

数学ユーティリティー

(Ver.1.3)

2016 年 5 月

株式会社 アイディール

目次

eyemMathCalcInnerProduct	1
機能 内積	1
eyemMathCalcOuterProduct	2
機能 外積	2
eyemMathCalcAngle.....	3
機能 2ベクトルのなす角	3
eyemMathCalcNorm.....	4
機能 ベクトルの大きさ(ユークリッドノルム)	4
eyemMathCalcArgument	5
機能 2次元ベクトルの偏角	5
eyemMathNormalization.....	6
機能 ベクトルの正規化(単位ベクトル化)	6
eyemMathStat.....	7
機能 データの統計量(平均, 分散, 標準偏差)	7
eyemMathMedianI.....	8
機能 int 型データのメディアン	8
eyemMathMedianD	9
機能 double 型データのメディアン	9
eyemMathOtsuThreshold1d.....	10
機能 判別分析法によるデータの分離しきい値.....	10
eyemMathRotatePoint.....	11
機能 指定点まわりの θ 回転(座標変換)	11
eyemMathGetDistFromPointToEllipse.....	12
機能 点から楕円への最近点.....	12
eyemMathTransAbcToRq.....	13
機能 ヘッセの標準形.....	13
eyemMathExtremumOfQuadraticCurves.....	14
機能 2次曲線(放物線)の極値とその x 座標	14
eyemMathQuadraticRoots	15
機能 2次方程式の実数解.....	15
eyemMathCubicRoots.....	16
機能 3次方程式の実数解.....	16
eyemMathQuarticRoots.....	17
機能 4次方程式の実数解.....	17
eyemMathHorner.....	18
機能 ホーナー法による n 次多項式の計算	18

eyemMathCalcInnerProduct

機 能	内積						
形 式	<pre>#include "eyemLib.h" double eyedMathCalcInnerProduct (int iDim, double daV0[], double daV1[]);</pre>						
解 説	<p>二つの n 次元ベクトル \mathbf{v}_0, \mathbf{v}_1 の内積 $(\mathbf{v}_0, \mathbf{v}_1)$ を求めます.</p> <p>$\mathbf{v}_0 = (x_1, x_2, \dots, x_n)^T$, $\mathbf{v}_1 = (y_1, y_2, \dots, y_n)^T$ とするとき, 内積 $(\mathbf{v}_0, \mathbf{v}_1)$ は</p> $(\mathbf{v}_0, \mathbf{v}_1) = x_1 y_1 + x_2 y_2 + \dots + x_n y_n$ <p>となります.</p>						
引 数	<table><tr><td>iDim</td><td>ベクトルの次元 n です.</td></tr><tr><td>daV0[]</td><td>ベクトル \mathbf{v}_0 の1次元配列です.</td></tr><tr><td>daV1[]</td><td>ベクトル \mathbf{v}_1 の1次元配列です.</td></tr></table>	iDim	ベクトルの次元 n です.	daV0[]	ベクトル \mathbf{v}_0 の1次元配列です.	daV1[]	ベクトル \mathbf{v}_1 の1次元配列です.
iDim	ベクトルの次元 n です.						
daV0[]	ベクトル \mathbf{v}_0 の1次元配列です.						
daV1[]	ベクトル \mathbf{v}_1 の1次元配列です.						
戻り値	内積 $(\mathbf{v}_0, \mathbf{v}_1)$ です.						
留意事項	特にありません.						

eyemMathCalcOuterProduct

機 能	外積						
形 式	<pre>#include "eyemLib.h" void eyedMathCalcOuterProduct (double daV0[], double daV1[], double daV2[]);</pre>						
解 説	<p>二つの3次元ベクトル \mathbf{v}_0, \mathbf{v}_1 の外積 $\mathbf{v}_0 \times \mathbf{v}_1$ を求めます.</p> <p>$\mathbf{v}_0 = (x_1, x_2, x_3)^T$, $\mathbf{v}_1 = (y_1, y_2, y_3)^T$ とするとき, 外積 $\mathbf{v}_0 \times \mathbf{v}_1$ は</p> $\mathbf{v}_0 \times \mathbf{v}_1 = (x_2 y_3 - x_3 y_2, x_3 y_1 - x_1 y_3, x_1 y_2 - x_2 y_1)$ <p>となります.</p>						
引 数	<table><tr><td>daV0[]</td><td>ベクトル \mathbf{v}_0 の1次元配列です.</td></tr><tr><td>daV1[]</td><td>ベクトル \mathbf{v}_1 の1次元配列です.</td></tr><tr><td>daV2[]</td><td>外積 $\mathbf{v}_0 \times \mathbf{v}_1$ が格納されます.</td></tr></table>	daV0[]	ベクトル \mathbf{v}_0 の1次元配列です.	daV1[]	ベクトル \mathbf{v}_1 の1次元配列です.	daV2[]	外積 $\mathbf{v}_0 \times \mathbf{v}_1$ が格納されます.
daV0[]	ベクトル \mathbf{v}_0 の1次元配列です.						
daV1[]	ベクトル \mathbf{v}_1 の1次元配列です.						
daV2[]	外積 $\mathbf{v}_0 \times \mathbf{v}_1$ が格納されます.						
戻り値	ありません.						
留意事項	特にありません.						

