PBblas

Go Up

Table of Contents

Implements Cholesky factorization of A = U**T * U if Triangular.Upper requested or A = L * L**T if Triangualr.Lower is requested
scal.ecl Scale a matrix by a constant Result is alpha * X This supports a ”myriad” style interface in that X may be a set of independent matrices separated by different work-item ids
tran.ecl Transpose a matrix and sum into base matrix
trsm.ecl Partitioned block parallel triangular matrix solver
Types.ecl Types for the Parallel Block Basic Linear Algebra Sub-programs support WARNING: attributes marked with WARNING can not be changed without making corresponding changes to the C++ attributes
Vector2Diag.ecl Convert a vector into a diagonal matrix

PBblas/

Apply2Elements

Go Up

IMPORTS

PBblas | PBblas.Types | std.BLAS |

DESCRIPTIONS

FUNCTION Apply2Elements

DATASET(Layout_Cell)	**Apply2Elements**
(DATASET(Layout_Cell) X, IElementFunc f)	

Apply a function to each element of the matrix Use PBblas.IElementFunc as the prototype function. Input and ouput may be a single matrix, or myriad matrixes with different work item ids.

PARAMETER **X** A matrix (or multiple matrices) in Layout_Cell form

PARAMETER **f** A function based on the IElementFunc prototype

RETURN A matrix (or multiple matrices) in Layout_Cell form

SEE PBblas/IElementFunc

SEE PBblas/Types.Layout_Cell

PBblas/

asum

Go Up

IMPORTS

PBblas | PBblas.Types | PBblas.internal | PBblas.internal.Types | PBblas.internal.MatDims | PBblas.internal.Converted | std.BLAS |

DESCRIPTIONS

FUNCTION asum

DATASET(Layout_Norm)	**asum**
(DATASET(Layout_Cell) X)	

Absolute sum – the "Entrywise" 1-norm Compute SUM(ABS(X))

PARAMETER **X** Matrix or set of matrices in Layout_Cell format

RETURN DATASET(Layout_Norm) with one record per work item

SEE PBblas/Types.Layout_Cell

PBblas/

axpy

Go Up

IMPORTS

PBblas | PBblas.Types |

DESCRIPTIONS

FUNCTION axpy

DATASET(Layout_Cell)	**axpy**
(value_t alpha, DATASET(Layout_Cell) X, DATASET(Layout_Cell) Y)	

Implements alpha*X + Y X and Y must have same shape

PARAMETER **alpha** Scalar multiplier for the X matrix

PARAMETER **X** X matrix in DATASET(Layout_Cell) form

PARAMETER **Y** Y matrix in DATASET(Layout_Call) form

RETURN Matrix in DATASET(Layout_Cell) form

SEE PBblas/Types.layout_cell

Constants

Go Up

DESCRIPTIONS

MODULE Constants

	Constants

Children

1. Block_Minimum
2. Block_NoSplit
3. Block_Maximum
4. Block_Vec_Rows
5. Dimension_Incompat
6. Dimension_IncompatZ
7. Distribution_Error
8. Distribution_ErrorZ
9. Not_Square
10. Not_SquareZ
11. Not_PositiveDef
12. Not_PositiveDefZ
13. Not_Single_Block
14. Not_Single_BlockZ
15. Not_Block_Vector

16. Not_Block_VectorZ

ATTRIBUTE Block_Minimum

Constants \

	Block_Minimum

ATTRIBUTE Block_NoSplit

Constants \

	Block_NoSplit

ATTRIBUTE Block_Maximum

Constants \

	Block_Maximum

ATTRIBUTE Block_Vec_Rows

Constants \

	Block_Vec_Rows

ATTRIBUTE Dimension_Incompat

Constants \

	Dimension_Incompat

ATTRIBUTE Dimension_IncompatZ

Constants \

	Dimension_IncompatZ

ATTRIBUTE Distribution_Error

Constants \

	Distribution_Error

ATTRIBUTE Distribution_ErrorZ

Constants \

	Distribution_ErrorZ

ATTRIBUTE Not_Square

Constants \

	Not_Square

ATTRIBUTE Not_SquareZ

Constants \

	Not_SquareZ

ATTRIBUTE Not_PositiveDef

Constants \

	Not_PositiveDef

ATTRIBUTE Not_PositiveDefZ

Constants \

	Not_PositiveDefZ

ATTRIBUTE Not_Single_Block

Constants \

	Not_Single_Block

ATTRIBUTE Not_Single_BlockZ

Constants \

	Not_Single_BlockZ

ATTRIBUTE Not_Block_Vector

Constants \

	Not_Block_Vector

ATTRIBUTE Not_Block_VectorZ

Constants \

	Not_Block_VectorZ

Converted

Go Up

IMPORTS

PBblas | PBblas.Types | ML_Core.Types |

DESCRIPTIONS

MODULE Converted

	Converted

Module to convert between ML_Core/Types Field layouts (i.e. NumericField and DiscreteField) and PBblas matrix layout (i.e. Layout_Cell)

