

MatrixLab

1.2

Generated by Doxygen 1.8.1.1

Mon Apr 27 2015 23:45:49



# Contents

<b>1</b>	<b>Matrixlab</b>	<b>1</b>
1.1	Introduction . . . . .	1
<b>2</b>	<b>matrixlab</b>	<b>3</b>
<b>3</b>	<b>Data Structure Index</b>	<b>5</b>
3.1	Data Structures . . . . .	5
<b>4</b>	<b>File Index</b>	<b>7</b>
4.1	File List . . . . .	7
<b>5</b>	<b>Data Structure Documentation</b>	<b>9</b>
5.1	mat_bayes_model Struct Reference . . . . .	9
5.1.1	Detailed Description . . . . .	9
5.1.2	Field Documentation . . . . .	9
5.1.2.1	class_covars . . . . .	9
5.1.2.2	class_labels . . . . .	9
5.1.2.3	class_means . . . . .	9
5.1.2.4	class_priors . . . . .	9
5.1.2.5	num_of_classes . . . . .	10
5.1.2.6	num_of_features . . . . .	10
5.2	mat_gnode Struct Reference . . . . .	10
5.2.1	Detailed Description . . . . .	10
5.2.2	Field Documentation . . . . .	10
5.2.2.1	next . . . . .	10
5.2.2.2	v . . . . .	10
5.2.2.3	weight . . . . .	10
5.3	mat_graph Struct Reference . . . . .	10
5.3.1	Detailed Description . . . . .	11
5.3.2	Field Documentation . . . . .	11
5.3.2.1	adj . . . . .	11
5.3.2.2	dad . . . . .	11
5.3.2.3	id . . . . .	11

5.3.2.4	nedges	11
5.3.2.5	nvertices	11
5.3.2.6	pq	11
5.3.2.7	val	11
5.3.2.8	vseq	11
5.3.2.9	weighted	11
5.3.2.10	z	11
5.4	mat_int_priorityqueue Struct Reference	11
5.4.1	Detailed Description	12
5.4.2	Field Documentation	12
5.4.2.1	element	12
5.4.2.2	length	12
5.4.2.3	p	12
5.4.2.4	type	12
5.5	mat_int_queue Struct Reference	12
5.5.1	Detailed Description	12
5.5.2	Field Documentation	13
5.5.2.1	head	13
5.5.2.2	p	13
5.5.2.3	tail	13
5.6	mat_int_stack Struct Reference	13
5.6.1	Detailed Description	13
5.6.2	Field Documentation	13
5.6.2.1	length	13
5.6.2.2	p	13
5.6.2.3	stack	13
5.7	mat_intpqnode Struct Reference	14
5.7.1	Detailed Description	14
5.7.2	Field Documentation	14
5.7.2.1	data	14
5.7.2.2	priority	14
5.8	mat_kdnode Struct Reference	14
5.8.1	Field Documentation	14
5.8.1.1	idx	14
5.8.1.2	left	14
5.8.1.3	right	14
5.8.1.4	x	14
5.9	mat_kdtree Struct Reference	15
5.9.1	Field Documentation	15
5.9.1.1	_is_allocated	15

5.9.1.2	data	15
5.9.1.3	kdtree	15
5.9.1.4	length	15
5.9.1.5	ndims	15
5.10	mat_mtype_priorityqueue Struct Reference	15
5.10.1	Detailed Description	15
5.10.2	Field Documentation	15
5.10.2.1	element	15
5.10.2.2	length	16
5.10.2.3	p	16
5.10.2.4	type	16
5.11	mat_mtype_queue Struct Reference	16
5.11.1	Detailed Description	16
5.11.2	Field Documentation	16
5.11.2.1	head	16
5.11.2.2	p	16
5.11.2.3	tail	16
5.12	mat_mtype_stack Struct Reference	17
5.12.1	Detailed Description	17
5.12.2	Field Documentation	17
5.12.2.1	length	17
5.12.2.2	p	17
5.12.2.3	stack	17
5.13	mat_mtypepqnode Struct Reference	17
5.13.1	Detailed Description	17
5.13.2	Field Documentation	18
5.13.2.1	data	18
5.13.2.2	priority	18
5.14	mat_perceptron Struct Reference	18
5.14.1	Detailed Description	18
5.14.2	Field Documentation	18
5.14.2.1	class_labels	18
5.14.2.2	class_weights	18
5.14.2.3	istrained	18
5.14.2.4	num_of_classes	18
5.14.2.5	num_of_features	19
5.14.2.6	num_of_iterations	19
5.15	mat_qintnode Struct Reference	19
5.15.1	Detailed Description	19
5.15.2	Field Documentation	19

5.15.2.1	data	19
5.15.2.2	next	19
5.16	mat_qmtypenode Struct Reference	19
5.16.1	Detailed Description	20
5.16.2	Field Documentation	20
5.16.2.1	data	20
5.16.2.2	next	20
5.17	mat_tree_node Struct Reference	20
5.17.1	Detailed Description	20
5.17.2	Field Documentation	20
5.17.2.1	element	20
5.17.2.2	left	20
5.17.2.3	right	20
<b>6</b>	<b>File Documentation</b>	<b>21</b>
6.1	matabs.c File Reference	21
6.1.1	Function Documentation	21
6.1.1.1	mat_abs	21
6.2	matadd.c File Reference	21
6.2.1	Function Documentation	22
6.2.1.1	int_vec_add	22
6.2.1.2	int_vec_adds	22
6.2.1.3	mat_add	22
6.2.1.4	mat_adds	22
6.3	matcompress.c File Reference	23
6.4	matconcat.c File Reference	23
6.4.1	Function Documentation	23
6.4.1.1	int_vec_concat	23
6.4.1.2	mat_concat	23
6.5	matconv.c File Reference	23
6.5.1	Function Documentation	24
6.5.1.1	int_vec2_mat	24
6.5.1.2	mat_2int_vec	24
6.5.1.3	mat_vectorize	24
6.5.1.4	mat_vectorize_tr	24
6.6	matcreat.c File Reference	25
6.6.1	Function Documentation	26
6.6.1.1	int_vec_append	26
6.6.1.2	int_vec_copy	26
6.6.1.3	int_vec_creat	26