eyemMathCalcAngle

機 能 2ベクトルのなす角

形 式 `#include "eyemLib.h"`
`int eyemMathCalcAngle (int iDim, double daV0[], double daV1[], double *dpAngle);`

解 説 二つの n 次元ベクトル \mathbf{v}_0 , \mathbf{v}_1 のなす角 θ (rad 単位)を求めます.
 \mathbf{v}_0 , \mathbf{v}_1 のなす角 θ は, 内積 $(\mathbf{v}_0, \mathbf{v}_1)$ およびユークリッドノルム $\|\mathbf{v}_0\|$, $\|\mathbf{v}_1\|$ を用いて,
$$\theta = \cos^{-1} \frac{(\mathbf{v}_0, \mathbf{v}_1)}{\|\mathbf{v}_0\| \|\mathbf{v}_1\|}$$
となります.

引 数 `iDim` ベクトルの次元 n です.
`daV0[]` ベクトル \mathbf{v}_0 の1次元配列です.
`daV1[]` ベクトル \mathbf{v}_1 の1次元配列です.
`*dpAngle` \mathbf{v}_0 , \mathbf{v}_1 のなす角 θ (rad単位)が格納されます.

戻り値 エラー報告です.

FUNC_OK	正常終了
FUNC_CANNOT_CALC	計算が不可 (\mathbf{v}_0 または \mathbf{v}_1 がゼロベクトル)

留意事項 特にありません.

eyemMathCalcNorm

機 能	ベクトルの大きさ(ユークリッドノルム)	
形 式	<pre>#include "eyemLib.h" double eyedMathCalcNorm (int iDim, double daV[]);</pre>	
解 説	<p>n 次元ベクトル \mathbf{v} の大きさ(ユークリッドノルム) $\ \mathbf{v}\$ を求めます.</p> <p>$\mathbf{v} = (x_1, x_2, \dots, x_n)^T$ とするとき, ユークリッドノルム $\ \mathbf{v}\$ は,</p> $\ \mathbf{v}\ = \sqrt{x_1^2 + x_2^2 + \dots + x_n^2}$ <p>となります.</p>	
引 数	iDim	ベクトルの次元 n です.
	daV[]	ベクトル \mathbf{v} の1次元配列です.
戻り値	ユークリッドノルム $\ \mathbf{v}\ $ です.	
留意事項	特にありません.	

eyemMathCalcArgument

機 能	2次元ベクトルの偏角
形 式	<pre>#include "eyemLib.h" double eyedMathCalcArgument (double daV[]);</pre>
解 説	<p>2次元ベクトル \mathbf{v} が x 軸となす角 (偏角) θ (rad 単位) を求めます. ベクトルの回転方向に合わせて, 符号がつきます.</p> <p>$\mathbf{v} = (x_1, x_2)^T$ とするとき, 偏角 θ は $\theta = \text{atan2}(x_2, x_1)$ で求めます.</p>
引 数	daV[] ベクトル \mathbf{v} の1次元配列です.
戻り値	偏角 θ (rad単位) です.
留意事項	特にありません.

eyemMathNormalization

機 能 ベクトルの正規化(単位ベクトル化)

形 式 #include “eyemLib.h”
int eyemMathNormalization (int iDim, double daN[]);

解 説 n 次元ベクトル \mathbf{v} を単位ベクトル \mathbf{n} に変換します。
 $\mathbf{v} = (x_1, x_2, \dots, x_n)^T$ とするとき, その単位ベクトル \mathbf{n} はユークリッドノルム $\|\mathbf{v}\|$ を用いて

$$\mathbf{n} = \frac{\mathbf{v}}{\|\mathbf{v}\|}$$

となります。

引 数 iDim ベクトルの次元 n です。
daN[] ベクトル \mathbf{v} の1次元配列であって, ここに単位ベクトル \mathbf{n} が格納されます。

戻り値 エラー報告です。

FUNC_OK	正常終了
FUNC_CANNOT_CALC	計算が不可 (\mathbf{v} がゼロベクトル)

