

行列計算ユーティリティー

(Ver.1.4)

2020 年 8 月

株式会社 アイディール

目次

eyemMatMallocVector	1
機能 ベクトル(n次元配列)の領域確保	1
eyemMatMallocMatrix	2
機能 行列(2次元配列)の領域確保	2
eyemMatFreeVector	3
機能 ベクトルの領域解放	3
eyemMatFreeMatrix	4
機能 行列の領域解放	4
eyemMatZero	5
機能 行列のゼロクリア	5
eyemMatDiagonal	6
機能 対角行列(単位行列のスカラー倍)の設定	6
eyemMatCopy	7
機能 行列のコピー	7
eyemMatAdd	8
機能 行列の加算	8
eyemMatSub	9
機能 行列の減算	9
eyemMatMulScalar	10
機能 行列のスカラー倍	10
eyemMatMul	11
機能 行列の乗算	11
eyemMatGenMul	12
機能 汎用的な行列の乗算	12
eyemMatMulTransposed	13
機能 行列とその転置行列との乗算	13
eyemMatMulTransposed2	14
機能 行列と別の行列の転置行列との乗算	14
eyemMatTransposed	15
機能 行列の転置	15
eyemMatTrace	16
機能 行列のトレース	16
eyemMatDet	17
機能 行列の行列式	17
eyemMatInvert	18
機能 行列の逆行列	18
eyemMatSolveLU	19

機能	連立 1 次方程式の解 (LU 分解による)	19
	eyemMatSolveSVD	20
機能	連立 1 次方程式の解 (特異値分解 (SVD) による), または最小二乗解	20
	eyemMatJacobiEigen	21
機能	実対称行列の固有値と固有ベクトル (Jacobi 法)	21
	eyemMatQrEigen	22
機能	実対称行列の固有値と固有ベクトル (QR 法)	22
	eyemMatSVD	23
機能	行列の特異値分解 (SVD)	23
	eyemMatCholeskyDecomp	24
機能	実対称正定値行列のコレスキー分解	24

eyemMatMallocVector

機 能	ベクトル (n次元配列) の領域確保
形 式	<pre>#include "eyemLib.h" double *eyemMatMallocVector (int n);</pre>
解 説	指定された次数のベクトル (配列) の領域を確保します.
引 数	n ベクトル (配列) の次数です.
戻り値	確保した領域の先頭アドレスです. ただし、領域不足の場合は NULL が返ります.
留意事項	本関数を使用した後は、必ず <code>eyemMatFreeVector()</code> を実行して領域を解放してください.

eyemMatMallocMatrix

機 能	行列(2次元配列)の領域確保		
形 式	#include “eyemLib.h” double **eyemMatMallocMatrix (int iRow, int iCol);		
解 説	指定された行数および列数の行列(2次元配列)の領域を確保します.		
引 数	iRow	行列の行数です.	
	iCol	行列の列数です.	
戻り値	確保した領域の先頭アドレスです. ただし、領域不足の場合は NULL が返ります.		
留意事項	(1) 本関数を使用した後は、必ず <code>eyemMatFreeMatrix()</code> を実行して領域を解放してください. (2) 例えば、本関数にて <div>double **mat = <code>eyemMatMallocMatrix</code>(m, n);</div> として <code>mat</code> を確保した場合、これを他の <code>eyemMat</code> シリーズの関数への引数として渡す場合には、必ず <code>*mat</code> または <code>mat[0]</code> を渡してください.		

eyemMatFreeVector

機 能	ベクトルの領域解放
形 式	<pre>#include "eyemLib.h" void eyedMatFreeVector (double *dpVec);</pre>
解 説	eyemMatMallocVecto()で確保したベクトルの領域を解放します.
引 数	*dpVec ベクトルの先頭アドレスです.
戻り値	ありません.
留意事項	特にありません.

eyemMatFreeMatrix

機 能	行列の領域解放
形 式	<pre>#include "eyemLib.h" void eyedMatFreeMatrix (double **dpMat);</pre>
解 説	eyemMatMallocMatrix() で確保した行列の領域を解放します.
引 数	**dpMat 行列の先頭アドレスです.
戻り値	ありません.
留意事項	特にありません.

