

PBblas

[Go Up](#)

Name	PBblas
Version	3.0.1
Description	Parallel Block Basic Linear Algebra Subsystem
License	http://www.apache.org/licenses/LICENSE-2.0
Copyright	Copyright (C) 2016, 2017 HPCC Systems
Authors	HPCCSystems
DependsOn	ML_Core
Platform	6.2.0

Table of Contents

Apply2Elements.ecl
Apply a function to each element of the matrix Use PBblas.IElementFunc as the prototype function
asum.ecl
Absolute sum – the "Entrywise" 1-norm
axpy.ecl
Implements $\alpha * X + Y$
Constants.ecl
Converted.ecl
Module to convert between ML_Core/Types Field layouts (i.e
ExtractTri.ecl
Extract the upper or lower triangle from the composite output from getrf (LU Factorization)
gemm.ecl
Extended Parallel Block Matrix Multiplication Module Implements: $Result = \alpha * op(A)op(B) + \beta * C$
getrf.ecl
LU Factorization Splits a matrix into Lower and Upper triangular factors Produces composite LU matrix for the diagonal blocks
HadamardProduct.ecl

Element-wise multiplication of $X * Y$
IElementFunc.ecl Function prototype for a function to apply to each element of the
MatUtils.ecl Provides various utility attributes for manipulating cell-based matrixes
potrf.ecl 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 output may be a single matrix, or myriad matrixes with different work item ids.

PARAMETER **f** ||| FUNCTION [REAL8 , UNSIGNED4 , UNSIGNED4] (REAL8) — A function based on the IElementFunc prototype

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

RETURN TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v }) — 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 $\text{SUM}(\text{ABS}(X))$

PARAMETER X ||| TABLE (Layout_Cell) — Matrix or set of matrices in Layout_Cell format

RETURN TABLE ({ UNSIGNED2 wi_id , REAL8 v }) — DATASET(Layout_Norm) with one record per work item

SEE PBblas/Types.Layout_Cell

[Go Up](#)

IMPORTS

PBblas | PBblas.Types |

DESCRIPTIONS

FUNCTION axpy

<code>DATASET(Layout_Cell)</code>	<code>axpy</code>
<code>(value_t alpha, DATASET(Layout_Cell) X, DATASET(Layout_Cell) Y)</code>	

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

PARAMETER alpha ||| REAL8 — Scalar multiplier for the X matrix

PARAMETER X ||| TABLE (Layout_Cell) — X matrix in DATASET(Layout_Cell) form

PARAMETER Y ||| TABLE (Layout_Cell) — Y matrix in DATASET(Layout_Cell) form

RETURN TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v }
) — Matrix in DATASET(Layout_Cell) form

SEE PBblas/Types.layout_cell

PBblas/ Constants

[Go Up](#)

DESCRIPTIONS

MODULE Constants

Constants

No Documentation Found

Children

1. [Block_Minimum](#) : No Documentation Found
2. [Block_NoSplit](#) : No Documentation Found
3. [Block_Maximum](#) : No Documentation Found
4. [Block_Vec_Rows](#) : No Documentation Found
5. [Dimension_Incompat](#) : No Documentation Found
6. [Dimension_IncompatZ](#) : No Documentation Found
7. [Distribution_Error](#) : No Documentation Found
8. [Distribution_ErrorZ](#) : No Documentation Found
9. [Not_Square](#) : No Documentation Found
10. [Not_SquareZ](#) : No Documentation Found
11. [Not_PositiveDef](#) : No Documentation Found
12. [Not_PositiveDefZ](#) : No Documentation Found
13. [Not_Single_Block](#) : No Documentation Found
14. [Not_Single_BlockZ](#) : No Documentation Found

- 15. [Not_Block_Vector](#) : No Documentation Found
- 16. [Not_Block_VectorZ](#) : No Documentation Found