Children

1. NFToMatrix : Convert NumericField dataset to Matrix
2. DFToMatrix : Convert DiscreteField dataset to Matrix
3. MatrixToNF : Convert Matrix to NumericField dataset
4. MatrixToDF : Convert Matrix to DiscreteField dataset

FUNCTION NFToMatrix

Converted \

DATASET(Layout_Cell)	**NFToMatrix**
(DATASET(NumericField) recs)	

Convert NumericField dataset to Matrix

PARAMETER **recs** Record Dataset in DATASET(NumericField) format

RETURN Matrix in DATASET(Layout_Cell) format

SEE PBblas/Types.Layout_Cell

SEE ML_Core/Types.NumericField

FUNCTION DFToMatrix

Converted \

DATASET(Layout_Cell)	**DFToMatrix**
(DATASET(DiscreteField) recs)	

Convert DiscreteField dataset to Matrix

PARAMETER **recs** Record Dataset in DATASET(DiscreteField) format

RETURN Matrix in DATASET(Layout_Cell) format

SEE PBblas/Types.Layout_Cell

SEE ML_Core/Types.DiscreteField

FUNCTION MatrixToNF

Converted \

DATASET(NumericField)	**MatrixToNF**
(DATASET(Layout_Cell) mat)	

Convert Matrix to NumericField dataset

PARAMETER **mat** Matrix in DATASET(Layout_Cell) format

RETURN NumericField Dataset

SEE PBblas/Types.Layout_Cell

SEE ML_Core/Types.NumericField

FUNCTION MatrixToDF

Converted \

DATASET(DiscreteField)	**MatrixToDF**
(DATASET(Layout_Cell) mat)	

Convert Matrix to DiscreteField dataset

PARAMETER **mat** Matrix in DATASET(Layout_Cell) format

RETURN DiscreteField Dataset

SEE PBblas/Types.Layout_Cell

SEE ML_Core/Types.DiscreteField

PBblas/

ExtractTri

Go Up

IMPORTS

PBblas | std.BLAS | PBblas.Types | PBblas.internal | PBblas.internal.Types | PBblas.internal.MatDims | PBblas.internal.Converted |

DESCRIPTIONS

FUNCTION ExtractTri

DATASET(Layout_Cell)	**ExtractTri**
(Triangle tri, Diagonal dt, DATASET(Layout_Cell) A)	

Extract the upper or lower triangle from the composite output from getrf (LU Factorization).

PARAMETER **tri** Triangle type: Upper or Lower (see Types.Triangle)

PARAMETER **dt** Diagonal type: Unit or non unit (see Types.Diagonal)

PARAMETER **A** Matrix of cells. See Types.Layout_Cell

RETURN Matrix of cells in Layout_Cell format representing a triangular matrix (upper or lower)

SEE Std.PBblas.Types

PBblas/

gemm

Go Up

IMPORTS

PBblas | PBblas.Types | PBblas.internal | PBblas.internal.Types | std.BLAS | PBblas.internal.MatDims | std.system.Thorlib |

DESCRIPTIONS

FUNCTION gemm

DATASET(Layout_Cell)	**gemm**
(BOOLEAN transposeA, BOOLEAN transposeB, value_t alpha, DATASET(Layout_Cell) A_in, DATASET(Layout_Cell) B_in, DATASET(Layout_Cell) C_in=emptyC, value_t beta=0.0)	