6.6.1.4	<a href="#">int_vec_fill</a>	27
6.6.1.5	<a href="#">int_vec_fill_type</a>	27
6.6.1.6	<a href="#">int_vec_free</a>	27
6.6.1.7	<a href="#">int_vecstack_creat</a>	27
6.6.1.8	<a href="#">int_vecstack_free</a>	28
6.6.1.9	<a href="#">mat_bayes_model_creat</a>	28
6.6.1.10	<a href="#">mat_bayes_model_free</a>	28
6.6.1.11	<a href="#">mat_colcopy</a>	28
6.6.1.12	<a href="#">mat_copy</a>	29
6.6.1.13	<a href="#">mat_creat</a>	29
6.6.1.14	<a href="#">mat_creat_diag</a>	29
6.6.1.15	<a href="#">mat_fgetmat</a>	29
6.6.1.16	<a href="#">mat_fill</a>	30
6.6.1.17	<a href="#">mat_fill_type</a>	30
6.6.1.18	<a href="#">mat_free</a>	30
6.6.1.19	<a href="#">mat_perceptron_creat</a>	30
6.6.1.20	<a href="#">mat_perceptron_free</a>	30
6.6.1.21	<a href="#">mat_rowcopy</a>	31
6.6.1.22	<a href="#">mat_xcopy</a>	31
6.6.1.23	<a href="#">mat_xjoin</a>	31
6.6.1.24	<a href="#">matstack_append</a>	32
6.6.1.25	<a href="#">matstack_creat</a>	32
6.6.1.26	<a href="#">matstack_free</a>	32
6.6.1.27	<a href="#">matvec_creat</a>	32
6.6.1.28	<a href="#">matvec_free</a>	32
6.7	<a href="#">matdatastruct.c File Reference</a>	33
6.7.1	<a href="#">Function Documentation</a>	34
6.7.1.1	<a href="#">mat_bs_delete</a>	34
6.7.1.2	<a href="#">mat_bs_find</a>	34
6.7.1.3	<a href="#">mat_bs_find_max</a>	34
6.7.1.4	<a href="#">mat_bs_find_min</a>	34
6.7.1.5	<a href="#">mat_bs_free</a>	34
6.7.1.6	<a href="#">mat_bs_inorder</a>	34
6.7.1.7	<a href="#">mat_bs_insert</a>	34
6.7.1.8	<a href="#">mat_bs_make_null</a>	34
6.7.1.9	<a href="#">mat_int_priorityqueue_creat</a>	34
6.7.1.10	<a href="#">mat_int_priorityqueue_dequeue</a>	34
6.7.1.11	<a href="#">mat_int_priorityqueue_enqueue</a>	34
6.7.1.12	<a href="#">mat_int_priorityqueue_free</a>	34
6.7.1.13	<a href="#">mat_int_priorityqueue_is_empty</a>	34

6.7.1.14	<a href="#">mat_int_priorityqueue_update</a>	34
6.7.1.15	<a href="#">mat_int_queue_creat</a>	34
6.7.1.16	<a href="#">mat_int_queue_dequeue</a>	34
6.7.1.17	<a href="#">mat_int_queue_enqueue</a>	34
6.7.1.18	<a href="#">mat_int_queue_free</a>	34
6.7.1.19	<a href="#">mat_int_queue_is_empty</a>	34
6.7.1.20	<a href="#">mat_int_stack_creat</a>	34
6.7.1.21	<a href="#">mat_int_stack_free</a>	34
6.7.1.22	<a href="#">mat_int_stack_is_empty</a>	34
6.7.1.23	<a href="#">mat_int_stack_pop</a>	34
6.7.1.24	<a href="#">mat_int_stack_push</a>	34
6.7.1.25	<a href="#">mat_mtype_priorityqueue_creat</a>	34
6.7.1.26	<a href="#">mat_mtype_priorityqueue_dequeue</a>	34
6.7.1.27	<a href="#">mat_mtype_priorityqueue_enqueue</a>	34
6.7.1.28	<a href="#">mat_mtype_priorityqueue_free</a>	35
6.7.1.29	<a href="#">mat_mtype_priorityqueue_is_empty</a>	35
6.7.1.30	<a href="#">mat_mtype_priorityqueue_update</a>	35
6.7.1.31	<a href="#">mat_mtype_queue_creat</a>	35
6.7.1.32	<a href="#">mat_mtype_queue_dequeue</a>	35
6.7.1.33	<a href="#">mat_mtype_queue_enqueue</a>	35
6.7.1.34	<a href="#">mat_mtype_queue_free</a>	35
6.7.1.35	<a href="#">mat_mtype_queue_is_empty</a>	35
6.7.1.36	<a href="#">mat_mtype_stack_creat</a>	35
6.7.1.37	<a href="#">mat_mtype_stack_free</a>	35
6.7.1.38	<a href="#">mat_mtype_stack_is_empty</a>	35
6.7.1.39	<a href="#">mat_mtype_stack_pop</a>	35
6.7.1.40	<a href="#">mat_mtype_stack_push</a>	35
6.8	<a href="#">matdet.c File Reference</a>	35
6.8.1	<a href="#">Function Documentation</a>	35
6.8.1.1	<a href="#">mat_cofact</a>	35
6.8.1.2	<a href="#">mat_det</a>	36
6.8.1.3	<a href="#">mat_minor</a>	36
6.9	<a href="#">matdiv.c File Reference</a>	36
6.9.1	<a href="#">Function Documentation</a>	36
6.9.1.1	<a href="#">int_vec_div</a>	36
6.9.1.2	<a href="#">int_vec_divs</a>	37
6.9.1.3	<a href="#">mat_div_dot</a>	37
6.9.1.4	<a href="#">mat_divs</a>	37
6.10	<a href="#">matdump.c File Reference</a>	37
6.10.1	<a href="#">Function Documentation</a>	38



6.10.1.1	<a href="#">int_vec_dump</a>	38
6.10.1.2	<a href="#">int_vec_dumpf</a>	38
6.10.1.3	<a href="#">int_vec_fdump</a>	38
6.10.1.4	<a href="#">int_vec_fdumpf</a>	38
6.10.1.5	<a href="#">mat_dump</a>	39
6.10.1.6	<a href="#">mat_dumpf</a>	39
6.10.1.7	<a href="#">mat_fdump</a>	39
6.10.1.8	<a href="#">mat_fdumpf</a>	39
6.11	<a href="#">matdurn.c File Reference</a>	39
6.11.1	<a href="#">Function Documentation</a>	39
6.11.1.1	<a href="#">mat_durbin</a>	39
6.11.1.2	<a href="#">mat_lsolve_durbin</a>	40
6.11.1.3	<a href="#">mat_qr</a>	40
6.12	<a href="#">materr.c File Reference</a>	40
6.12.1	<a href="#">Function Documentation</a>	41
6.12.1.1	<a href="#">gen_error</a>	41
6.12.1.2	<a href="#">graph_error</a>	41
6.12.1.3	<a href="#">int_vec_error</a>	41
6.12.1.4	<a href="#">int_vecstack_error</a>	41
6.12.1.5	<a href="#">mat_error</a>	42
6.12.1.6	<a href="#">matstack_error</a>	42
6.12.1.7	<a href="#">pq_error</a>	42
6.12.1.8	<a href="#">queue_error</a>	42
6.12.1.9	<a href="#">stack_error</a>	42
6.13	<a href="#">matfft.c File Reference</a>	42
6.13.1	<a href="#">Function Documentation</a>	43
6.13.1.1	<a href="#">mat_fft2</a>	43
6.14	<a href="#">matfilter.c File Reference</a>	43
6.14.1	<a href="#">Function Documentation</a>	43
6.14.1.1	<a href="#">mat_conv2</a>	43
6.15	<a href="#">matfit.c File Reference</a>	43
6.15.1	<a href="#">Function Documentation</a>	43
6.15.1.1	<a href="#">mat_least_squares</a>	44
6.15.1.2	<a href="#">mat_linear_ls_fit</a>	44
6.15.1.3	<a href="#">mat_rob_least_squares</a>	44
6.15.1.4	<a href="#">mat_robust_fit</a>	44
6.15.1.5	<a href="#">mat_w_least_squares</a>	44
6.16	<a href="#">matflip.c File Reference</a>	44
6.16.1	<a href="#">Function Documentation</a>	44
6.16.1.1	<a href="#">mat_fliplr</a>	44

