

MTX - 3.4

C ANSI/C++ BLAS-3 API

Developed by Eng. Juan Camilo Gómez Cadavid MSc.

Macro/Function/Keyword	Description	MATLAB Eq.
Generation		
matrix VAR=NULL mtxdef (VAR)	Matrix MTX type definition Keyword. matrix keyword, defines unallocated variables. Always initialize matrix-type MTX variables to NULL Use mtxdef to define an allocated empty MATRIX by default.	VAR=[];
M=mtx_new (n,m)	Returns a matrix M of $m \times n$ dimensions (initialized on 0)	M=zeros(n,m)
M=mtx_init (n,m,initval)	Returns a matrix M of $m \times n$ dimensions initialized on <i>initval</i>	M=initval*ones(n,m)
M=mtx_rand (n,m)	Returns a matrix M of $m \times n$ dimensions containing pseudorandom values drawn from the standard uniform distribution on the open interval (0,1)	M=rand(n,m)
M=mtx_1dtomtx (A)	Allocates a MTX matrix from the 1D array A	
M=mtx_2dtomtx (A)	Allocates a MTX matrix from the 2D array A	
M=mtx_eye (n,alpha)	Returns the $n \times n$ identity matrix $M = \alpha I$. α is scalar and I represents the identity matrix	M=alpha*eye(n)
mtx_del (M)	Releases the specified block of memory generated by M, back to the heap	clear M
M=mtx_cpy (A)	Generates a copy of matrix A into M .	M=A
mtx_memcpy (M,A)	Generates a copy of matrix A into M . (M is allocated, and has the same dimensions of A)	M=A
Indexing		
x=A->pos [r][c]	Get the element at index <i>row-r</i> and <i>column-c</i>	x=A(r,c)
A->pos [r][c] = value	Set a single element at index <i>row-r</i> and <i>column-c</i>	A(r,c)=value
M=mtx_submatget (A,r1,r2,c1,c2)	Get submatrix $M \leftarrow A$ (M from A), from rows $r1$ to $r2$ and columns $c1$ to $c2$	A(r1:r2,c1:c2)
M=mtx_getrow (A,r)	Get the r -row from matrix A	M=A(r,:)
M=mtx_getcol (A,c)	Get the c -column from matrix A	M=A(:,c)
M=mtx_getrows (A,r1,r2)	Get rows from matrix A . From rows $r1$ to $r2$	M=A(r1:r2,:)
M=mtx_getcols (A,c1,c2)	Get columns from matrix A . From columns $c1$ to $c2$	M=A(:,c1:c2)
mxt_submatset (M,A,r1,r2,c1,c2)	Put submatrix A into M from row $f1$ to row $f2$ and cols $c1$ to col $c2$	M(r1:r2,c1:c2)=A

mtx_setrow(M,A,r)	Set the r -row from matrix M , with the row vector A	$M(r,:)=A$
mtx_setcol(M,A,c)	Set the c -column from matrix M , with the column vector A	$M(:,c)=A$
Basic information		
x=mtx_isempty(M)	TRUE if M is an empty matrix	isempty(M)
x=mtx_isrow(M)	TRUE if M is a row vector	isrow(M)
x=mtx_iscolumn(M)	TRUE if M is a column vector	iscolumn(M)
x=mtx_isvector(M)	TRUE if M is vector	isvector(M)
x=mtx_numel(M)	Number of elements in matrix M	x=numel(M)
x=mtx_length(M)	Largest matrix dimension of M	x=length(M)
x=mtx_det(M)	Determinant of matrix M	x=det(M)
x=mtx_trace(M)	Trace of matrix M	x=trace(M)
D=mtx_diag(M)	Returns a column vector with all diagonal elements of M	D=diag(M)
M=mtx_produ(A)	Product of matrix elements - by columns	M=prod(A)
x=mtx_cumprod(A)	Total product of matrix elements	x=prod(A(:))
M=mtx_sum(A)	Sum of matrix elements - by columns	M=sum(A)
x=mtx_cumsum(A)	Total sum of matrix elements	x=sum(A(:))
M=mtx_mean(A)	Mean of matrix elements – by columns	M=mean(A)
M=mtx_max(A)	If A is a vector, returns the largest element in A . If A is a matrix, treats the columns of A as vectors, returning a row vector containing the maximum element from each column.	M=max(A)
x=mtx_cummax(A)	Largest element in matrix	x=max(A(:))
M=mtx_min(A)	If A is a vector, returns the smallest element in A . If A is a matrix, treats the columns of A as vectors, returning a row vector containing the smallest element from each column.	M=min(A)
x=mtx_cummin(A)	Smallest element in matrix	x=min(A(:))
BLAS-Operations		
M=mtx_t(A)	Returns the transpose of A	M=A'
M=mtx_gadd(alpha,A,beta,B)	Returns $M=\alpha A + \beta B$ α and β are scalars, and A and B matrices.	M=alpha*A+beta*B
M=mtx_prod(alpha,A,B)	Returns matrix Multiplication $M=\alpha AB$	M=alpha*A*B

