case the output will be of type KDCOMPLEX. In all cases, the S matrix is always of type KDOUBLE since singular values are always positive and always real.

The singular value decomposition produces a diagonal matrix S (available via -o2) of the same dimensions as the input matrix. and with nonnegative diagonal elements in decreasing order. Also produced are unitary matrices U (available via -o1) and V (available via -o3) such that A=U*S*V', where A is the input matrix and V' means the transpose of V. The columns of U and V are the left and right singular vectors, respectively. The singular vectors are orthonormal to each other within both U and V.

For a real, symmetric, square input matrix, U=V and the singular vectors are the eigenvectors. The eigenvalues are the square root of the singular values.

References

LAPACK Users' Guide, E. Anderson et. al., SIAM 1992. ISBN 0-89871-294-7

Command Line Arguments

```
Usage for msvd: Compute Singular Value Decomposition (SVD)
% msvd
  -i        (infile)  Input matrix
  -o1       (outfile) Output matrix U
  -o2       (outfile) Output matrix S
  -o3       (outfile) Output matrix V
```

B.30. mtranspose — *Transpose Rows and Columns of Matrix*

Object Information

Category:	*Matrix*	Subcategory:	*Arithmetic*
Operator:	*Transpose Matrix*	Object Type:	*pane*
In Cantata:	*Yes*		

Description

mtranspose transposes the rows and columns of the input matrix. *mtranspose* is really a pane object for the *kaxis* operator in the DATAMANIP toolbox. See the man page for *kaxis* for details in the handling of masks, maps, etc.

See Also

DATAMANIP:kaxis(1)

Command Line Arguments

```
Usage for mtranspose: Transpose Rows and Columns of Matrix
% mtranspose
  -i         (infile)  Input data object
  -o         (outfile) Resulting output data object
```

This page left intentionally blank

Table of Contents