ATTRIBUTE Block_Minimum

[Constants](#) \

	Block_Minimum
--	---------------

No Documentation Found

RETURN INTEGER8 —

ATTRIBUTE Block_NoSplit

[Constants](#) \

	Block_NoSplit
--	---------------

No Documentation Found

RETURN INTEGER8 —

ATTRIBUTE Block_Maximum

[Constants](#) \

	Block_Maximum
--	---------------

No Documentation Found

RETURN INTEGER8 —

ATTRIBUTE Block_Vec_Rows

[Constants](#) \

	Block_Vec_Rows
--	----------------

No Documentation Found

RETURN INTEGER8 —

ATTRIBUTE Dimension_Incompat

[Constants](#) \

	Dimension_Incompat
--	--------------------

No Documentation Found

RETURN STRING34 —

ATTRIBUTE Dimension_IncompatZ

[Constants](#) \

	Dimension_IncompatZ
--	---------------------

No Documentation Found

RETURN INTEGER8 —

ATTRIBUTE Distribution_Error

[Constants](#) \

	Distribution_Error
--	--------------------

No Documentation Found

RETURN STRING32 —

ATTRIBUTE Distribution_ErrorZ

[Constants](#) \

	Distribution_ErrorZ
--	---------------------

No Documentation Found

RETURN INTEGER8 —

ATTRIBUTE Not_Square

[Constants](#) \

	Not_Square
--	------------

No Documentation Found

RETURN STRING20 —

ATTRIBUTE Not_SquareZ

[Constants](#) \

	Not_SquareZ
--	-------------

No Documentation Found

RETURN INTEGER8 —

ATTRIBUTE Not_PositiveDef

[Constants](#) \

	Not_PositiveDef
--	-----------------

No Documentation Found

RETURN STRING40 —

ATTRIBUTE Not_PositiveDefZ

[Constants](#) \

	Not_PositiveDefZ
--	------------------

No Documentation Found

RETURN INTEGER8 —

ATTRIBUTE Not_Single_Block

[Constants](#) \

	Not_Single_Block
--	------------------

No Documentation Found

RETURN STRING28 —

ATTRIBUTE Not_Single_BlockZ

[Constants](#) \

	Not_Single_BlockZ
--	-------------------

No Documentation Found

RETURN INTEGER8 —

ATTRIBUTE Not_Block_Vector

[Constants](#) \

	Not_Block_Vector
--	------------------

No Documentation Found

RETURN STRING25 —

ATTRIBUTE Not_Block_VectorZ

[Constants](#) \

	Not_Block_VectorZ
--	-------------------

No Documentation Found

RETURN INTEGER8 —

PBblas/ 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** ||| TABLE (NumericField) — Record Dataset in DATASET(NumericField) format

RETURN TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v }) — 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** ||| TABLE (DiscreteField) — Record Dataset in DATASET(DiscreteField) format

RETURN TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v }) — 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 ||| TABLE (Layout_Cell) — Matrix in DATASET(Layout_Cell) format

RETURN TABLE ({ UNSIGNED2 wi , UNSIGNED8 id , UNSIGNED4 number , REAL8 value }) — 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 ||| TABLE (Layout_Cell) — Matrix in DATASET(Layout_Cell) format

RETURN TABLE ({ UNSIGNED2 wi , UNSIGNED8 id , UNSIGNED4 number , INTEGER4 value }) — 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** ||| UNSIGNED1 — Triangle type: Upper or Lower (see Types.Triangle)

PARAMETER **A** ||| TABLE (Layout_Cell) — Matrix of cells. See Types.Layout_Cell

PARAMETER **dt** ||| UNSIGNED1 — Diagonal type: Unit or non unit (see Types.Diagonal)

RETURN TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v }
) — 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`

<code>DATASET(Layout_Cell)</code>	<code>gemm</code>
<code>(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)</code>	

Extended Parallel Block Matrix Multiplication Module Implements: $\text{Result} = \alpha * \text{op}(\text{A})\text{op}(\text{B}) + \beta * \text{C}$. 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 alpha ||| REAL8 — Scalar multiplier for $\alpha * A * B$

PARAMETER A_in ||| TABLE (Layout_Cell) — 'A' matrix (multiplier) in Layout_Cell format

PARAMETER beta ||| REAL8 — A scalar multiplier for $\beta * C$, scales the C matrix before addition. May be omitted.