6.16.1.2	<code>mat_flipud</code>	44
6.17	<code>matfuncs.c</code> File Reference	44
6.17.1	Function Documentation	45
6.17.1.1	<code>__mat_addfunc</code>	45
6.17.1.2	<code>__mat_arccosh</code>	45
6.17.1.3	<code>__mat_arcsinh</code>	45
6.17.1.4	<code>__mat_arctanh</code>	46
6.17.1.5	<code>__mat_bisquare_wt</code>	46
6.17.1.6	<code>__mat_divfunc</code>	46
6.17.1.7	<code>__mat_huber_wt</code>	46
6.17.1.8	<code>__mat_logplusone</code>	47
6.17.1.9	<code>__mat_mulfunc</code>	47
6.17.1.10	<code>__mat_sqrfunc</code>	47
6.17.1.11	<code>__mat_sqrtfunc</code>	47
6.17.1.12	<code>__mat_subfunc</code>	48
6.17.1.13	<code>mat_bisquare_wt</code>	48
6.17.1.14	<code>mat_gfunc</code>	48
6.17.1.15	<code>mat_huber_wt</code>	48
6.18	<code>matgraph.c</code> File Reference	49
6.18.1	Function Documentation	49
6.18.1.1	<code>mat_graph_adjlist</code>	49
6.18.1.2	<code>mat_graph_adjm_to_adjl</code>	49
6.18.1.3	<code>mat_graph_creat</code>	49
6.18.1.4	<code>mat_graph_dump</code>	49
6.18.1.5	<code>mat_graph_dumpf</code>	49
6.18.1.6	<code>mat_graph_reverse</code>	49
6.18.1.7	<code>mat_graph_search</code>	49
6.18.1.8	<code>mat_graph_visit</code>	49
6.19	<code>matinnerprod.c</code> File Reference	49
6.19.1	Function Documentation	49
6.19.1.1	<code>mat_innerprod</code>	49
6.19.1.2	<code>mat_norm_inf</code>	49
6.19.1.3	<code>mat_norm_one</code>	49
6.19.1.4	<code>mat_norm_p</code>	50
6.20	<code>matintegrate.c</code> File Reference	50
6.20.1	Function Documentation	50
6.20.1.1	<code>mat_int_qadrat</code>	50
6.20.1.2	<code>mat_int_simpson</code>	50
6.20.1.3	<code>mat_int_trapezoid</code>	50
6.21	<code>matinv.c</code> File Reference	51

6.21.1	Function Documentation	51
6.21.1.1	mat_inv	51
6.21.1.2	mat_reg_inv	51
6.22	matkdtree.c File Reference	51
6.22.1	Function Documentation	52
6.22.1.1	mat_kdtree_free	52
6.22.1.2	mat_kdtree_k_nearest	52
6.22.1.3	mat_kdtree_make_tree	52
6.22.1.4	mat_kdtree_nearest	52
6.23	matmaxmin.c File Reference	53
6.23.1	Function Documentation	53
6.23.1.1	mat_max	53
6.23.1.2	mat_min	53
6.24	matmds.c File Reference	53
6.24.1	Function Documentation	53
6.24.1.1	__mat_mds_metric	53
6.24.1.2	__mat_mds_nonmetric	53
6.24.1.3	mat_mds	53
6.25	matmean.c File Reference	53
6.25.1	Function Documentation	53
6.25.1.1	mat_mean	53
6.25.1.2	mat_mean_col	54
6.25.1.3	mat_mean_row	54
6.26	matmisc.c File Reference	54
6.26.1	Function Documentation	54
6.26.1.1	__mat_cart2pol	54
6.26.1.2	__mat_pol2cart	54
6.26.1.3	int_vec_permute_vect	54
6.26.1.4	int_vec_unique	55
6.26.1.5	mat_bsxfun	55
6.26.1.6	mat_calc_dist_sq	55
6.26.1.7	mat_cart2pol	55
6.26.1.8	mat_find_within_dist	56
6.26.1.9	mat_fnextline	56
6.26.1.10	mat_get_sub_matrix_from_cols	56
6.26.1.11	mat_get_sub_matrix_from_rows	56
6.26.1.12	mat_get_sub_vector	57
6.26.1.13	mat_nextline	57
6.26.1.14	mat_pick_col	57
6.26.1.15	mat_pick_row	57

6.26.1.16	mat_pol2cart	57
6.26.1.17	mats_isinf	58
6.26.1.18	mats_isnan	58
6.27	matmul.c File Reference	58
6.27.1	Function Documentation	58
6.27.1.1	int_vec_mul	58
6.27.1.2	int_vec_muls	58
6.27.1.3	mat_diagmul	58
6.27.1.4	mat_mul	59
6.27.1.5	mat_mul_dot	59
6.27.1.6	mat_mul_fast	59
6.27.1.7	mat_muls	59
6.28	matpca.c File Reference	59
6.28.1	Function Documentation	59
6.28.1.1	mat_corcol	59
6.28.1.2	mat_covcol	59
6.28.1.3	mat_eig_sym	59
6.28.1.4	mat_pca	59
6.28.1.5	mat_scpcol	59
6.28.1.6	mat_tqli	59
6.28.1.7	mat_tred2	59
6.29	matpinv.c File Reference	59
6.29.1	Function Documentation	59
6.29.1.1	mat_pinv	59
6.29.1.2	mat_wpinv	60
6.30	matpoly.c File Reference	60
6.30.1	Function Documentation	61
6.30.1.1	mat_binom	61
6.30.1.2	mat_binom_init	61
6.30.1.3	mat_cheby	61
6.30.1.4	mat_cheby_approx	61
6.30.1.5	mat_cheby_coeffs_to_poly	62
6.30.1.6	mat_cheby_init	62
6.30.1.7	mat_legendre	62
6.30.1.8	mat_legendre_init	62
6.30.1.9	mat_poly_add	62
6.30.1.10	mat_poly_diff	63
6.30.1.11	mat_poly_diff_eval	63
6.30.1.12	mat_poly_div	63
6.30.1.13	mat_poly_eval	63

