- Main Page
- Data Structures
- Files

- File List
- Globals

# matrix.h File Reference

ARToolkit algebric mathematics subroutines. More...

```
#include <math.h>
#include <AR/config.h>
```

## Data Structures

| | |
|---|---|
| struct | **ARMat**<br>*matrix structure. More...* |
| struct | **ARVec**<br>*vector structure. More...* |

## Defines

| | |
|---|---|
| #define | **ARELEM0**(mat, r, c)   ((mat)->m[(r)*((mat)->clm)+(c)])<br>*macro function that give direct access to an element (0 origin)* |
| #define | **ARELEM1**(mat, row, clm)   ARELEM0(mat,row-1,clm-1)<br>*macro function that give direct access to an element (1 origin)* |

## Functions

| | |
|---|---|
| **ARMat** * | **arMatrixAlloc** (int row, int clm)<br>*creates a new matrix.* |
| int | **arMatrixFree** (**ARMat** *m)<br>*deletes a matrix.* |
| int | **arMatrixDup** (**ARMat** *dest, **ARMat** *source)<br>*copy a matrix* |
| **ARMat** * | **arMatrixAllocDup** (**ARMat** *source)<br>*dumps a new matrix* |
| int | **arMatrixUnit** (**ARMat** *unit)<br>*Creates a unit matrix.* |
| **ARMat** * | **arMatrixAllocUnit** (int dim)<br>*Creates a unit matrix.* |
| int | **arMatrixMul** (**ARMat** *dest, **ARMat** *a, **ARMat** *b)<br>*Multiply two matrix.* |
| **ARMat** * | **arMatrixAllocMul** (**ARMat** *a, **ARMat** *b)<br>*Multiply two matrix with memory allocation.* |
| int | **arMatrixTrans** (**ARMat** *dest, **ARMat** *source)<br>*transposes a matrix.* |
| **ARMat** * | **arMatrixAllocTrans** (**ARMat** *source)<br>*transposes a matrix with allocation.* |
| int | **arMatrixInv** (**ARMat** *dest, **ARMat** *source)<br>*inverse a matrix.* |
| int | **arMatrixSelfInv** (**ARMat** *m)<br>*inverses a matrix.* |
| **ARMat** * | **arMatrixAllocInv** (**ARMat** *source)<br>*inverses a matrix.* |
| double | **arMatrixDet** (**ARMat** *m)<br>*compute determinant of a matrix.* |
| int | **arMatrixPCA** (**ARMat** *input, **ARMat** *evec, **ARVec** *ev, **ARVec** *mean)<br>*compute the PCA of a matrix.* |
| int | **arMatrixPCA2** (**ARMat** *input, **ARMat** *evec, **ARVec** *ev)<br>*compute the PCA of a matrix.* |
| int | **arMatrixDisp** (**ARMat** *m)<br>*display content of a matrix.* |
| **ARVec** * | **arVecAlloc** (int clm)<br>*creates a new vector.* |
| int | **arVecFree** (**ARVec** *v)<br>*delete a vector.* |

| | |
|---|---|
| int | **arVecDisp** (**ARVec** *v)<br>*display a vector.* |
| double | **arVecHousehold** (**ARVec** *x)<br>*XXXBK.* |
| double | **arVecInnerproduct** (**ARVec** *x, **ARVec** *y)<br>*Computes the inner product of 2 vectors.* |
| int | **arVecTridiagonalize** (**ARMat** *a, **ARVec** *d, **ARVec** *e)<br>*XXXBK.* |

## Detailed Description

ARToolkit algebric mathematics subroutines.

This package include matrix, vector manipulation routine. In complement to must classical routines (inversion, innerproduct), it includes a PCA (Principal) Component Analysis) routine. For the structure of the matrix see **ARMat**.

**Remarks:**

History :

**Author:**
Hirokazu Kato kato@sys.im.hiroshima-cu.ac.jp

**Version:**

**Date:**

## Function Documentation

```
ARMat * arMatrixAlloc( int row,
                       int clm
                     )
```

creates a new matrix.

Allocate and initialize a new matrix structure. XXXBK initializing ?? to 0 m ??

**Parameters:**

| | |
|---|---|
| *row* | number of line |
| *clm* | number of column |

**Returns:**
the matrix structure, NULL if allocation is impossible

```
ARMat * arMatrixAllocDup( ARMat * source )
```

dumps a new matrix

Allocates and recopy the original source matrix.

**Parameters:**

| | |
|---|---|
| *source* | the source matrix to copy |

**Returns:**
the matrix if success, NULL if error

```
int arMatrixAllocInv( ARMat * source )
```

inverses a matrix.

Inverses a matrix and copy the result in in a new allocated structure.

**Parameters:**

| | |
|---|---|
| *source* | the matrix to inverse |

**Returns:**
the inversed matrix if success, NULL if error

```
ARMat * arMatrixAllocMul( ARMat * a,
                          ARMat * b
```

**)**

Multiply two matrix with memory allocation.

multiply two matrix and copy the result in a new allocate matrix (the source matrix is unmodified). the product is this one : dest = a * b

**Parameters:**

*a* first matrix
*b* second matrix

**Returns:**

the allocated matrix if success, NULL if error

**ARMat * arMatrixAllocTrans( ARMat * *source* )**

transposes a matrix with allocation.

transposes a matrix and copy the result in a new allocate matrix (the source matrix is unmodified).