PARAMETER B_in ||| TABLE (Layout_Cell) — Same as above for the 'B' matrix (multiplicand)

PARAMETER transposeA ||| BOOLEAN — Boolean indicating whether matrix A should be transposed before multiplying

PARAMETER C_in ||| TABLE (Layout_Cell) — Same as above for the 'C' matrix (addend). May be omitted.

PARAMETER transposeB ||| BOOLEAN — Same as above but for matrix B

RETURN TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v }) — Result matrix in Layout_Cell format.

SEE PBblas/Types.Layout_Cell

PBblas/ getrf

[Go Up](#)

IMPORTS

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

DESCRIPTIONS

FUNCTION `getrf`

<code>DATASET(Layout_Cell)</code>	<code>getrf</code>
<code>(DATASET(Layout_Cell) A)</code>	

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** ||| TABLE (Layout_Cell) — The input matrix in Types.Layout_Cell format

RETURN TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v }
) — 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 ||| TABLE (Layout_Cell) — A matrix (or multiple matrices) in Layout_Cell form

PARAMETER Y ||| TABLE (Layout_Cell) — A matrix (or multiple matrices) in Layout_Cell form

RETURN TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v }
) — A matrix (or multiple matrices) in Layout_Cell form

SEE PBblas/Types.Layout_Cell

PBblas/ IElementFunc

[Go Up](#)

IMPORTS

PBblas |

DESCRIPTIONS

FUNCTION IElementFunc

<code>value_t</code>	IElementFunc
<code>(value_t v, dimension_t r, dimension_t c)</code>	

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

PARAMETER `r` ||| UNSIGNED4 — Row number (1 based)

PARAMETER `c` ||| UNSIGNED4 — Column number (1 based)

PARAMETER `v` ||| REAL8 — Input value

RETURN REAL8 — 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 ||| TABLE (Layout_Cell) — A matrix in Layout_Cell format

RETURN TABLE ({ UNSIGNED2 wi_id }) — 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 cols_to_insert ||| UNSIGNED8 — the number of columns to insert, default 1

PARAMETER insert_val ||| REAL8 — the value for each cell of the new column(s), default 0

PARAMETER M ||| TABLE (Layout_Cell) — the input matrix

RETURN TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v }) — matrix in Layout_Cell format with additional column(s)

FUNCTION Transpose

MatUtils \

<code>DATASET(Layout_Cell)</code>	Transpose
<code>(DATASET(Layout_Cell) M)</code>	

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

PARAMETER M ||| TABLE (Layout_Cell) — A matrix represented as DATASET(Layout_Cell)

RETURN TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v }
) — 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

<code>DATASET(Layout_Cell)</code>	<code>potrf</code>
<code>(Triangle tri, DATASET(Layout_Cell) A_in)</code>	

Implements Cholesky factorization of $A = U^{**T} * U$ if Triangulr.Upper requested or $A = L * L^{**T}$ if Triangulr.Lower is requested. The matrix A must be symmetric positive definite.

$$\begin{array}{|cc|} \hline A11 & A12 \\ \hline A21 & A22 \\ \hline \end{array} == \begin{array}{|cc|} \hline L11 & 0 \\ \hline L21 & L22 \\ \hline \end{array} * \begin{array}{|cc|} \hline L11^{**T} & L21^{**T} \\ \hline 0 & L22 \\ \hline \end{array}$$
$$== \begin{array}{|cc|} \hline L11 * L11^{**T} & L11 * L21^{**T} \\ \hline L21 * L11^{**T} & L21 * L21^{**T} + L22 * L22^{**T} \\ \hline \end{array}$$

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 ||| UNSIGNED1 — Types.Triangle enumeration indicating whether we are looking for the Upper or the Lower factor

PARAMETER A_in ||| TABLE (Layout_Cell) — The matrix or matrixes to be factored in Types.Layout_Cell format

RETURN TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v }) — 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

<code>DATASET(Layout_Cell)</code>	<code>scal</code>
<code>(value_t alpha, DATASET(Layout_Cell) X)</code>	

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 ||| REAL8 — A scalar multiplier

PARAMETER X ||| TABLE (Layout_Cell) — The matrix(es) to be scaled in Layout_Cell format

