

# Function index

In the descriptions below, matrices are represented by capital letters, vectors by lower case letters and scalars by greek lower case letters.

Function	Description	Page
<code>_add_()</code>	Add arrays	69
<code>arnoldi()</code>	Arnoldi routine	119
<code>bisvd()</code>	SVD of bi-diagonal matrix	91
<code>BKPfactor()</code>	Bunch–Kaufman–Parlett factorisation	71
<code>BKPsolve()</code>	Bunch–Kaufman–Parlett solver	71
<code>catch()</code>	Catch a raised error (macro)	34
<code>catchall()</code>	Catch any raised error (macro)	34
<code>catch_FPE()</code>	Catch floating point error (sets flag)	34
<code>cgs()</code>	CGS algorithm of Sonneveld	113
<code>cg_set_maxiter()</code>	Sets maximum number of iterations for iterative conjugate-gradient based routines	113
<code>CHfactor()</code>	Dense Cholesky factorisation	73
<code>CHsolve()</code>	Cholesky solver	73
<code>cp_ivec()</code>	Copy integer vector	36
<code>cp_mat()</code>	Copy dense matrix	36
<code>cp_perm()</code>	Copy permutation	36
<code>cp_vec()</code>	Copy vector	36
<code>Dsolve()</code>	Solve $Dx = y$ , $D$ diagonal	84
<code>dump_mat()</code>	Dump matrix data structure to a stream	42
<code>dump_perm()</code>	Dump permutation data structure to a stream	42
<code>dump_vec()</code>	Dump vector data structure to a stream	42
<code>ERRABORT()</code>	Abort on error (sets flag, macro)	39
<code>ERREXIT()</code>	Exit on error (sets flag, macro)	39
<code>error()</code>	Raise an error (macro, see <code>ev_err()</code> )	37
<code>ev_err()</code>	Raise an error (function)	37
<code>fin_ivec()</code>	Input integer vector from a stream	40
<code>fin_mat()</code>	Input matrix from a stream	40
<code>fin_perm()</code>	Input permutation from a stream	40
<code>fininput()</code>	Input a simple data item from a stream	43
<code>fin_vec()</code>	Input vector from a stream	40

Function	Description	Page
fout_ivec()	Output integer vector to a stream	42
fout_mat()	Output matrix to a stream	42
fout_perm()	Output permutation to a stream	42
fout_row()	Output sparse row to a stream	106
fout_vec()	Output vector to a stream	42
fprompter()	Print prompt to <code>stderr</code>	43
freeivec()	Free (deallocate) an integer vector (macro)	44
freemat()	Free (deallocate) a matrix (macro)	44
freeperm()	Free (deallocate) a permutation (macro)	44
freevec()	Free (deallocate) a vector (macro)	44
get_col()	Extract a column from a matrix	46
get_ivec()	Allocate and initialise an integer vector	45
get_mat()	Allocate and initialise a matrix	45
get_perm()	Allocate and initialise a permutation	45
get_row()	Extract a row from a matrix	46
get_vec()	Allocate and initialise a vector	45
givens()	Compute Givens parameters	80
hhtrcols()	Compute $AP^T$ where $P$ is a Householder matrix	82
hhtrrows()	Compute $PA$ where $P$ is a Householder matrix	82
hhtrvec()	Compute $Px$ where $P$ is a Householder matrix	82
hhvec()	Compute parameters for a Householder matrix	82
id_mat()	Sets matrix to identity matrix	47
in_ivec()	Input integer vector from <code>stdin</code> (macro)	40
in_mat()	Input matrix from <code>stdin</code> (macro)	40
in_perm()	Input permutation from <code>stdin</code> (macro)	40
in_prod()	Inner product of vectors	49
input()	Input a simple data item from <code>stdin</code> (macro)	43
in_vec()	Input vector from <code>stdin</code> (macro)	40
_ip_()	Inner product of arrays	69
iv_add()	Add integer vectors	50
iv_free()	Free (deallocate) integer vector (function)	44
iv_resize()	Resize an integer vector	51
iv_sub()	Subtract integer vectors	50
lanczos()	Lanczos routine (no reorthogonalisation)	116
lanczos2()	Lanczos routine (Cullum & Willoughby version)	116
LDLfactor()	$LDL^T$ factorisation	73
LDLsolve()	$LDL^T$ solver	73
LDLupdate()	Update $LDL^T$ factorisation	86
Lsolve()	Solve $Lx = y$ , $L$ lower triangular	84
lsqr()	LSQR algorithm of Paige and Saunders	113
LTsolve()	Solve $L^T x = y$ , $L$ lower triangular	84
LUcondest()	Estimate a condition number using $LU$ factors	75
LUfactor()	Compute $LU$ factors with implicit scaled partial pivoting	75
LUsolve()	Solve $Ax = b$ using $LU$ factors	75
LUTsolve()	Solve $A^T x = b$ using $LU$ factors	75