6.30.1.14	mat_poly_mul	64
6.30.1.15	mat_poly_scale	64
6.30.1.16	mat_poly_shift	64
6.30.2	Variable Documentation	65
6.30.2.1	mat_binom_series_table	65
6.30.2.2	mat_cheby_series_table	65
6.30.2.3	mat_legendre_series_table	65
6.31	matprec.c File Reference	65
6.31.1	Function Documentation	65
6.31.1.1	mat_bayes_classifier_test	65
6.31.1.2	mat_bayes_classifier_train	65
6.31.1.3	mat_kmeans	65
6.31.1.4	mat_perceptron_test	65
6.31.1.5	mat_perceptron_train	65
6.31.1.6	mat_perceptron_train_	65
6.32	matpursuit.c File Reference	65
6.32.1	Function Documentation	65
6.32.1.1	mat_omp	65
6.33	matrand.c File Reference	65
6.33.1	Function Documentation	66
6.33.1.1	__mat_rand	66
6.33.1.2	__mat_randexp	66
6.33.1.3	__mat_randfun	66
6.33.1.4	__mat_randn	66
6.33.1.5	int_vec_randperm	66
6.33.1.6	mat_rand	66
6.33.1.7	mat_randexp	66
6.33.1.8	mat_randfun	66
6.33.1.9	mat_randn	66
6.33.1.10	mat_randperm	66
6.33.1.11	mat_randperm_n	66
6.33.1.12	mat_set_seed	66
6.33.2	Variable Documentation	66
6.33.2.1	MAT_SEED	66
6.33.2.2	MAT_SET_SEED	66
6.34	matrix.c File Reference	66
6.35	matrix.h File Reference	66
6.35.1	Typedef Documentation	77
6.35.1.1	INT_VECSTACK	77
6.35.1.2	INT_VECTOR	77

6.35.1.3	<code>mat_bayes_model</code>	77
6.35.1.4	<code>MAT_BAYES_MODEL</code>	77
6.35.1.5	<code>mat_gnode</code>	77
6.35.1.6	<code>MAT_GNODE</code>	77
6.35.1.7	<code>mat_graph</code>	77
6.35.1.8	<code>MAT_GRAPH</code>	77
6.35.1.9	<code>mat_int_priorityqueue</code>	77
6.35.1.10	<code>MAT_INT_PRIORITYQUEUE</code>	78
6.35.1.11	<code>mat_int_queue</code>	78
6.35.1.12	<code>MAT_INT_QUEUE</code>	78
6.35.1.13	<code>mat_int_stack</code>	78
6.35.1.14	<code>MAT_INT_STACK</code>	78
6.35.1.15	<code>mat_intpqnode</code>	78
6.35.1.16	<code>MAT_INTPQNODE</code>	78
6.35.1.17	<code>mat_kdnode</code>	78
6.35.1.18	<code>MAT_KDNODE</code>	78
6.35.1.19	<code>mat_kdtree</code>	78
6.35.1.20	<code>MAT_KDTREE</code>	78
6.35.1.21	<code>mat_mtype_priorityqueue</code>	78
6.35.1.22	<code>MAT_MTYPE_PRIORITYQUEUE</code>	78
6.35.1.23	<code>mat_mtype_queue</code>	79
6.35.1.24	<code>MAT_MTYPE_QUEUE</code>	79
6.35.1.25	<code>mat_mtype_stack</code>	79
6.35.1.26	<code>MAT_MTYPE_STACK</code>	79
6.35.1.27	<code>mat_mtypepqnode</code>	79
6.35.1.28	<code>MAT_MTYPEPQNODE</code>	79
6.35.1.29	<code>mat_perceptron</code>	79
6.35.1.30	<code>MAT_PERCEPTRON</code>	79
6.35.1.31	<code>mat_qintnode</code>	79
6.35.1.32	<code>MAT_QINTNODE</code>	79
6.35.1.33	<code>mat_qmtypenode</code>	80
6.35.1.34	<code>MAT_QMTYPENODE</code>	80
6.35.1.35	<code>MAT_TREE</code>	80
6.35.1.36	<code>mat_tree_node</code>	80
6.35.1.37	<code>MAT_TREE_NODE</code>	80
6.35.1.38	<code>MATRIX</code>	80
6.35.1.39	<code>MATSTACK</code>	80
6.35.1.40	<code>MATVEC_DPOINTER</code>	80
6.35.2	Function Documentation	80
6.35.2.1	<code>__int_vec_creat</code>	80

6.35.2.2	<a href="#">__int_vecstack_creat</a>	80
6.35.2.3	<a href="#">__mat_addfunc</a>	80
6.35.2.4	<a href="#">__mat_arccosh</a>	81
6.35.2.5	<a href="#">__mat_arcsinh</a>	81
6.35.2.6	<a href="#">__mat_arctanh</a>	81
6.35.2.7	<a href="#">__mat_bisquare_wt</a>	81
6.35.2.8	<a href="#">__mat_cart2pol</a>	82
6.35.2.9	<a href="#">__mat_creat</a>	82
6.35.2.10	<a href="#">__mat_divfunc</a>	82
6.35.2.11	<a href="#">__mat_fft</a>	82
6.35.2.12	<a href="#">__mat_huber_wt</a>	82
6.35.2.13	<a href="#">__mat_kd_find_median</a>	82
6.35.2.14	<a href="#">__mat_kdtree_k_nearest</a>	82
6.35.2.15	<a href="#">__mat_kdtree_make_tree</a>	82
6.35.2.16	<a href="#">__mat_kdtree_nearest</a>	82
6.35.2.17	<a href="#">__mat_lint</a>	82
6.35.2.18	<a href="#">__mat_logplusone</a>	82
6.35.2.19	<a href="#">__mat_mds_metric</a>	83
6.35.2.20	<a href="#">__mat_mds_nonmetric</a>	83
6.35.2.21	<a href="#">__mat_mulfunc</a>	83
6.35.2.22	<a href="#">__mat_pol2cart</a>	83
6.35.2.23	<a href="#">__mat_powerof2</a>	83
6.35.2.24	<a href="#">__mat_quicksort</a>	83
6.35.2.25	<a href="#">__mat_rand</a>	83
6.35.2.26	<a href="#">__mat_randexp</a>	83
6.35.2.27	<a href="#">__mat_randfun</a>	83
6.35.2.28	<a href="#">__mat_randn</a>	83
6.35.2.29	<a href="#">__mat_sqrfunc</a>	83
6.35.2.30	<a href="#">__mat_sqrtfunc</a>	83
6.35.2.31	<a href="#">__mat_subfunc</a>	84
6.35.2.32	<a href="#">__matstack_creat</a>	84
6.35.2.33	<a href="#">gen_abs_ceil</a>	84
6.35.2.34	<a href="#">gen_eq</a>	84
6.35.2.35	<a href="#">gen_error</a>	84
6.35.2.36	<a href="#">gen_gt</a>	84
6.35.2.37	<a href="#">gen_lt</a>	84
6.35.2.38	<a href="#">graph_error</a>	84
6.35.2.39	<a href="#">int_vec2_mat</a>	84
6.35.2.40	<a href="#">int_vec_add</a>	85
6.35.2.41	<a href="#">int_vec_adds</a>	85