留意事項 特にありません。

eyemMathStat

機 能 データの統計量(平均, 分散, 標準偏差)

形 式

```
#include "eyemLib.h"

int      eyemMathStat ( int iDataNum, double daData[],
                        double *dpMean, double *dpVar, double *dpStdDev );
```

解 説 データの平均, 分散および標準偏差を求めます.
 n 個のデータを d_1, d_2, \dots, d_n とするとき, それらの平均 \bar{d} , 分散 S^2 , 標準偏差 S は

$$\bar{d} = \frac{1}{n} \sum_{i=1}^n d_i, \quad S^2 = \frac{1}{n} \sum_{i=1}^n (d_i - m)^2, \quad S = \sqrt{S^2}$$

となります.

引 数

iDataNum	データ数です.
daData[]	データの1次元配列です.
*dpMean	データの平均が格納されます.
*dpVar	データの分散が格納されます.
*dpStdDev	データの標準偏差が格納されます.

戻り値 エラー報告です.

FUNC_OK	正常終了
FUNC_CANNOT_CALC	計算が不可(データ数がゼロ)

留意事項 特にありません.

eyemMathMedianI

機 能	int 型データのメディアン				
形 式	<pre>#include "eyemLib.h" double eyedMathMedianI (int n, int iaData[]);</pre>				
解 説	<p>int 型データのメディアン(中央値)を求めます.</p> <p>ソートされた n 個のデータを d_1, d_2, \dots, d_n とするとき, それらのメディアン med は</p> $med = \begin{cases} d_{(n+1)/2}, & (n \text{ が奇数のとき}) \\ \frac{1}{2}(d_{n/2} + d_{n/2+1}), & (n \text{ が偶数のとき}) \end{cases}$ <p>となります.</p>				
引 数	<table><tr><td>n</td><td>データ数です.</td></tr><tr><td>iaData[]</td><td>データの1次元配列です.</td></tr></table>	n	データ数です.	iaData[]	データの1次元配列です.
n	データ数です.				
iaData[]	データの1次元配列です.				
戻り値	メディアン(中央値)です.				
留意事項	ソートを行うため, 出力時の配列要素順は入力時と異なります.				

eyemMathMedianD

機 能	double 型データのメディアン				
形 式	<pre>#include "eyemLib.h" double eyemMathMedianD (int n, double daData[]);</pre>				
解 説	<p>double 型データのメディアン(中央値)を求めます.</p> <p>ソートされた n 個のデータを d_1, d_2, \dots, d_n とするとき, それらのメディアン med は</p> $med = \begin{cases} d_{(n+1)/2}, & (n \text{ が奇数のとき}) \\ \frac{1}{2}(d_{n/2} + d_{n/2+1}), & (n \text{ が偶数のとき}) \end{cases}$ <p>となります.</p>				
引 数	<table><tr><td>n</td><td>データ数です.</td></tr><tr><td>daData[]</td><td>データの1次元配列です.</td></tr></table>	n	データ数です.	daData[]	データの1次元配列です.
n	データ数です.				
daData[]	データの1次元配列です.				
戻り値	メディアン(中央値)です.				
留意事項	ソートを行うため, 出力時の配列要素順は入力時と異なります.				

eyemMathOtsuThreshold1d

機 能 判別分析法によるデータの分離しきい値

形 式 `#include "eyemLib.h"`
`int eyemMathOtsuThreshold1d (int n, double daData[], double *dpThreshold);`

解 説 判別分析法(大津の方法)により, 1次元データを2分割する分離しきい値を求めます.
 n 個のデータを昇順に d_i ($i=1, \dots, n$) とし, それらの平均を μ_T とします. このとき, $i=k$ における次式のクラス間分散 $\sigma_B^2(k)$ が最大となる $k=k^*$ を求めます.

$$\sigma_B^2(k) = \frac{\left(\mu_T k - \sum_{i=1}^k d_i \right)^2}{k(n-k)}.$$

すると, 出力される分離しきい値は, $(d_{k^*} + d_{k^*+1})/2$ となります.

引 数 `n` データ数です.
`daData[]` データの1次元配列です.
`*dpThreshold` データの分離しきい値が格納されます.

戻り値 エラー報告です.