RETURN TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v }
) — 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

<code>DATASET(Layout_Cell)</code>	tran
<code>(value_t alpha, DATASET(Layout_Cell) A, value_t beta=0, DATASET(Layout_Cell) C=empty_c)</code>	

Transpose a matrix and sum into base matrix result $\leq \alpha * A^{**t} + \beta * C$, A is n by m, C is m by n A^{**T} (A Transpose) and C must have same shape

PARAMETER alpha ||| REAL8 — Scalar multiplier for the A^{**T} matrix

PARAMETER C ||| TABLE (Layout_Cell) — C matrix in DATASET(Layout_Call) form

PARAMETER beta ||| REAL8 — Scalar multiplier for the C matrix

PARAMETER A ||| TABLE (Layout_Cell) — A matrix in DATASET(Layout_Cell) form

RETURN TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v }
) — Matrix in DATASET(Layout_Cell) form $\alpha * A^{**T} + \beta * C$

SEE PBblas/Types.layout_cell

PBblas/ 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**

<code>DATASET(Layout_Cell)</code>	trsm
<code>(Side s, Triangle tri, BOOLEAN transposeA, Diagonal diag, value_t alpha, DATASET(Layout_Cell) A_in, DATASET(Layout_Cell) B_in)</code>	

Partitioned block parallel triangular matrix solver. Solves for X using: $AX = B$ or $XA = B$ A 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 alpha ||| REAL8 — Multiplier to scale A

PARAMETER A_in ||| TABLE (Layout_Cell) — The A matrix in Layout_Cell format

PARAMETER diag ||| UNSIGNED1 — 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 B_in ||| TABLE (Layout_Cell) — The B matrix in Layout_Cell format

PARAMETER transposeA ||| BOOLEAN — Boolean indicating whether or not to transpose the A matrix before solving

PARAMETER tri ||| UNSIGNED1 — Types.Triangle enumeration indicating whether we are solving an Upper or Lower triangle.

PARAMETER s ||| UNSIGNED1 — Types.Side enumeration indicating whether we are solving $AX = B$ or $XA = B$

RETURN TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v }) — 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 \

<code>dimension_t</code>

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

RETURN UNSIGNED4 —

ATTRIBUTE `partition_t`

Types \

<code>partition_t</code>

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

RETURN UNSIGNED2 —

ATTRIBUTE `work_item_t`

Types \

	<code>work_item_t</code>
--	--------------------------

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

RETURN UNSIGNED2 —

ATTRIBUTE `value_t`

[Types \](#)

	<code>value_t</code>
--	----------------------

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

RETURN REAL8 —

ATTRIBUTE `m_label_t`

[Types \](#)

	<code>m_label_t</code>
--	------------------------

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

RETURN STRING3 —

ATTRIBUTE `Triangle`

[Types \](#)

Triangle

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

RETURN UNSIGNED1 —

ATTRIBUTE Diagonal

[Types \](#)

Diagonal

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

RETURN UNSIGNED1 —

ATTRIBUTE Side

[Types \](#)

Side

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

RETURN UNSIGNED1 —

ATTRIBUTE t_mu_no

[Types \](#)

<code>t_mu_no</code>

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

RETURN UNSIGNED2 —

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 x ||| UNSIGNED4 — 1-based row position within the matrix

FIELD v ||| REAL8 — Real value for the cell

FIELD wi_id ||| UNSIGNED2 — Work Item Number – An identifier from 1 to 64K-1 that separates and identifies individual matrixes

FIELD y ||| UNSIGNED4 — 1-based column position within the matrix

SEE matrix_t

SEE Std/PBBblas/MakeR8Set.ecl

SEE Std/PBBblas/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 ||| UNSIGNED2 — Work Item Number – An identifier from 1 to 64K-1 that separates and identifies individual matrixes

FIELD v ||| REAL8 — 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 $D = \text{diag}(V)$. 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** ||| TABLE (Layout_Cell) — A row or column vector (i.e. N x 1 or 1 x N) in Layout_Cell format

RETURN TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v }) — An N x N matrix in Layout_Cell format

SEE PBblas/Types.Layout_cell