6.35.2.42 int_vec_append . . . . .	85
6.35.2.43 int_vec_concat . . . . .	85
6.35.2.44 int_vec_copy . . . . .	86
6.35.2.45 int_vec_creat . . . . .	86
6.35.2.46 int_vec_div . . . . .	86
6.35.2.47 int_vec_divs . . . . .	86
6.35.2.48 int_vec_dump . . . . .	87
6.35.2.49 int_vec_dumpf . . . . .	87
6.35.2.50 int_vec_error . . . . .	87
6.35.2.51 int_vec_fdump . . . . .	87
6.35.2.52 int_vec_dumpf . . . . .	87
6.35.2.53 int_vec_fill . . . . .	88
6.35.2.54 int_vec_fill_type . . . . .	88
6.35.2.55 int_vec_find . . . . .	88
6.35.2.56 int_vec_free . . . . .	88
6.35.2.57 int_vec_mul . . . . .	88
6.35.2.58 int_vec_muls . . . . .	88
6.35.2.59 int_vec_permute_vect . . . . .	88
6.35.2.60 int_vec_randperm . . . . .	89
6.35.2.61 int_vec_sub . . . . .	89
6.35.2.62 int_vec_subs . . . . .	89
6.35.2.63 int_vec_unique . . . . .	89
6.35.2.64 int_vecstack_creat . . . . .	89
6.35.2.65 int_vecstack_error . . . . .	89
6.35.2.66 int_vecstack_free . . . . .	89
6.35.2.67 mat_2int_vec . . . . .	90
6.35.2.68 mat_abs . . . . .	90
6.35.2.69 mat_add . . . . .	90
6.35.2.70 mat_adds . . . . .	90
6.35.2.71 mat_backsubs1 . . . . .	91
6.35.2.72 mat_bayes_classifier_test . . . . .	91
6.35.2.73 mat_bayes_classifier_train . . . . .	91
6.35.2.74 mat_bayes_model_creat . . . . .	91
6.35.2.75 mat_bayes_model_free . . . . .	91
6.35.2.76 mat_binom . . . . .	91
6.35.2.77 mat_binom_init . . . . .	91
6.35.2.78 mat_bisquare_wt . . . . .	91
6.35.2.79 mat_bs_delete . . . . .	92
6.35.2.80 mat_bs_find . . . . .	92
6.35.2.81 mat_bs_find_max . . . . .	92



6.35.2.82	<code>mat_bs_find_min</code>	92
6.35.2.83	<code>mat_bs_free</code>	92
6.35.2.84	<code>mat_bs_inorder</code>	92
6.35.2.85	<code>mat_bs_insert</code>	92
6.35.2.86	<code>mat_bs_make_null</code>	92
6.35.2.87	<code>mat_bsxfun</code>	92
6.35.2.88	<code>mat_calc_dist_sq</code>	92
6.35.2.89	<code>mat_cart2pol</code>	93
6.35.2.90	<code>mat_cheby</code>	93
6.35.2.91	<code>mat_cheby_approx</code>	93
6.35.2.92	<code>mat_cheby_coeffs_to_poly</code>	93
6.35.2.93	<code>mat_cheby_init</code>	94
6.35.2.94	<code>mat_cholesky</code>	94
6.35.2.95	<code>mat_cofact</code>	94
6.35.2.96	<code>mat_colcopy</code>	94
6.35.2.97	<code>mat_concat</code>	94
6.35.2.98	<code>mat_conjgrad</code>	95
6.35.2.99	<code>mat_conv2</code>	95
6.35.2.100	<code>mat_copy</code>	95
6.35.2.101	<code>mat_corcol</code>	95
6.35.2.102	<code>mat_count_words_in_line</code>	95
6.35.2.103	<code>mat_covcol</code>	96
6.35.2.104	<code>mat_creat</code>	96
6.35.2.105	<code>mat_creat_diag</code>	96
6.35.2.106	<code>mat_det</code>	96
6.35.2.107	<code>mat_diagmul</code>	96
6.35.2.108	<code>mat_div_dot</code>	97
6.35.2.109	<code>mat_divs</code>	97
6.35.2.110	<code>mat_dlmread</code>	97
6.35.2.111	<code>mat_dlmwrite</code>	97
6.35.2.112	<code>mat_dump</code>	97
6.35.2.113	<code>mat_dumppf</code>	98
6.35.2.114	<code>mat_durbin</code>	98
6.35.2.115	<code>mat_eig_sym</code>	98
6.35.2.116	<code>mat_error</code>	98
6.35.2.117	<code>mat_fdump</code>	98
6.35.2.118	<code>mat_fdumppf</code>	98
6.35.2.119	<code>mat_fft2</code>	99
6.35.2.120	<code>mat_fgetmat</code>	99
6.35.2.121	<code>mat_fill</code>	99

6.35.2.122	<code>mat_fill_type</code>	99
6.35.2.123	<code>mat_find</code>	100
6.35.2.124	<code>mat_find_within_dist</code>	100
6.35.2.125	<code>mat_fliplr</code>	100
6.35.2.126	<code>mat_flipud</code>	100
6.35.2.127	<code>mat_fnextline</code>	100
6.35.2.128	<code>mat_free</code>	100
6.35.2.129	<code>mat_get_sub_matrix_from_cols</code>	100
6.35.2.130	<code>mat_get_sub_matrix_from_rows</code>	101
6.35.2.131	<code>mat_get_sub_vector</code>	101
6.35.2.132	<code>mat_gfunc</code>	101
6.35.2.133	<code>mat_go_next_word</code>	101
6.35.2.134	<code>mat_graph_adjlist</code>	102
6.35.2.135	<code>mat_graph_adjm_to_adjl</code>	102
6.35.2.136	<code>mat_graph_creat</code>	102
6.35.2.137	<code>mat_graph_dump</code>	102
6.35.2.138	<code>mat_graph_dumpf</code>	102
6.35.2.139	<code>mat_graph_reverse</code>	102
6.35.2.140	<code>mat_graph_search</code>	102
6.35.2.141	<code>mat_graph_visit</code>	102
6.35.2.142	<code>mat_huber_wt</code>	102
6.35.2.143	<code>mat_innerprod</code>	102
6.35.2.144	<code>mat_int_priorityqueue_creat</code>	102
6.35.2.145	<code>mat_int_priorityqueue_dequeue</code>	102
6.35.2.146	<code>mat_int_priorityqueue_enqueue</code>	102
6.35.2.147	<code>mat_int_priorityqueue_free</code>	102
6.35.2.148	<code>mat_int_priorityqueue_is_empty</code>	102
6.35.2.149	<code>mat_int_priorityqueue_update</code>	102
6.35.2.150	<code>mat_int_qadrat</code>	102
6.35.2.151	<code>mat_int_queue_creat</code>	103
6.35.2.152	<code>mat_int_queue_dequeue</code>	103
6.35.2.153	<code>mat_int_queue_enqueue</code>	103
6.35.2.154	<code>mat_int_queue_free</code>	103
6.35.2.155	<code>mat_int_queue_is_empty</code>	103
6.35.2.156	<code>mat_int_simpson</code>	103
6.35.2.157	<code>mat_int_stack_creat</code>	103
6.35.2.158	<code>mat_int_stack_free</code>	103
6.35.2.159	<code>mat_int_stack_is_empty</code>	103
6.35.2.160	<code>mat_int_stack_pop</code>	103
6.35.2.161	<code>mat_int_stack_push</code>	103