Extended Parallel Block Matrix Multiplication Module Implements: Result = alpha * op(A)op(B) + beta * C. op is No Transpose or Transpose. Multiplies two matrixes A and B, with an optional pre-multiply transpose for each Optionally scales the product by the scalar "alpha". Then adds an optional C matrix to the product after scaling C by the scalar "beta". A, B, and C are specified as DATASET(Layout_Cell), as is the Resulting matrix. Layout_Cell describes a sparse matrix stored as a list of x, y, and value. This interface also provides a "Myriad" capability allowing multiple similar operations to be performed on independent sets of matrixes in parallel. This is done by use of the work-item id (wi_id) in each cell of the matrixes. Cells with the same wi_id are considered part of the same matrix. In the myriad form, each input matrix A, B, and (optionally) C can contain many independent matrixes. The wi_ids are matched up such that each operation involves the A, B, and C with the same wi_id. A and B must therefore contain the same set of wi_ids, while C is optional for any wi_id. The same parameters: alpha, beta, transposeA, and transposeB are used for all work-items. The result will contain cells from all provided work-items. Result has same shape as C if provided. Note that matrixes are not explicitly

dimensioned. The shape is determined by the highest value of x and y for each work-item.

PARAMETER **transposeA** Boolean indicating whether matrix A should be transposed before multiplying

PARAMETER **transposeB** Same as above but for matrix B

PARAMETER **alpha** Scalar multiplier for alpha * A * B

PARAMETER **A_in** 'A' matrix (multiplier) in Layout_Cell format

PARAMETER **B_in** Same as above for the 'B' matrix (multiplicand)

PARAMETER **C_in** Same as above for the 'C' matrix (addend). May be omitted.

PARAMETER **beta** A scalar multiplier for beta * C, scales the C matrix before addition. May be omitted.

RETURN Result matrix in Layout_Cell format.

SEE PBblas/Types.Layout_Cell

---

getrf

Go Up

IMPORTS

PBblas | PBblas.Types | PBblas.internal | PBblas.internal.Types | std.BLAS | PBblas.internal.MatDims | std.system.Thorlib |

DESCRIPTIONS

FUNCTION getrf

DATASET(Layout_Cell)	**getrf**
(DATASET(Layout_Cell) A)	

LU Factorization Splits a matrix into Lower and Upper triangular factors Produces composite LU matrix for the diagonal blocks. Iterates through the matrix a row of blocks and column of blocks at a time. Partition A into M block rows and N block columns. The A11 cell is a single block. A12 is a single row of blocks with N-1 columns. A21 is a single column of blocks with M-1 rows. A22 is a sub-matrix of M-1 x N-1 blocks. | A11 A12 | | L11 0 | | U11 U12 | | A21 A22 | == | L21 L22 | * | 0 U22 | | L11*U11 L11*U12 | == | L21*U11 L21*U12 + L22*U22 | Based upon PB-BLAS: A set of parallel block basic linear algebra subprograms by Choi and Dongarra This module supports the ”Myriad” style interface, allowing many independent problems to be worked on at once. The A matrix can contain multiple matrixes to be factored, indicated by different values for work-item id (wi_id). Note: The returned matrix includes both the upper and lower factors. This matrix can be used directly by trsm which will only use the part indicated by trsm’s ’triangle’ parameter (i.e. upper or lower). To extract the upper or lower triangle explicitly for other purposes, use the ExtractTri function. When passing the Lower matrix to the triangle solver (trsm), set the ”Diagonal” parameter to ”UnitTri”. This is necessary because both triangular matrixes returned from this function are packed into a square matrix with only one diagonal. By convention, The Lower triangle is assumed to be a Unit Triangle (diagonal all ones), so the diagonal

contained in the returned matrix is for the Upper factor and must be ignored (i.e. assumed to be all ones) when referencing the Lower triangle.

PARAMETER **A** The input matrix in Types.Layout_Cell format

RETURN Resulting factored matrix in Layout_Cell format

SEE Types.Layout_Cell

SEE ExtractTri

PBblas/

HadamardProduct

Go Up

IMPORTS

PBblas | PBblas.internal | PBblas.internal.MatDims | PBblas.Types | PBblas.internal.Types | PBblas.internal.Converted | std.BLAS | std.system.Thorlib |

DESCRIPTIONS

FUNCTION HadamardProduct

DATASET(Layout_Cell)	**HadamardProduct**
(DATASET(Layout_Cell) X, DATASET(Layout_Cell) Y)	

Element-wise multiplication of X * Y. Supports the "myriad" style interface – X and Y may contain multiple separate matrixes. Each X will be multiplied by the Y with the same work-item id. Note: This performs element-wise multiplication. For dot-product matrix multiplication, use PBblas.gemm.