eyemMatZero

機 能	行列のゼロクリア	
形 式	<pre>#include "eyemLib.h" void eyedMatZero (int m, int n, void *vpA);</pre>	
解 説	$m \times n$ 行列 A のすべての要素をゼロに設定します.	
引 数	m	行列 A の行数です.
	n	行列 A の列数です.
	*vpA	$m \times n$ 行列 A です. (double[m][n] または double[m*n] の配列)
戻り値	ありません.	
留意事項	特にありません.	

eyemMatDiagonal

機 能	対角行列(単位行列のスカラー倍)の設定	
形 式	<pre>#include "eyemLib.h" void eyedMatDiagonal (double dAlpha, int n, void *vpA);</pre>	
解 説	正方行列 A を単位行列のスカラー倍 α として設定します.	
引 数	dAlpha	対角成分の値 α です.
	n	正方行列 A の次数です.
	*vpA	$n \times n$ 正方行列 A です. (double[n][n] または double[n*n] の配列)
戻り値	ありません.	
留意事項	特にありません.	

eyemMatCopy

機 能	行列のコピー	
形 式	<pre>#include "eyemLib.h" void eyedMatCopy (int m, int n, void *vpA, void *vpB);</pre>	
解 説	行列 A を行列 B にコピーします.	
引 数	m	行列の行数です.
	n	行列の列数です.
	*vpA	コピー元の $m \times n$ 行列 A です. (double[m][n] または double[m*n] の配列)
	*vpB	コピー先の $m \times n$ 行列 B です. (double[m][n] または double[m*n] の配列)
戻り値	ありません.	
留意事項	特にありません.	

eyemMatAdd

機 能 行列の加算

形 式 `#include "eyemLib.h"`
`void eyedMatAdd (int m, int n, void *vpA, void *vpB, void *vpC);`

解 説 行列 **A** と行列 **B** の加算を行い, 行列 **C** に結果 (**A + B**) を格納します.

引 数 `m` 行列の行数です.
 `n` 行列の列数です.
 `*vpA` $m \times n$ 行列 **A** です. (`double[m][n]` または `double[m*n]` の配列)
 `*vpB` $m \times n$ 行列 **B** です. (`double[m][n]` または `double[m*n]` の配列)
 `*vpC` 加算 (**A + B**) の結果を格納する $m \times n$ 行列 **C** です. `vpA` または `vpB` と同じでも構いません. (`double[m][n]` または `double[m*n]` の配列)

戻り値 ありません.

留意事項 特にありません.

eyemMatSub

機 能 行列の減算

形 式 `#include "eyemLib.h"`
`void eyedMatSub (int m, int n, void *vpA, void *vpB, void *vpC);`

解 説 行列 **A** と行列 **B** の減算を行い, 行列 **C** に結果 (**A** − **B**) を格納します.

引 数 `m` 行列の行数です.
 `n` 行列の列数です.
 `*vpA` $m \times n$ 行列 **A** です. (`double[m][n]` または `double[m*n]` の配列)
 `*vpB` $m \times n$ 行列 **B** です. (`double[m][n]` または `double[m*n]` の配列)
 `*vpC` 減算 (**A** − **B**) の結果を格納する $m \times n$ 行列 **C** です. `vpA` または `vpB` と同じでも構いません. (`double[m][n]` または `double[m*n]` の配列)

戻り値 ありません.

留意事項 特にありません.

eyemMatMulScalar

機 能	行列のスカラー倍	
形 式	<pre>#include "eyemLib.h" void eyedMatMulScalar (double dAlpha, int m, int n, void *vpA, void *vpB);</pre>	
解 説	行列 A をスカラー α 倍し, 行列 B に結果 ($\alpha\mathbf{A}$) を格納します.	
引 数	dAlpha	乗ずるスカラー値 α です.
	m	行列の行数です.
	n	行列の列数です.
	*vpA	$m \times n$ 行列 A です. (double[m][n] または double[m*n] の配列)
	*vpB	スカラー倍の結果 ($\alpha\mathbf{A}$) を格納する $m \times n$ 行列 B です. vpAと同じでも構いません. (double[m][n] または double[m*n] の配列)
戻り値	ありません.	
留意事項	特にありません.	

eyemMatMul

機 能 行列の乗算

形 式 `#include "eyemLib.h"`
`int eyemMatMul (int am, int an, void *vpA, int bm, int bn, void *vpB, void *vpC);`

解 説 行列 **A** と行列 **B** を掛け算し, 行列 **C** に結果 (**AB**) を格納します.