ret=mtx_dgemm(alpha,A,TA,B,TB,beta,C)	α is scalar, A and B matrices. Computes $C = \alpha AB + \beta C$ and related operations. α and β are scalars, and A , B and C are matrices. $TA = \text{'T' or 't'}$ $\rightarrow A$ is considered transposed (A^T). $TB = \text{'T' or 't'}$ $\rightarrow B$ is considered transposed (B^T). On success, returns (0) and matrix C is overwritten, otherwise returns (-1).	$C = \alpha A * B + \beta * C$ $C = \alpha A' * B + \beta * C$ $C = \alpha A * B' + \beta * C$ $C = \alpha A' * B' + \beta * C$
M=mtx_powui(A,n)	Returns A^n (Matrix power - n must be an unsigned int and A an square matrix)	$M=A^n$
M=mtx_ptpprod(A,B)	Returns $A.B$ (point by point product) (A and B must have the same dimensions)	$M=A.*B$
M=mtx_koper(A,Op,beta)	Returns standard scalar matrix operations. $C = A(Op)\beta$ Related operations (Op) = '+', '-', '*', '/' and '^' (power) Example: M=mtx_koper(A,'+',beta);	$M=A+\beta$ $M=A-\beta$ $M=A*\beta$ $M=A/\beta$ $M=A^\beta$
M=mtx_inv(A)	Returns A^{-1} (Matrix inverse)	$M=\text{inv}(A)$
M=mtx_linsolve(A,B)	Returns $M= A^{-1}B$	$M=\text{inv}(A)*B$ $M=A\backslash B$
mtx_lu(L,U,A)	Expresses the matrix A , as the product of two essentially triangular matrices, one of them, a permutation of a lower triangular matrix and the other an upper triangular matrix. (L and U , must be allocated previously and must have the same dimensions of A)	$[L,U] = \text{lu}(A)$
Q=mtx_qr(A,R)	Orthogonal-triangular decomposition. (R is allocated previously and must have the same dimensions of A)	$[Q,R]=\text{qr}(A)$
M=mtx_rpinv(A,tol)	Returns the Moore-Penrose right pseudoinverse (tol = tolerance)	
M=mtx_lpinv(A,tol)	Returns the Moore-Penrose left pseudoinverse (tol = tolerance)	
M=mtx_expm(A, alpha)	Returns matrix exponential $e^{\alpha A}$. mtx_expm uses the Padé approximation with scaling and squaring.	$M=\text{expm}(\alpha A)$
M=mtx_fxptp(A, fx)	Returns function fx evaluated for each element of matrix A Examples: M=mtx_fxptp(A, sin); \rightarrow Element-wise sine M=mtx_fxptp(A, fabs); \rightarrow Element-wise absolute value	$M=\text{fx}(A)$
Manipulation		
M=mtx_vcat(A,B)	Returns $[A ; B]$ (Vertical concatenation - append by columns)	$M=[A;B]$

M=mtx_hcat(A,B)	Returns $[A \text{ , } B]$ (Horizontal concatenation - append by rows)	M=[A,B]
M=mtx_vcatn(A,B,C,...,X,Y,Z,NULL)	Return $[A \text{ ; } B \text{ ; } C; \dots \text{ , } X \text{ ; } Y \text{ ; } Z]$ (Vertical concatenation - append by columns)	M=[A;B;C;...;X;Y;Z]
M=mtx_hcatn(A,B,C,...,X,Y,Z,NULL)	Return $[A \text{ , } B \text{ , } C, \dots \text{ , } X \text{ , } Y \text{ , } Z]$ (Horizontal concatenation - append by rows)	M=[A,B,C,...,X,Y,Z]
Visualization		
mtx_disp(A)	Display matrix A	disp(A)
mtx_show(A)	Display matrix with variable name	