PARAMETER **X** A matrix (or multiple matrices) in Layout_Cell form

PARAMETER **Y** A matrix (or multiple matrices) in Layout_Cell form

RETURN A matrix (or multiple matrices) in Layout_Cell form

SEE PBblas/Types.Layout_Cell

PBblas/

IElementFunc

Go Up

IMPORTS

PBblas |

DESCRIPTIONS

FUNCTION IElementFunc

value_t	**IElementFunc**
(value_t v, dimension_t r, dimension_t c)	

Function prototype for a function to apply to each element of the distributed matrix Base your function on this prototype:

PARAMETER **v** Input value

PARAMETER **r** Row number (1 based)

PARAMETER **c** Column number (1 based)

RETURN Output value

SEE PBblas/Apply2Elements

PBblas/

MatUtils

Go Up

IMPORTS

PBblas | PBblas.Types | PBblas.internal | PBblas.internal.Types | PBblas.internal.MatDims |

DESCRIPTIONS

MODULE MatUtils

	MatUtils

Provides various utility attributes for manipulating cell-based matrixes

SEE Std/PBblas/Types.Layout_Cell

Children

1. GetWorkItems : Get a list of work-item ids from a matrix containing one or more work items
2. InsertCols : Insert one or more columns of a fixed value into a matrix
3. Transpose : Transpose a matrix This attribute supports the myriad interface

---

FUNCTION GetWorkItems

MatUtils \

DATASET(Layout_WI_ID)	GetWorkItems
(DATASET(Layout_Cell) cells)	

Get a list of work-item ids from a matrix containing one or more work items

PARAMETER **cells** A matrix in Layout_Cell format

RETURN DATASET(Layout_WI_ID), one record per work-item

SEE PBblas/Types.Layout_Cell

SEE PBblas/Types.Layout_WI_ID

FUNCTION InsertCols

MatUtils \

DATASET(Layout_Cell)	InsertCols
(DATASET(Layout_Cell) M, UNSIGNED cols_to_insert=1, value_t insert_val=1)	

Insert one or more columns of a fixed value into a matrix. Columns are inserted before the first original column. This attribute supports the myriad interface. Multiple independent matrixes can be represented by M.

PARAMETER **M** the input matrix

PARAMETER **cols_to_insert** the number of columns to insert, default 1

PARAMETER **insert_val** the value for each cell of the new column(s), default 0

RETURN matrix in Layout_Cell format with additional column(s)

FUNCTION Transpose

MatUtils \

DATASET(Layout_Cell)	**Transpose**
(DATASET(Layout_Cell) M)	

Transpose a matrix This attribute supports the myriad interface. Multiple independent matrixes can be represented by M.

PARAMETER **M** A matrix represented as DATASET(Layout_Cell)

RETURN Transposed matrix in Layout_Cell format

SEE PBblas/Types.Layout_Cell

PBblas/

potrf

Go Up

IMPORTS

PBblas | PBblas.Types | std.BLAS | PBblas.internal | PBblas.internal.Types | PBblas.internal.MatDims | PBblas.internal.Converted | std.system.Thorlib |

DESCRIPTIONS

FUNCTION potrf

DATASET(Layout_Cell)	**potrf**
(Triangle tri, DATASET(Layout_Cell) A_in)	

Implements Cholesky factorization of A = U**T * U if Triangular.Upper requested or A = L * L**T if Triangualr.Lower is requested. The matrix A must be symmetric positive definite.

```
| A11   A12 |      |  L11   0   |    | L11**T     L21**T |
| A21   A22 |  ==  |  L21   L22 | *  |  0           L22  |

                   | L11*L11**T          L11*L21**T      |
               ==  | L21*L11**T  L21*L21**T + L22*L22**T |
```

So, use Cholesky on the first block to get L11. L21 = A21*L11**T**-1 which can be found by dtrsm on each column block A22' is A22 - L21*L21**T

Based upon PB-BLAS: A set of parallel block basic linear algebra subprograms by Choi and Dongarra

This module supports the ”Myriad” style interface, allowing many independent problems to be worked on at once. The A matrix can contain multiple matrixes to be factored, indicated by different values for work-item id (wi_id).