引 数

<code>am</code>	行列 A の行数です.
<code>an</code>	行列 A の列数です.
<code>*vpA</code>	$am \times an$ 行列 A です. (<code>double[am][an]</code> または <code>double[am*an]</code> の配列)
<code>bm</code>	行列 B の列数です. <code>an</code> と等しいことに注意してください.
<code>bn</code>	行列 B の列数です.
<code>*vpB</code>	$bm \times bn$ 行列 B です. (<code>double[bm][bn]</code> または <code>double[bm*bn]</code> の配列)
<code>*vpC</code>	乗算の結果 (AB) を格納する $am \times bn$ 行列 C です. (<code>double[am][bn]</code> または <code>double[am*bn]</code> の配列)

戻り値 エラー報告です.

FUNC_OK	正常終了
FUNC_CANNOT_CALC	計算が不可 ($an \neq bm$ である)

留意事項 特にありません.

eyemMatGenMul

機 能 汎用的な行列の乗算

形 式

```
#include "eyemLib.h"

int      eyedMatGenMul ( double dAlpha, int am, int an, void *vpA, int bm, int bn,
                        void *vpB, double dBeta, void *vpC, void *vpD );
```

解 説 行列 **A**, **B**, **C** とスカラー α , β に対して, 汎用的な掛け算 $\alpha\mathbf{AB} + \beta\mathbf{C}$ を計算し, 行列 **D** に結果を格納します.

引 数

dAlpha	スカラー倍 α です.
am	行列 A の行数です.
an	行列 A の列数です.
*vpA	$am \times an$ 行列 A です. (double[am][an] または double[am*an] の配列)
bm	行列 B の列数です. an と等しいことに注意してください.
bn	行列 B の列数です.
*vpB	$bm \times bn$ 行列 B です. (double[bm][bn] または double[bm*bn] の配列)
dBeta	スカラー倍 β です.
*vpC	$am \times bn$ 行列 C です. (double[am][bn] または double[am*bn] の配列)
*vpD	汎用的な乗算の結果 ($\alpha\mathbf{AB} + \beta\mathbf{C}$) を格納する $am \times bn$ 行列 D です. (double[am][bn] または double[am*bn] の配列)

戻り値 エラー報告です.

FUNC_OK	正常終了
FUNC_CANNOT_CALC	計算が不可 (an \neq bm である)

留意事項 特にありません.

eyemMatMulTransposed

機 能	行列とその転置行列との乗算	
形 式	<pre>#include "eyemLib.h" void eyedMatMulTransposed (int iOrder, int m, int n, void *vpA, void *vpB);</pre>	
解 説	行列 A とその転置行列 A^T に対して、それらの掛け算 A × A^T または A^T × A を計算し、行列 B に結果を格納します。	
引 数	iOrder	掛け算の順番です。 A × A^T は 0 , A^T × A は 1 を指定します。
	m	行列 A の行数です。
	n	行列 A の列数です。
	*vpA	m × n 行列 A です。(double[m][n] または double[m*n] の配列)
	*vpB	乗算の結果を格納する m × n 行列 B です。(double[m][n] または double[m*n] の配列)
戻り値	ありません。	
留意事項	特にありません。	

eyemMatMulTransposed2

機 能 行列と別の行列の転置行列との乗算

形 式

```
#include "eyemLib.h"

int      eyedMatMulTransposed2 ( int iOrder, int am, int an, void *vpA, int bm, int bn,
                                void *vpB, void *vpC );
```

解 説 行列 **A** と行列 **B** のそれぞれの転置行列 \mathbf{A}^T , \mathbf{B}^T に対して, それらの掛け算 $\mathbf{A} \times \mathbf{B}^T$ または $\mathbf{A}^T \times \mathbf{B}$ を計算し, 行列 **C** に結果を格納します.