**Parameters:**

*source* the matrix to transpose

**Returns:**

the allocated matrix if success, NULL if error (creation or transposition impossible)

**int arMatrixAllocUnit( int *dim* )**

Creates a unit matrix.

Allocates and initializes a matrix to a an identity matrix.

**Parameters:**

*dim* dimensions of the unit matrix (square)

**Returns:**

the matrix allocated if success, NULL if error

**int arMatrixDet( ARMat * *m* )**

compute determinant of a matrix.

Compute the determinant of a matrix.

**Parameters:**

*m* matrix source

**Returns:**

the computed determinant

**int arMatrixDisp( ARMat * *m* )**

display content of a matrix.

Display in current console, the content of the matrix. The display is done line by line.

**Parameters:**

*m*

**Returns:**

0

**int arMatrixDup( ARMat * *dest*,**
**ARMat * *source***
**)**

copy a matrix

copy one matrix to another. The two **ARMat** must be allocated.

**Parameters:**

| | |
|---|---|
| *dest* | the destination matrix of the copy |
| *source* | the original matrix source |

**Returns:**
0 if success, -1 if error (matrix with different size)

```
int arMatrixFree( ARMat * m )
```

deletes a matrix.

Delete a matrix structure (deallocate used memory).

**Parameters:**

| | |
|---|---|
| *m* | matrix to delete |

**Returns:**
0

```
int arMatrixInv( ARMat * dest,
                 ARMat * source
               )
```

inverse a matrix.

inverse a matrix and copy the result in a new one (the source matrix is unmodified). the destination matrix must be allocated. the source matrix need to be a square matrix.

**Parameters:**

| | |
|---|---|
| *dest* | result matrix of the inverse operation |
| *source* | source matrix |

**Returns:**
0 if success, -1 if error (not square matrix)

```
int arMatrixMul( ARMat * dest,
                 ARMat * a,
                 ARMat * b
               )
```

Multiply two matrix.

Multiply two matrix and copy the result in another the product is this one : dest = a * b. The destination matrix must be allocated. Matrix a and b need to have the same size (the source matrix is unmodified).

**Parameters:**

| | |
|---|---|
| *dest* | final matrix product |
| *a* | first matrix |
| *b* | second matrix |

**Returns:**
0 if success, -1 if error (multiplication impossible, or destination matrix have not comptabile size)

```
int arMatrixPCA( ARMat * input,
                 ARMat * evec,
                 ARVec * ev,
                 ARVec * mean
               )
```

compute the PCA of a matrix.

Compute the Principal Component Analysis (PCA) of a matrix.

**Parameters:**

| | |
|---|---|
| *input* | source matrix |
| *evec* | eigen vector computed |
| *ev* | eigen value computed |
| *mean* | mean computed |

**Returns:**

0 if success to compute, -1 otherwise

```
int arMatrixPCA2( ARMat * input,
                  ARMat * evec,
                  ARVec * ev
                )
```

compute the PCA of a matrix.

Compute the Principal Component Analysis (PCA) of a matrix.

**Parameters:**

*input* source matrix
*evec* result matrix
*ev* egein value computed

**Returns:**

0 if success to compute, -1 otherwise

```
int arMatrixSelfInv( ARMat * m )
```

inverses a matrix.

Inverses a matrix and copy the result in the same structure.

**Parameters:**

*m* the matrix to inverse

**Returns:**

0 if success, -1 if error

```
int arMatrixTrans( ARMat * dest,
                   ARMat * source
                 )
```

transposes a matrix.

Transposes a matrix. The destination matrix must be allocated (the source matrix is unmodified).

**Parameters:**

*dest* the destination matrix of the copy
*source* the source matrix

**Returns:**

0 if success, -1 if error (source and destination matrix have different size)

```
int arMatrixUnit( ARMat * unit )
```

Creates a unit matrix.

Transforms the source parameter matrix to a unit matrix (all values are modified). the unit matrix needs to be allocated.

**Parameters:**

*unit* the matrix to transform

**Returns:**

0 if success, -1 if error

```
ARVec * arVecAlloc( int clm )
```

creates a new vector.

Allocates and initializes new vector structure.

**Parameters:**

*clm* dimension of vector

**Returns:**

the allocated vector, NULL if error (impossible allocation)

```
int arVecDisp( ARVec * v )
```

display a vector.

Display element of a vector.

**Parameters:**

*v* the vector to display

**Returns:**

0

```
int arVecFree( ARVec * v )
```

delete a vector.

Delete a vector structure (deallocate used memory).

**Parameters:**

*v* the vector to delete

**Returns:**

0

```
double arVecHousehold( ARVec * x )
```

XXXBK.

XXXBK: for QR decomposition ?? (can't success to find french translation of this term)

**Parameters:**

*x* XXXBK

**Returns:**

XXXBK

```
double arVecInnerproduct( ARVec * x,
                          ARVec * y
                        )
```

Computes the inner product of 2 vectors.

computes the inner product of the two argument vectors. the operation done is a=x.y (and a is return)

**Parameters:**

*x* first vector source

*y* second vector source

**Returns:**

the computed innerproduct

```
int arVecTridiagonalize( ARMat * a,
                         ARVec * d,
                         ARVec * e
                       )
```

XXXBK.

XXXBK

**Parameters:**

*a* XXXBK

*d* XXXBK

*e* XXXBK

**Returns:**

XXXBK

Generated with Doxygen