PARAMETER **tri** Types.Triangle enumeration indicating whether we are looking for the Upper or the Lower factor

PARAMETER **A_in** The matrix or matrixes to be factored in Types.Layout_Cell format

RETURN Triangular matrix in Layout_Cell format

SEE Std.PBblas.Types.Layout_Cell

SEE Std.PBblas.Types.Triangle

PBblas/

scal

Go Up

IMPORTS

PBblas | PBblas.Types |

DESCRIPTIONS

FUNCTION scal

DATASET(Layout_Cell)	scal
(value_t alpha, DATASET(Layout_Cell) X)	

Scale a matrix by a constant Result is alpha * X This supports a ”myriad” style interface in that X may be a set of independent matrices separated by different work-item ids.

PARAMETER **alpha** A scalar multiplier

PARAMETER **X** The matrix(es) to be scaled in Layout_Cell format

RETURN Matrix in Layout_Cell form, of the same shape as X

SEE PBblas/Types.Layout_Cell

PBblas/

tran

Go Up

IMPORTS

PBblas | PBblas.Types | PBblas.internal | PBblas.internal.Types | PBblas.internal.MatDims | PBblas.internal.Converted | std.BLAS | std.system.Thorlib |

DESCRIPTIONS

FUNCTION tran

DATASET(Layout_Cell)	**tran**
(value_t alpha, DATASET(Layout_Cell) A, value_t beta=0, DATASET(Layout_Cell) C=empty_c)	

Transpose a matrix and sum into base matrix result <== alpha * A**t + beta * C, A is n by m, C is m by n A**T (A Transpose) and C must have same shape

PARAMETER **alpha** Scalar multiplier for the A**T matrix

PARAMETER **A** A matrix in DATASET(Layout_Cell) form

PARAMETER **beta** Scalar multiplier for the C matrix

PARAMETER **C** C matrix in DATASET(Layout_Call) form

RETURN Matrix in DATASET(Layout_Cell) form alpha * A**T + beta * C

SEE PBblas/Types.layout_cell

trsm

Go Up

IMPORTS

PBblas | PBblas.Types | std.BLAS | PBblas.internal | PBblas.internal.Types | PBblas.internal.MatDims | PBblas.internal.Converted | std.system.Thorlib |

DESCRIPTIONS

FUNCTION trsm

DATASET(Layout_Cell)	**trsm**
(Side s, Triangle tri, BOOLEAN transposeA, Diagonal diag, value_t alpha, DATASET(Layout_Cell) A_in, DATASET(Layout_Cell) B_in)	

Partitioned block parallel triangular matrix solver. Solves for X using: AX = B or XA = B A is is a square triangular matrix, X and B have the same dimensions. A may be an upper triangular matrix (UX = B or XU = B), or a lower triangular matrix (LX = B or XL = B). Allows optional transposing and scaling of A. Partially based upon an approach discussed by MJ DAYDE, IS DUFF, AP CERFACS. A Parallel Block implementation of Level-3 BLAS for MIMD Vector Processors ACM Tran. Mathematical Software, Vol 20, No 2, June 1994 pp 178-193 and other papers about PB-BLAS by Choi and Dongarra This module supports the "Myriad" style interface, allowing many independent problems to be worked on at once. Corresponding A and B matrixes are related by a common work-item identifier (wi_id) within each cell of the matrix. The returned X matrix will contain cells for the same set of work-items as specified for the A and B matrices.

PARAMETER **s** Types.Side enumeration indicating whether we are solving AX = B or XA = B

PARAMETER **tri** Types.Triangle enumeration indicating whether we are solving an Upper or Lower triangle.

PARAMETER **transposeA** Boolean indicating whether or not to transpose the A matrix before solving

PARAMETER **diag** Types.Diagonal enumeration indicating whether A is a unit matrix or not. This is primarily used after factoring matrixes using getrf (LU factorization). That module produces a factored matrix stored within the same space as the original matrix. Since the diagonal is used by both factors, by convention, the Lower triangle has a unit matrix (diagonal all 1's) while the Upper triangle uses the diagonal cells. Setting this to UnitTri, causes the contents of the diagonal to be ignored, and assumed to be 1. NotUnitTri should be used for most other cases.

PARAMETER **alpha** Multiplier to scale A

PARAMETER **A_in** The A matrix in Layout_Cell format

PARAMETER **B_in** The B matrix in Layout_Cell format

RETURN X solution matrix in Layout_Cell format

SEE Types.Layout_Cell

SEE Types.Triangle

SEE Types.Side

PBblas/

Types

Go Up

IMPORTS

ML_Core | ML_Core.Types |

DESCRIPTIONS

MODULE Types

	Types

Types for the Parallel Block Basic Linear Algebra Sub-programs support WARNING: attributes marked with WARNING can not be changed without making corresponding changes to the C++ attributes.