引 数

iOrder	掛け算の順番です. $\mathbf{A} \times \mathbf{B}^T$ は 0 , $\mathbf{A}^T \times \mathbf{B}$ は 1 を指定します.
am	行列 A の行数です.
an	行列 A の列数です.
*vpA	$am \times an$ 行列 A です. (double[am][an] または double[am*an] の配列)
bm	行列 B の行数です.
bn	行列 B の列数です.
*vpB	$bm \times bn$ 行列 B です. (double[bm][bn] または double[bm*bn] の配列)
*vpC	乗算の結果を格納する行列 C です. (iOrder = 0 の場合: double[am][bn] または double[am*bn] の配列) (iOrder = 1 の場合: double[an][bn] または double[an*bn] の配列)

戻り値 エラー報告です.

FUNC_OK	正常終了
FUNC_CANNOT_CALC	計算が不可 (iOrder = 0 の場合: $an \neq bn$ である) (iOrder = 1 の場合: $am \neq bn$ である)

留意事項 特にありません.

eyemMatTransposed

機 能 行列の転置

形 式 `#include "eyemLib.h"`
`void eyedMatTransposed (int m, int n, void *vpA, void *vpB);`

解 説 行列 **A** を転置し, 行列 **B** に結果 (\mathbf{A}^T) を格納します.

引 数 `m` 行列 **A** の行数です.
 `n` 行列 **A** の列数です.
 `*vpA` $m \times n$ 行列 **A** です. (`double[m][n]` または `double[m*n]` の配列)
 `*vpB` 転置の結果 (\mathbf{A}^T) を格納する $n \times m$ 行列 **B** です. (`double[n][m]` または `double[n*m]` の配列)

戻り値 ありません.

留意事項 特にありません.

eyemMatTrace

機 能	行列のトレース				
形 式	<pre>#include "eyemLib.h" double eyedMatTrace (int n, void *vpA);</pre>				
解 説	正方行列 A のトレース(対角成分の和)を計算します.				
引 数	<table><tr><td>n</td><td>正方行列 A の次数です.</td></tr><tr><td>*vpA</td><td>n×n正方行列 A です. (double[n][n] または double[n*n] の配列)</td></tr></table>	n	正方行列 A の次数です.	*vpA	n×n正方行列 A です. (double[n][n] または double[n*n] の配列)
n	正方行列 A の次数です.				
*vpA	n×n正方行列 A です. (double[n][n] または double[n*n] の配列)				
戻り値	行列のトレース(対角成分の和)です.				
留意事項	特にありません.				

eyemMatDet

機 能 行列の行列式

形 式 `#include "eyemLib.h"`
`int eyedMatDet(int n, void *vpA, double *dpDet);`

解 説 正方行列 **A** の行列式を計算します.

引 数 `n` 正方行列 **A** の次数です.
 `*vpA` `n×n` 正方行列 **A** です. (`double[n][n]` または `double[n*n]` の配列)
 `*dpDet` 行列式の値を格納します.

戻り値 エラー報告です.

FUNC_OK	正常終了
FUNC_NOT_ENOUGH_MEM	メモリ不足

留意事項 特にありません.

eyemMatInvert

機 能 行列の逆行列

形 式 `#include "eyemLib.h"`
`int eyedMatInvert (int n, void *vpA, void *vpInvA);`

解 説 正方行列 **A** の逆行列を計算し, 行列 *InvA* に結果 (\mathbf{A}^{-1}) を格納します.

引 数 `n` 正方行列 **A** の次数です.
 `*vpA` $n \times n$ 正方行列 **A** です. (double[n][n] または double[n*n] の配列)
 `*vpInvA` 逆行列 (\mathbf{A}^{-1}) を格納する $n \times n$ 正方行列 *InvA* です. (double[n][n] または double[n*n] の配列)

戻り値 エラー報告です.

FUNC_OK	正常終了
FUNC_NOT_ENOUGH_MEM	メモリ不足
FUNC_DET_EQ_ZERO	行列式がゼロ

留意事項 特にありません.

eyemMatSolveLU

機 能 連立 1 次方程式の解 (LU 分解による)

形 式 `#include "eyemLib.h"`
`int eyemMatSolveLU (int n, void *vpA, double daB[], double daX[]);`

解 説 連立 1 次方程式 $\mathbf{Ax} = \mathbf{b}$ の解ベクトル \mathbf{x} を LU 分解により求めます.