6.35.2.162	<a href="#">mat_int_trapezoid</a>	103
6.35.2.163	<a href="#">mat_inv</a>	104
6.35.2.164	<a href="#">mat_isnumeric</a>	104
6.35.2.165	<a href="#">mat_kdtree_free</a>	104
6.35.2.166	<a href="#">mat_kdtree_k_nearest</a>	104
6.35.2.167	<a href="#">mat_kdtree_make_tree</a>	105
6.35.2.168	<a href="#">mat_kdtree_nearest</a>	105
6.35.2.169	<a href="#">mat_kmeans</a>	105
6.35.2.170	<a href="#">mat_least_squares</a>	105
6.35.2.171	<a href="#">mat_legendre</a>	105
6.35.2.172	<a href="#">mat_legendre_init</a>	105
6.35.2.173	<a href="#">mat_linear_ls_fit</a>	106
6.35.2.174	<a href="#">mat_lsolve</a>	106
6.35.2.175	<a href="#">mat_lsolve_durbin</a>	106
6.35.2.176	<a href="#">mat_lu</a>	106
6.35.2.177	<a href="#">mat_max</a>	106
6.35.2.178	<a href="#">mat_mds</a>	106
6.35.2.179	<a href="#">mat_mean</a>	106
6.35.2.180	<a href="#">mat_mean_col</a>	106
6.35.2.181	<a href="#">mat_mean_row</a>	106
6.35.2.182	<a href="#">mat_median</a>	106
6.35.2.183	<a href="#">mat_min</a>	107
6.35.2.184	<a href="#">mat_minor</a>	107
6.35.2.185	<a href="#">mat_mtype_priorityqueue_creat</a>	107
6.35.2.186	<a href="#">mat_mtype_priorityqueue_dequeue</a>	107
6.35.2.187	<a href="#">mat_mtype_priorityqueue_enqueue</a>	107
6.35.2.188	<a href="#">mat_mtype_priorityqueue_free</a>	107
6.35.2.189	<a href="#">mat_mtype_priorityqueue_is_empty</a>	107
6.35.2.190	<a href="#">mat_mtype_priorityqueue_update</a>	107
6.35.2.191	<a href="#">mat_mtype_queue_creat</a>	107
6.35.2.192	<a href="#">mat_mtype_queue_dequeue</a>	107
6.35.2.193	<a href="#">mat_mtype_queue_enqueue</a>	107
6.35.2.194	<a href="#">mat_mtype_queue_free</a>	107
6.35.2.195	<a href="#">mat_mtype_queue_is_empty</a>	107
6.35.2.196	<a href="#">mat_mtype_stack_creat</a>	107
6.35.2.197	<a href="#">mat_mtype_stack_free</a>	107
6.35.2.198	<a href="#">mat_mtype_stack_is_empty</a>	107
6.35.2.199	<a href="#">mat_mtype_stack_pop</a>	107
6.35.2.200	<a href="#">mat_mtype_stack_push</a>	107
6.35.2.201	<a href="#">mat_mul</a>	107

6.35.2.202mat_mul_dot . . . . .	108
6.35.2.203mat_mul_fast . . . . .	108
6.35.2.204mat_muls . . . . .	108
6.35.2.205mat_nextline . . . . .	108
6.35.2.206mat_norm_inf . . . . .	108
6.35.2.207mat_norm_one . . . . .	108
6.35.2.208mat_norm_p . . . . .	108
6.35.2.209mat_omp . . . . .	108
6.35.2.210mat_order_statistic . . . . .	108
6.35.2.211mat_pca . . . . .	108
6.35.2.212mat_perceptron_creat . . . . .	108
6.35.2.213mat_perceptron_free . . . . .	108
6.35.2.214mat_perceptron_test . . . . .	109
6.35.2.215mat_perceptron_train . . . . .	109
6.35.2.216mat_perceptron_train_ . . . . .	109
6.35.2.217mat_pick_col . . . . .	109
6.35.2.218mat_pick_row . . . . .	109
6.35.2.219mat_pinv . . . . .	109
6.35.2.220mat_pol2cart . . . . .	109
6.35.2.221mat_poly_add . . . . .	110
6.35.2.222mat_poly_diff . . . . .	110
6.35.2.223mat_poly_diff_eval . . . . .	110
6.35.2.224mat_poly_div . . . . .	110
6.35.2.225mat_poly_eval . . . . .	111
6.35.2.226mat_poly_mul . . . . .	111
6.35.2.227mat_poly_scale . . . . .	111
6.35.2.228mat_poly_shift . . . . .	112
6.35.2.229mat_qr . . . . .	112
6.35.2.230mat_qsort . . . . .	112
6.35.2.231mat_rand . . . . .	112
6.35.2.232mat_randexp . . . . .	112
6.35.2.233mat_randfun . . . . .	112
6.35.2.234mat_randn . . . . .	112
6.35.2.235mat_randperm . . . . .	112
6.35.2.236mat_randperm_n . . . . .	112
6.35.2.237mat_read_word . . . . .	113
6.35.2.238mat_reg_inv . . . . .	113
6.35.2.239mat_rob_least_squares . . . . .	113
6.35.2.240mat_robust_fit . . . . .	113
6.35.2.241mat_rowcopy . . . . .	113

6.35.2.242	<a href="#">mat_spcol</a>	113
6.35.2.243	<a href="#">mat_set_seed</a>	113
6.35.2.244	<a href="#">mat_sub</a>	113
6.35.2.245	<a href="#">mat_submat</a>	113
6.35.2.246	<a href="#">mat_subs</a>	114
6.35.2.247	<a href="#">mat_sum</a>	114
6.35.2.248	<a href="#">mat_sum_col</a>	114
6.35.2.249	<a href="#">mat_sum_row</a>	114
6.35.2.250	<a href="#">mat_syntoeplz</a>	114
6.35.2.251	<a href="#">mat_tic</a>	114
6.35.2.252	<a href="#">mat_toc</a>	114
6.35.2.253	<a href="#">mat_toc_print</a>	114
6.35.2.254	<a href="#">mat_tqli</a>	114
6.35.2.255	<a href="#">mat_tran</a>	114
6.35.2.256	<a href="#">mat_tred2</a>	114
6.35.2.257	<a href="#">mat_vectorize</a>	115
6.35.2.258	<a href="#">mat_vectorize_tr</a>	115
6.35.2.259	<a href="#">mat_w_least_squares</a>	115
6.35.2.260	<a href="#">mat_wpinv</a>	115
6.35.2.261	<a href="#">mat_xcopy</a>	115
6.35.2.262	<a href="#">mat_xjoin</a>	116
6.35.2.263	<a href="#">mats_isinf</a>	116
6.35.2.264	<a href="#">mats_isnan</a>	116
6.35.2.265	<a href="#">matstack_append</a>	116
6.35.2.266	<a href="#">matstack_creat</a>	117
6.35.2.267	<a href="#">matstack_error</a>	117
6.35.2.268	<a href="#">matstack_free</a>	117
6.35.2.269	<a href="#">matvec_creat</a>	117
6.35.2.270	<a href="#">matvec_free</a>	117
6.35.2.271	<a href="#">pq_error</a>	118
6.35.2.272	<a href="#">queue_error</a>	118
6.35.2.273	<a href="#">stack_error</a>	118
6.35.3	Variable Documentation	118
6.35.3.1	<a href="#">mat_binom_series_table</a>	118
6.35.3.2	<a href="#">mat_cheby_series_table</a>	118
6.35.3.3	<a href="#">MAT_CLOCK_TIME</a>	118
6.35.3.4	<a href="#">mat_legendre_series_table</a>	118
6.35.3.5	<a href="#">MAT_SEED</a>	118
6.35.3.6	<a href="#">MAT_SET_SEED</a>	118
6.36	<a href="#">matsearch.c File Reference</a>	118