<code>FUNC_OK</code>	正常終了
<code>FUNC_CANNOT_CALC</code>	計算が不可(データ数がゼロ)

留意事項 特にありません.

eyemMathRotatePoint

機 能 指定点まわりの θ 回転(座標変換)

形 式

```
#include "eyemLib.h"

void      eyedMathRotatePoint ( double dCx, double dCy, double dTheta, double dSrcPtX,
                                double dSrcPtY, double *dpDstPtX, double *dpDstPtY );
```

解 説 指定された点を回転中心とした θ 回転の座標変換を行います。
点 (p_x, p_y) が回転中心 (C_x, C_y) のまわりの θ 回転により点 (P_x, P_y) に移ったとすると、

$$\begin{pmatrix} P_x \\ P_y \end{pmatrix} = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} p_x - C_x \\ p_y - C_y \end{pmatrix} + \begin{pmatrix} C_x \\ C_y \end{pmatrix}$$

となります。

引 数	dCx	回転中心 x 座標 C_x です。
	dCy	回転中心 y 座標 C_y です。
	dTheta	回転角 θ (rad単位) です。
	dSrcPtX	回転を施す点の x 座標 p_x です。
	dSrcPtY	回転を施す点の y 座標 p_y です。
	*dpDstPtX	回転後の点の x 座標 P_x が格納されます。
	*dpDstPtY	回転後の点の y 座標 P_y が格納されます。

戻り値 ありません。

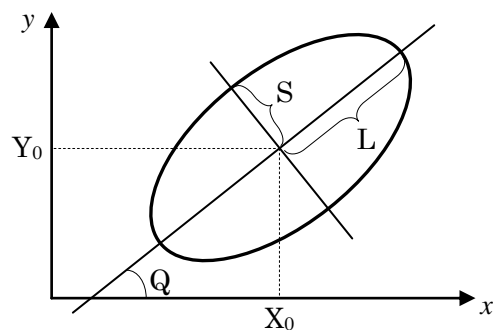
留意事項 特にありません。

eyemMathGetDistFromPointToEllipse

機 能 点から楕円への最近点

形 式 `#include "eyemLib.h"`
`int eyedMathGetDistFromPointToEllipse (double dXo, double dYo, double dL,`
`double dS, double dQ, double d_vXp, double d_vYp,`
`double *dp_vXe, double *dp_vYe, double *dpDist);`

解 説 指定点から指定された楕円への最近点と、そこまでの距離を求めます.



引 数	dXo	楕円の中心の x 座標 X_0 です.
	dYo	楕円の中心の y 座標 Y_0 です.
	dL	楕円の長軸半径 L です.
	dS	楕円の短軸半径 S です.
	dQ	楕円の傾斜角 Q です (単位:rad). x 軸から測った長軸のなす角です.
	d_vXp	指定点の x 座標です.
	d_vYp	指定点の y 座標です.
	*dp_vXe	最近点の x 座標が格納されます.
	*dp_vYe	最近点の y 座標が格納されます.
	*dpDist	最近点までの距離が格納されます.

戻り値 エラー報告です.

FUNC_OK	正常終了
FUNC_CANNOT_CALC	計算が不可

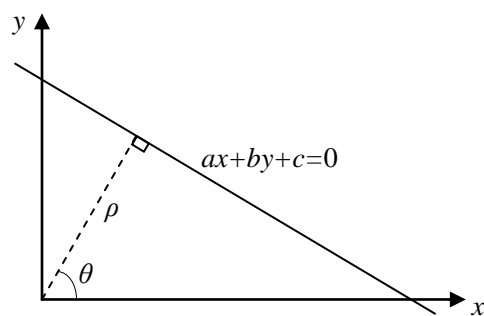
留意事項 特にありません.

eyemMathTransAbcToRq

機 能 ヘッセの標準形

形 式 `#include "eyemLib.h"`
`void eyedMathTransAbcToRq (double dA, double dB, double dC, double *dpR,`
`double *dpQ);`

解 説 直線 $ax + by + c = 0$ をヘッセの標準形 $\rho = x \cos \theta + y \sin \theta$ で表したときの ρ および θ を求めます.



引 数	dA	直線 $ax + by + c = 0$ の係数 a です.
	dB	直線 $ax + by + c = 0$ の係数 b です.
	dC	直線 $ax + by + c = 0$ の係数 c です.
	*dpR	ヘッセの標準形の ρ が格納されます.
	*dpQ	ヘッセの標準形の θ が格納されます.

戻り値 ありません.