引 数

<code>n</code>	係数行列 \mathbf{A} の次数です.
<code>*vpA</code>	係数行列 \mathbf{A} ($n \times n$ 正方行列) です. (<code>double[n][n]</code> または <code>double[n*n]</code> の配列)
<code>daB[]</code>	定数ベクトル \mathbf{b} です. (<code>double[n]</code> の配列)
<code>daX[]</code>	連立方程式の解を格納するベクトル \mathbf{x} です. (<code>double[n]</code> の配列)

戻り値 エラー報告です.

FUNC_OK	正常終了
FUNC_NOT_ENOUGH_MEM	メモリ不足
FUNC_DET_EQ_ZERO	行列式がゼロ

留意事項 特にありません.

eyemMatSolveSVD

機 能 連立 1 次方程式の解(特異値分解(SVD)による), または最小二乗解

形 式 `#include "eyemLib.h"`
`int eyedMatSolveSVD (int am, int an, void *vpA, double daB[], double daX[]);`

解 説 連立 1 次方程式 $\mathbf{Ax} = \mathbf{b}$ の解ベクトル \mathbf{x} を, 特異値分解により求めます. なお, 係数行列 \mathbf{A} が正方行列でない場合の解は, 最小二乗解となります.

引 数

am	係数行列 \mathbf{A} の行数です.
an	係数行列 \mathbf{A} の列数です.
*vpA	係数行列 \mathbf{A} ($am \times an$ 行列) です. (<code>double[am][an]</code> または <code>double[am*an]</code> の配列)
daB[]	定数ベクトル \mathbf{b} です. (<code>double[am]</code> の配列)
daX[]	連立方程式の解を格納するベクトル \mathbf{x} です. (<code>double[an]</code> の配列)

戻り値 エラー報告です.

FUNC_OK	正常終了
FUNC_NOT_ENOUGH_MEM	メモリ不足
FUNC_FAILED_SVD	特異値分解失敗

留意事項 特にありません.

eyemMatJacobiEigen

機 能 実対称行列の固有値と固有ベクトル (Jacobi 法)

形 式 `#include "eyemLib.h"`
`int eyemMatJacobiEigen (int n, void *vpA, double daD[], void *vpW);`

解 説 実対称行列 **A** の固有値と固有ベクトルを Jacobi 法により求め、ベクトル **d** に固有値、行列 **W** にその固有ベクトルを格納します。処理時間の観点から、Jacobi 法は、行列の次数が10程度までの場合に有効な方法です。精度の観点では、QR 法より良いです。

引 数

<code>n</code>	実対称行列 A の次数です。
<code>*vpA</code>	$n \times n$ 実対称行列 A です。(double[n][n] または double[n*n] の配列)
<code>daD[]</code>	固有値を格納するベクトル d です(double[n]の配列)。ベクトル d には、固有値が降順に格納されます。なお、固有値が不要な場合は、NULLを入力してください。
<code>*vpW</code>	固有ベクトルを格納する行列 W です(double[n][n] または double[n*n] の配列)。固有値 <code>daD[k]</code> に対する固有ベクトルは、 <code>daW[k][0], ..., daW[k][n-1]</code> に格納されます。各固有ベクトルは単位ベクトルとなっています。なお、固有ベクトルが不要な場合は、NULLを入力してください。

戻り値 エラー報告です。

FUNC_OK	正常終了
FUNC_NOT_ENOUGH_MEM	メモリ不足
FUNC_FAILED_EIGEN	固有値計算失敗

留意事項 行列の次数が10を大きく超える場合は処理時間がかかります。

eyemMatQrEigen

機 能 実対称行列の固有値と固有ベクトル (QR 法)

形 式

```
#include "eyemLib.h"

int      eyedMatQrEigen ( int n, void *vpA, double daD[], void *vpW );
```

解 説 実対称行列 **A** の固有値と固有ベクトルを QR 法により求め、ベクトル **d** に固有値、行列 **W** にその固有ベクトルを格納します。処理時間の観点から、QR 法は、行列の次数が10を超える場合に有効な方法です。精度の観点では、Jacobi 法の方が良いです。