6.36.1	Function Documentation	118
6.36.1.1	int_vec_find	118
6.36.1.2	mat_find	119
6.37	matsolve.c File Reference	119
6.37.1	Function Documentation	119
6.37.1.1	mat_backsubs1	119
6.37.1.2	mat_cholesky	119
6.37.1.3	mat_conjgrad	119
6.37.1.4	mat_lsolve	119
6.37.1.5	mat_lu	119
6.38	matsort.c File Reference	119
6.38.1	Function Documentation	120
6.38.1.1	mat_median	120
6.38.1.2	mat_order_statistic	120
6.38.1.3	mat_qsort	120
6.39	matstdrels.c File Reference	120
6.39.1	Function Documentation	121
6.39.1.1	gen_abs_ceil	121
6.39.1.2	gen_eq	121
6.39.1.3	gen_gt	121
6.39.1.4	gen_lt	121
6.40	matsub.c File Reference	121
6.40.1	Function Documentation	121
6.40.1.1	int_vec_sub	121
6.40.1.2	int_vec_subs	121
6.40.1.3	mat_sub	121
6.40.1.4	mat_subs	121
6.41	matsubx.c File Reference	121
6.41.1	Function Documentation	121
6.41.1.1	mat_submat	121
6.42	matsum.c File Reference	122
6.42.1	Function Documentation	122
6.42.1.1	mat_sum	122
6.42.1.2	mat_sum_col	122
6.42.1.3	mat_sum_row	122
6.43	mattext.c File Reference	122
6.43.1	Function Documentation	122
6.43.1.1	mat_count_words_in_line	122
6.43.1.2	mat_dlmread	123
6.43.1.3	mat_dlmwrite	123

6.43.1.4	<a href="#">mat_go_next_word</a>	123
6.43.1.5	<a href="#">mat_isnumeric</a>	123
6.43.1.6	<a href="#">mat_read_word</a>	123
6.44	<a href="#">mattimers.c File Reference</a>	124
6.44.1	<a href="#">Function Documentation</a>	124
6.44.1.1	<a href="#">mat_tic</a>	124
6.44.1.2	<a href="#">mat_toc</a>	124
6.44.1.3	<a href="#">mat_toc_print</a>	124
6.44.2	<a href="#">Variable Documentation</a>	124
6.44.2.1	<a href="#">MAT_CLOCK_TIME</a>	124
6.45	<a href="#">mattoepz.c File Reference</a>	124
6.45.1	<a href="#">Function Documentation</a>	125
6.45.1.1	<a href="#">mat_symtoepz</a>	125
6.46	<a href="#">mattran.c File Reference</a>	125
6.46.1	<a href="#">Function Documentation</a>	125
6.46.1.1	<a href="#">mat_tran</a>	125
6.47	<a href="#">README.md File Reference</a>	125





# Chapter 1

## Matrixlab

### 1.1 Introduction

Matrixlab is a generic C library for matrix routines. It contains over 250 functions for matrix operations. Many of the functions are multi-threaded.



## Chapter 2

# matrixlab

C Matrix Library



## Chapter 3

# Data Structure Index

### 3.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">mat_bayes_model</a>	Bayes Classifier Model Structure . . . . .	9
<a href="#">mat_gnode</a>	Graph Node Structure . . . . .	10
<a href="#">mat_graph</a>	Graph Structure . . . . .	10
<a href="#">mat_int_priorityqueue</a>	Integer Priority Queue Structure . . . . .	11
<a href="#">mat_int_queue</a>	Integer Queue Structure . . . . .	12
<a href="#">mat_int_stack</a>	Integer Stack Structure . . . . .	13
<a href="#">mat_intpqnode</a>	Integer Priority Queue Node Structure . . . . .	14
<a href="#">mat_kdnode</a>	. . . . .	14
<a href="#">mat_kdtree</a>	. . . . .	15
<a href="#">mat_mtype_priorityqueue</a>	Mtype Priority Queue Structure . . . . .	15
<a href="#">mat_mtype_queue</a>	Mtype Queue Structure . . . . .	16
<a href="#">mat_mtype_stack</a>	Mtype Stack Structure . . . . .	17
<a href="#">mat_mtypepqnode</a>	Mtype Priority Queue Node Structure . . . . .	17
<a href="#">mat_perceptron</a>	Perceptron Classifier Model Structure . . . . .	18
<a href="#">mat_qintnode</a>	Integer Queue Node Structure . . . . .	19
<a href="#">mat_qmtypenode</a>	Mtype Queue Node Structure . . . . .	19
<a href="#">mat_tree_node</a>	Search Tree Node Structure . . . . .	20



## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

<a href="#">matabs.c</a>	21
<a href="#">matadd.c</a>	21
<a href="#">matcompress.c</a>	23
<a href="#">matconcat.c</a>	23
<a href="#">matconv.c</a>	23
<a href="#">matcreat.c</a>	25
<a href="#">matdatastruct.c</a>	33
<a href="#">matdet.c</a>	35
<a href="#">matdiv.c</a>	36
<a href="#">matdump.c</a>	37
<a href="#">matdurn.c</a>	39
<a href="#">materr.c</a>	40
<a href="#">matfft.c</a>	42
<a href="#">matfilter.c</a>	43
<a href="#">matfit.c</a>	43
<a href="#">matflip.c</a>	44
<a href="#">matfuncs.c</a>	44
<a href="#">matgraph.c</a>	49
<a href="#">matinnerprod.c</a>	49
<a href="#">matintegrate.c</a>	50
<a href="#">matinv.c</a>	51
<a href="#">matkdtree.c</a>	51
<a href="#">matmaxmin.c</a>	53
<a href="#">matmds.c</a>	53
<a href="#">matmean.c</a>	53
<a href="#">matmisc.c</a>	54
<a href="#">matmul.c</a>	58
<a href="#">matpca.c</a>	59
<a href="#">matpinv.c</a>	59
<a href="#">matpoly.c</a>	60
<a href="#">matprec.c</a>	65
<a href="#">matpursuit.c</a>	65
<a href="#">matrand.c</a>	65
<a href="#">matrix.c</a>	66
<a href="#">matrix.h</a>	66
<a href="#">matsearch.c</a>	118
<a href="#">matsolve.c</a>	119
<a href="#">matsort.c</a>	119

matstdrels.c	120
matsub.c	121
matsubx.c	121
matsum.c	122
mattext.c	122
mattimers.c	124
mattoepz.c	124
mattran.c	125



## Chapter 5

# Data Structure Documentation

### 5.1 `mat_bayes_model` Struct Reference

Bayes Classifier Model Structure.

```
#include <matrix.h>
```

#### Data Fields

- `int num_of_classes`
- `int num_of_features`
- `INT_VECTOR class_labels`
- `MATRIX class_priors`
- `MATSTACK class_means`
- `MATSTACK class_covars`

#### 5.1.1 Detailed Description

Bayes Classifier Model Structure.