Children

1. dimension_t : Type for matrix dimensions
2. partition_t : Type for partition id – only supports up to 64K partitions
3. work_item_t : Type for work-item id – only supports up to 64K work items
4. value_t : Type for matrix cell values
5. m_label_t : Type for matrix label
6. Triangle : Enumeration for Triangle type
7. Diagonal : Enumeration for Diagonal type
8. Side : Enumeration for Side type

9. t_mu_no : Type for matrix universe number
10. Layout_Cell : Layout for Matrix Cell Main representation of Matrix cell at interface to all PBBlas functions
11. Layout_Norm : Layout for Norm results

ATTRIBUTE dimension_t

Types \

	dimension_t

Type for matrix dimensions. Uses UNSIGNED four as matrixes are not designed to support more than 4 B rows or columns.

ATTRIBUTE partition_t

Types \

	partition_t

Type for partition id – only supports up to 64K partitions

ATTRIBUTE work_item_t

Types \

	work_item_t

Type for work-item id – only supports up to 64K work items

ATTRIBUTE value_t

Types \

	value_t

Type for matrix cell values WARNING: type used in C++ attribute

---

ATTRIBUTE m_label_t

Types \

	m_label_t

Type for matrix label. Used for Matrix dimensions (see Layout_Dims) and for partitions (see Layout_Part)

ATTRIBUTE Triangle

Types \

	Triangle

Enumeration for Triangle type WARNING: type used in C++ attribute

---

ATTRIBUTE Diagonal

Types \

	Diagonal

Enumeration for Diagonal type WARNING: type used in C++ attribute

ATTRIBUTE Side

Types \

	Side

Enumeration for Side type WARNING: type used in C++ attribute

---

ATTRIBUTE t_mu_no

Types \

	t_mu_no

Type for matrix universe number Allow up to 64k matrices in one universe

---

RECORD Layout_Cell

Types \

	Layout_Cell

Layout for Matrix Cell Main representation of Matrix cell at interface to all PBBlas functions. Matrixes are represented as DATASET(Layout_Cell), where each cell describes the row and column position of the cell as well as its value. Only the non-zero cells need to be contained in the dataset in order to describe the matrix since all unspecified cells are considered to have a value of zero. The cell also contains a work-item number that allows multiple separate matrixes to be carried in the same dataset. This supports the "myriad" style interface that allows the same operations to be performed on many different sets of data at once. Note that these matrixes do not have an explicit size. They are sized implicitly, based on the maximum row and column presented in the data. A matrix can be converted to an explicit dense form (see matrix_t) by using the utility module MakeR8Set. This module should only be used for known small matrixes (< 1M cells) or for partitions of a larger matrix. The Converted module provides utility functions to convert to and from a set of partitions (See Layout_parts).

FIELD **wi_id** Work Item Number – An identifier from 1 to 64K-1 that separates and identifies individual matrixes

FIELD **x** 1-based row position within the matrix

FIELD **y** 1-based column position within the matrix

FIELD **v** Real value for the cell

SEE matrix_t

SEE Std/PBblas/MakeR8Set.ecl

SEE Std/PBblas/Converted.ecl WARNING: Used as C++ attribute. Do not change without corresponding changes to MakeR8Set.

RECORD Layout_Norm

Types \

	Layout_Norm

Layout for Norm results.

FIELD **wi_id** Work Item Number – An identifier from 1 to 64K-1 that separates and identifies individual matrixes

FIELD **v** Real value for the norm

PBblas/

Vector2Diag

Go Up

IMPORTS

PBblas | PBblas.internal | PBblas.internal.MatDims | PBblas.Types | PBblas.internal.Types | PBblas.Constants |

DESCRIPTIONS

FUNCTION Vector2Diag

DATASET(Layout_Cell)	**Vector2Diag**
(DATASET(Layout_Cell) X)	

Convert a vector into a diagonal matrix. The typical notation is D = diag(V). The input X must be a 1 x N column vector or an N x 1 row vector. The resulting matrix, in either case will be N x N, with zero everywhere except the diagonal.

PARAMETER **X** A row or column vector (i.e. N x 1 or 1 x N) in Layout_Cell format

RETURN An N x N matrix in Layout_Cell format

SEE PBblas/Types.Layout_cell