Function	Description	Page
<code>m_add()</code>	Add matrices	54
<code>makeQ()</code>	Form $Q$ matrix for $QR$ factorisation	79
<code>makeR()</code>	Form $R$ matrix for $QR$ factorisation	79
<code>MCHfactor()</code>	Modified Cholesky factorisation (actually factors $A+D$ , $D$ diagonal, instead of $A$ )	73
<code>mem_copy()</code>	Copy memory (macro)	105
<code>mem_zero()</code>	Zero memory (macro)	105
<code>m_free()</code>	Free (deallocate) matrix (function)	44
<code>m_inverse()</code>	Invert matrix	75
<code>m_load()</code>	Load matrix in MATLAB format	55
<code>_mltadd_()</code>	Forms $x + \alpha y$ for arrays	69
<code>m_mlt()</code>	Multiplies matrices	54
<code>mmtr_mlt()</code>	Computes $AB^T$	57
<code>m_norm1()</code>	Computes $\ A\ _1$ of a matrix	58
<code>m_norm_frob()</code>	Computes the Frobenius norm of a matrix	58
<code>m_norm_inf()</code>	Computes $\ A\ _\infty$ of a matrix	58
<code>m_resize()</code>	Resize matrix	51
<code>m_save()</code>	Save matrix in MATLAB format	55
<code>m_sub()</code>	Subtract matrices	54
<code>m_transp()</code>	Transpose matrix	57
<code>mtrm_mlt()</code>	Computes $A^T B$	57
<code>mv_mlt()</code>	Computes $Ax$	59
<code>mv_mltadd()</code>	Computes $y \leftarrow Ax + y$	59
<code>ON_ERROR()</code>	Error handler (macro)	39
<code>ones_mat()</code>	Set matrix to all 1's	47
<code>ones_vec()</code>	Set vector to all 1's	47
<code>out_ivec()</code>	Output integer vector to <code>stdout</code> (macro)	42
<code>out_mat()</code>	Output matrix to <code>stdout</code> (macro)	42
<code>out_perm()</code>	Output permutation to <code>stdout</code> (macro)	42
<code>out_vec()</code>	Output vector to <code>stdout</code> (macro)	42
<code>pccg()</code>	Pre-conditioned conjugate gradients	113
<code>prompter()</code>	Print prompt message to <code>stdout</code>	43
<code>px_cols()</code>	Permute the columns of a matrix	61
<code>px_free()</code>	Free (deallocate) permutation (function)	44
<code>px_id()</code>	Sets permutation to identity	60
<code>px_inv()</code>	Invert permutation	60
<code>px_invvec()</code>	Computes $P^T x$ where $P$ is a permutation matrix	61
<code>px_mlt()</code>	Multiply permutations	60
<code>px_resize()</code>	Resize a permutation	51
<code>px_rows()</code>	Permute the rows of a matrix	61
<code>px_vec()</code>	Computes $PX$ where $P$ is a permutation matrix	61
<code>QRCPfactor()</code>	$QR$ factorisation with column pivoting	77
<code>QRfactor()</code>	$QR$ factorisation	77
<code>QRsolve()</code>	Solve $Ax = b$ using $QR$ factorisation	77
<code>QRupdate()</code>	Update explicit $QR$ factors	86
<code>rand_mat()</code>	Randomise entries of a matrix	47
<code>rand_vec()</code>	Randomise entries of a vector	47