留意事項 特にありません.

eyemMathExtremumOfQuadraticCurves

機 能	2次曲線(放物線)の極値とその x 座標	
形 式	<pre>#include "eyemLib.h" double eyedMathExtremumOfQuadraticCurves (double dA, double dB, double dC, double *dpX);</pre>	
解 説	<p>2次曲線(放物線)の極値(最大値または最小値)とその値をとる x 座標を求めます.</p> <p>放物線 $y = ax^2 + bx + c$ の極値をとる x 座標は,</p> $x = -\frac{b}{2a}$ <p>となります.</p>	
引 数	dA	放物線の係数 a です.
	dB	放物線の係数 b です.
	dC	放物線の係数 c です.
	*dpX	極値をとる x 座標が格納されます.
戻り値	極値(最大値または最小値)です.	
留意事項	特にありません.	

eyemMathQuadraticRoots

機 能	2次方程式の実数解				
形 式	<pre>#include "eyemLib.h" int eyedMathQuadraticRoots (double daCoeffs[], double daRoots[]);</pre>				
解 説	2次方程式 $ax^2 + bx + c = 0$ の実数解を求めます.				
引 数	<table><tr><td>daCoeffs[]</td><td>2次方程式の係数です. [0]に係数 a , [1]に係数 b , [2]に係数 c を入力して下さい.</td></tr><tr><td>daRoots[]</td><td>実数解が格納されます. 要素数2以上の配列を用意して下さい.</td></tr></table>	daCoeffs[]	2次方程式の係数です. [0]に係数 a , [1]に係数 b , [2]に係数 c を入力して下さい.	daRoots[]	実数解が格納されます. 要素数2以上の配列を用意して下さい.
daCoeffs[]	2次方程式の係数です. [0]に係数 a , [1]に係数 b , [2]に係数 c を入力して下さい.				
daRoots[]	実数解が格納されます. 要素数2以上の配列を用意して下さい.				
戻り値	実数解の個数です.				
留意事項	特にありません.				

eyemMathCubicRoots

機 能	3次方程式の実数解				
形 式	<pre>#include "eyemLib.h" int eyedMathCubicRoots (double daCoeffs[], double daRoots[]);</pre>				
解 説	3次方程式 $ax^3 + bx^2 + cx + d = 0$ の実数解を求めます.				
引 数	<table><tr><td>daCoeffs[]</td><td>3次方程式の係数です. [0]に係数 a , [1]に係数 b , [2]に係数 c , [3]に係数 d を入力して下さい.</td></tr><tr><td>daRoots[]</td><td>実数解が格納されます. 要素数3以上の配列を用意して下さい.</td></tr></table>	daCoeffs[]	3次方程式の係数です. [0]に係数 a , [1]に係数 b , [2]に係数 c , [3]に係数 d を入力して下さい.	daRoots[]	実数解が格納されます. 要素数3以上の配列を用意して下さい.
daCoeffs[]	3次方程式の係数です. [0]に係数 a , [1]に係数 b , [2]に係数 c , [3]に係数 d を入力して下さい.				
daRoots[]	実数解が格納されます. 要素数3以上の配列を用意して下さい.				
戻り値	実数解の個数です.				
留意事項	特にありません.				

eyemMathQuarticRoots

機 能	4次方程式の実数解	
形 式	<pre>#include "eyemLib.h" int eyedMathQuarticRoots (double daCoeffs[], double daRoots[]);</pre>	
解 説	4次方程式 $ax^4 + bx^3 + cx^2 + dx + e = 0$ の実数解を求めます.	
引 数	daCoeffs[]	4次方程式の係数です. [0]に係数 a , [1]に係数 b , [2]に係数 c , [3]に係数 d , [4]に係数 e を 入力して下さい.
	daRoots[]	実数解が格納されます. 要素数4以上の配列を用意して下さい.
戻り値	実数解の個数です.	
留意事項	特にありません.	

eyemMathHorner

機 能	ホーナー法によるn次多項式の計算	
形 式	<pre>#include "eyemLib.h" double eyedMathHorner(int n, double daCoef[], double dX);</pre>	
解 説	<p>n次多項式をホーナー法で計算します. すなわち, 以下のように変形して計算します.</p> $a_0x^n + a_1x^{n-1} + \cdots + a_{n-1}x + a_n = ((\cdots((a_0x + a_1)x + a_2)x + \cdots)x + a_{n-1})x + a_n .$	
引 数	n	多項式の次数です.
	daCoef []	多項式の係数です. [0]に係数 a_0 , [1]に係数 a_1 , \cdots [n-1]に係数 a_{n-1} , [n]に係数 a_n を入力して下さい.
	dX	x 値です.
戻り値	多項式の値です.	
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"> 3関数 (eyemMathOtsuThreshold1d, eyedMathGetDistFromPointToEllipse および eyedMathTransAbcToRq) の追加.
1.2	<ul style="list-style-type: none"> 2関数 (eyemMathMedianI, eyedMathMedianD) の追加.
1.3	<ul style="list-style-type: none"> ホーナー法による多項式計算関数 (eyemMathHorner) の追加