引 数

n	実対称行列 A の次数です。
*vpA	$n \times n$ 実対称行列 A です。(double[n][n] または double[n*n] の配列)
daD[]	固有値を格納するベクトル d です(double[n]の配列)。ベクトル d には、固有値が降順に格納されます。なお、固有値が不要な場合は、NULLを入力してください。
*vpW	固有ベクトルを格納する行列 W です(double[n][n] または double[n*n] の配列)。固有値 daD[k] に対する固有ベクトルは、daW[k][0], ..., daW[k][n-1] に格納されます。各固有ベクトルは単位ベクトルとなっています。なお、固有ベクトルが不要な場合は、NULLを入力してください。

戻り値 エラー報告です。

FUNC_OK	正常終了
FUNC_NOT_ENOUGH_MEM	メモリ不足
FUNC_FAILED_EIGEN	固有値計算失敗

留意事項 特にありません。

eyemMatSVD

機 能 行列の特異値分解(SVD)

形 式 `#include "eyemLib.h"`
`int eyedMatSVD (int m, int n, void *vpA, void *vpU, double daD[], void *vpVT);`

解 説 行列 **A** の特異値分解 ($\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^T$) を行い, 列直交行列 **U**, 特異値 **D** (ベクトルで格納) および直交行列 \mathbf{V}^T を求めます.

引 数

m	行列 A の行数です ($m \geq n$ とします).
n	行列 A の列数です ($m \geq n$ とします).
*vpA	$m \times n$ 行列 A です. (double[m][n] または double[m*n] の配列)
*vpU	行列 U ($m \times n$ 行列) を格納する配列です (double[m][n] または double[m*n] の配列). なお, 行列 U が不要な場合は, NULL を入力してください.
daD[]	特異値を格納するベクトル D です (double[n] の配列). ベクトル D には, 特異値が降順に格納されます. なお, 特異値が不要な場合は, NULL を入力してください.
*vpVT	行列 \mathbf{V}^T ($n \times n$ 行列) を格納する配列です (double[n][n] または double[n*n] の配列). なお, 行列 \mathbf{V}^T が不要な場合は, NULL を入力してください.

戻り値 エラー報告です.

FUNC_OK	正常終了
FUNC_NOT_ENOUGH_MEM	メモリ不足
FUNC_FAILED_SVD	特異値分解失敗

留意事項 行数は列数以上 ($m \geq n$) とします. もし, $m < n$ ならば, あらかじめ行列 **A** の下部にゼロばかりの行を追加して, 正方行列 ($m = n$) にしてから関数を用いてください.

eyemMatCholeskyDecomp

機 能 実対称正定値行列のコレスキー分解

形 式 `#include "eyemLib.h"`
`int eyemMatCholeskyDecomp (int n, void *vpA, void *vpU);`

解 説 行列 **A** のコレスキー分解 ($\mathbf{A} = \mathbf{U}^T \times \mathbf{U}$) を行い, 上三角行列 **U** を求めます.

引 数

<code>n</code>	行列 A の次数です.
<code>*vpA</code>	実対称正定値行列 A ($n \times n$ 行列) です. (<code>double[n][n]</code> または <code>double[n*n]</code> の配列)
<code>*vpU</code>	上三角行列 U ($n \times n$ 行列) を格納する配列です. (<code>double[n][n]</code> または <code>double[n*n]</code> の配列)

戻り値 エラー報告です.

FUNC_OK	正常終了
FUNC_FAILED_CHOLESKY	コレスキー分解失敗 (行列Aが正定値でない)

留意事項 特にありません.

改訂履歴

Version No.	内 容
1.0	<ul style="list-style-type: none"> 新規発行
1.1	<ul style="list-style-type: none"> 誤記の修正および説明文の追記
1.2	<ul style="list-style-type: none"> 固有値 (eyemMatJacobiEigen) 関数の説明追記
1.3	<ul style="list-style-type: none"> 固有値 (eyemMatJacobiEigen) 関数の説明修正 固有値 (eyemMatQrEigen) 関数の新規追加
1.4	<ul style="list-style-type: none"> 行列式計算 (eyemMatDet) 関数の新規追加