Function	Description	Page
rot_cols()	Apply Givens rotation to the columns of a matrix	80
rot_rows()	Apply Givens rotation to the rows of a matrix	80
rot_vec()	Apply Givens rotation to a vector	80
_row_mltadd()	Sparse row vector multiply-and-add	106
row_set_val()	Set the value of a sparse row	106
row_xpd()	Expand a sparse row	106
schur()	Compute real Schur form	88
schur_evals()	Compute eigenvalues from the real Schur form	90
schur_vecs()	Compute eigenvectors from the real Schur form	90
set_col()	Set the column of a matrix to a given vector	63
set_err_flag()	Control behaviour of <code>ev_err()</code>	37
set_row()	Set the row of a matrix to a given vector	63
_smlt_()	Scalar-vector multiplication for arrays	69
sm_mlt()	Scalar-matrix multiplication	54
sp_arnoldi()	Arnoldi routine using sparse matrix data structure	119
sp_cgs()	CGS routine using sparse matrix data structure	113
spCHfactor()	Sparse Cholesky factorisation	108
spCHsolve()	Sparse Cholesky solver	108
spCHsymb()	Symbolic sparse Cholesky factorisation (no floating point operations)	108
sp_col_access()	Sets up column access paths for a sparse matrix	99
sp_compact()	Eliminates zero entries in a sparse matrix	94
sp_cp_mat()	Copies a sparse matrix	94
sp_cp_mat2()	Copies a sparse matrix into another	94
sp_diag_access()	Sets up diagonal access paths for a sparse matrix	99
sp_dump_mat()	Dump sparse matrix data structure to a stream	102
sp_fin_mat()	Input sparse matrix from a stream	104
sp_fout_mat()	Output a sparse matrix to a stream	102
sp_free_mat()	Free (deallocate) a sparse matrix	94
sp_get_idx()	Get location of an entry in a sparse row	106
sp_get_mat()	Allocate and initialise a sparse matrix	94
sp_get_row()	Allocate and initialise a sparse row	106
sp_get_val()	Get the $(i, j)$ entry of a sparse matrix	97
spICHfactor()	Sparse incomplete Cholesky factorisation	108
sp_in_mat()	Input a sparse matrix form <code>stdin</code>	104
sp_lsqr()	LSQR routine using sparse matrix data structure	113
spLUfactor()	Sparse $LU$ factorisation using partial pivoting	110
spLUsolve()	Solves $Ax = b$ using sparse $LU$ factors	110
spLUTsolve()	Solves $A^T x = b$ using sparse $LU$ factors	110
sp_mv_mlt()	Computes $Ax$ for sparse $A$	98
sp_out_mat()	Outputs a sparse matrix to a stream (macro)	102
sp_pccg()	Pre-conditioned conjugate gradients using sparse matrix data structures	113
sp_resize()	Resize a sparse matrix	94
sp_row_merge()	Merge two sparse rows	106
sp_set_val()	Set the $(i, j)$ entry of a sparse matrix	97
sp_vm_mlt()	Compute $x^T A$ for sparse $A$	98
sp_zero_mat()	Zero (but do not remove) all entries of a sparse matrix	101

Function	Description	Page
<code>__sub__()</code>	Subtract an array from another	69
<code>svd()</code>	Compute the SVD of a matrix	91
<code>sv_mlt()</code>	Scalar-vector multiply	64
<code>symmeig()</code>	Compute eigenvalues/vectors of a symmetric matrix	88
<code>tracecatch()</code>	Catch and re-raise errors (macro)	34
<code>trieig()</code>	Compute eigenvalues/vectors of a symmetric tridiagonal matrix	88
<code>Usolve()</code>	Solve $Ux = b$ where $U$ is upper triangular	84
<code>UTsolve()</code>	Solve $U^T x = b$ where $U$ is upper triangular	84
<code>v_free()</code>	Free (deallocate) vector (function)	44
<code>v_lincomb()</code>	Compute $\sum_i a_i x_i$ for an array of vectors	67
<code>v_linlist()</code>	Compute $\sum_i a_i x_i$ for a list of vectors	67
<code>v_max()</code>	Computes max vector entry & index	65
<code>v_min()</code>	Computes min vector entry & index	65
<code>v_mltadd()</code>	Computes $y \leftarrow \alpha x + y$ for vectors $x, y$	64
<code>vm_mlt()</code>	Computes $x^T A$	59
<code>vm_mltadd()</code>	Computes $y^T \leftarrow y^T + x^T A$	59
<code>v_norm1()</code>	Computes $\ x\ _1$ for a vector	68
<code>v_norm2()</code>	Computes $\ x\ _2$ (the Euclidean norm) of a vector	68
<code>v_norm_inf()</code>	Computes $\ x\ _\infty$ for a vector	68
<code>v_resize()</code>	Resize a vector	51
<code>v_save()</code>	Save a vector in MATLAB format	55
<code>v_slash()</code>	Componentwise multiplicative inverse	65
<code>v_sort()</code>	Sorts vector components	65
<code>v_star()</code>	Componentwise vector product	65
<code>__zero__()</code>	Zero an array	69
<code>zero_mat()</code>	Zero a matrix	47
<code>zero_vec()</code>	Zero a vector	47



# Contents