#### 5.1.2 Field Documentation

##### 5.1.2.1 `MATSTACK mat_bayes_model::class_covars`

Training data class covariances

##### 5.1.2.2 `INT_VECTOR mat_bayes_model::class_labels`

Training data class label vector

##### 5.1.2.3 `MATSTACK mat_bayes_model::class_means`

Training data class means

##### 5.1.2.4 `MATRIX mat_bayes_model::class_priors`

Training data prior information

#### 5.1.2.5 `int mat_bayes_model::num_of_classes`

Number of training class

#### 5.1.2.6 `int mat_bayes_model::num_of_features`

Number of training features

The documentation for this struct was generated from the following file:

- [matrix.h](#)

## 5.2 `mat_gnode` Struct Reference

Graph Node Structure.

```
#include <matrix.h>
```

### Data Fields

- `int v`
- `double weight`
- `struct mat_gnode * next`

### 5.2.1 Detailed Description

Graph Node Structure.

### 5.2.2 Field Documentation

#### 5.2.2.1 `struct mat_gnode* mat_gnode::next`

Pointer to next node

#### 5.2.2.2 `int mat_gnode::v`

Value

#### 5.2.2.3 `double mat_gnode::weight`

Node weight

The documentation for this struct was generated from the following file:

- [matrix.h](#)

## 5.3 `mat_graph` Struct Reference

Graph Structure.

```
#include <matrix.h>
```

## Data Fields

- int [nvertices](#)
- int [nedges](#)
- int \* [val](#)
- int \* [vseq](#)
- int [id](#)
- [MAT\\_GNODE](#) \* [adj](#)
- [MAT\\_GNODE](#) [z](#)
- int \* [dad](#)
- int [weighted](#)
- [MAT\\_INT\\_PRIORITYQUEUE](#) [pq](#)

### 5.3.1 Detailed Description

Graph Structure.

### 5.3.2 Field Documentation

5.3.2.1 [MAT\\_GNODE](#)\* [mat\\_graph::adj](#)

5.3.2.2 int\* [mat\\_graph::dad](#)

5.3.2.3 int [mat\\_graph::id](#)

5.3.2.4 int [mat\\_graph::nedges](#)

Number of edges

5.3.2.5 int [mat\\_graph::nvertices](#)

Number of vertices

5.3.2.6 [MAT\\_INT\\_PRIORITYQUEUE](#) [mat\\_graph::pq](#)

5.3.2.7 int\* [mat\\_graph::val](#)

5.3.2.8 int\* [mat\\_graph::vseq](#)

5.3.2.9 int [mat\\_graph::weighted](#)

5.3.2.10 [MAT\\_GNODE](#) [mat\\_graph::z](#)

The documentation for this struct was generated from the following file:

- [matrix.h](#)

## 5.4 mat\_int\_priorityqueue Struct Reference

Integer Priority Queue Structure.

```
#include <matrix.h>
```

## Data Fields

- int [p](#)
- int [type](#)
- int [length](#)
- [MAT\\_INTPQNODE](#) element

### 5.4.1 Detailed Description

Integer Priority Queue Structure.

### 5.4.2 Field Documentation

#### 5.4.2.1 [MAT\\_INTPQNODE](#) `mat_int_priorityqueue::element`

Pointer to priority queue data

#### 5.4.2.2 `int mat_int_priorityqueue::length`

Total allocated priority queue length

#### 5.4.2.3 `int mat_int_priorityqueue::p`

Current priority queue position

#### 5.4.2.4 `int mat_int_priorityqueue::type`

Priority type

The documentation for this struct was generated from the following file:

- [matrix.h](#)

## 5.5 `mat_int_queue` Struct Reference

Integer Queue Structure.

```
#include <matrix.h>
```

## Data Fields

- int [p](#)
- [MAT\\_QINTNODE](#) `head`
- [MAT\\_QINTNODE](#) `tail`

### 5.5.1 Detailed Description

Integer Queue Structure.

## 5.5.2 Field Documentation

### 5.5.2.1 MAT\_QINTNODE mat\_int\_queue::head

Queue head node

### 5.5.2.2 int mat\_int\_queue::p

Current queue position

### 5.5.2.3 MAT\_QINTNODE mat\_int\_queue::tail

Queue tail node

The documentation for this struct was generated from the following file:

- [matrix.h](#)

## 5.6 mat\_int\_stack Struct Reference

Integer Stack Structure.

```
#include <matrix.h>
```

### Data Fields

- int [p](#)
- int [length](#)
- int \* [stack](#)

### 5.6.1 Detailed Description

Integer Stack Structure.

### 5.6.2 Field Documentation

#### 5.6.2.1 int mat\_int\_stack::length

Total allocated stack length

#### 5.6.2.2 int mat\_int\_stack::p

Current stack position

#### 5.6.2.3 int\* mat\_int\_stack::stack

Pointer to stack data

The documentation for this struct was generated from the following file:

- [matrix.h](#)

## 5.7 mat\_intpqnode Struct Reference

Integer Priority Queue Node Structure.

```
#include <matrix.h>
```

### Data Fields

- int [data](#)
- int [priority](#)

### 5.7.1 Detailed Description

Integer Priority Queue Node Structure.

### 5.7.2 Field Documentation

#### 5.7.2.1 int mat\_intpqnode::data

Integer node data

#### 5.7.2.2 int mat\_intpqnode::priority

Node priority

The documentation for this struct was generated from the following file:

- [matrix.h](#)

## 5.8 mat\_kdnode Struct Reference

```
#include <matrix.h>
```

### Data Fields

- mtype [x](#) [MAT\_KDTREE\_MAX\_DIMS]
- int [idx](#)
- struct [mat\\_kdnode](#) \* [left](#)
- struct [mat\\_kdnode](#) \* [right](#)

### 5.8.1 Field Documentation

#### 5.8.1.1 int mat\_kdnode::idx

#### 5.8.1.2 struct mat\_kdnode\* mat\_kdnode::left

#### 5.8.1.3 struct mat\_kdnode \* mat\_kdnode::right

#### 5.8.1.4 mtype mat\_kdnode::x[MAT\_KDTREE\_MAX\_DIMS]

The documentation for this struct was generated from the following file:

- [matrix.h](#)

## 5.9 mat\_kdtree Struct Reference

```
#include <matrix.h>
```

### Data Fields

- int [ndims](#)
- int [length](#)
- int [\\_is\\_allocated](#)
- [MAT\\_KDNODE](#) data
- [MAT\\_KDNODE](#) kdtree

### 5.9.1 Field Documentation

5.9.1.1 int mat\_kdtree::\_is\_allocated

5.9.1.2 [MAT\\_KDNODE](#) mat\_kdtree::data

5.9.1.3 [MAT\\_KDNODE](#) mat\_kdtree::kdtree

5.9.1.4 int mat\_kdtree::length

5.9.1.5 int mat\_kdtree::ndims

The documentation for this struct was generated from the following file:

- [matrix.h](#)

## 5.10 mat\_mtype\_priorityqueue Struct Reference

Mtype Priority Queue Structure.

```
#include <matrix.h>
```

### Data Fields

- int [p](#)
- int [type](#)
- int [length](#)
- [MAT\\_MTYPEPQNODE](#) element

### 5.10.1 Detailed Description

Mtype Priority Queue Structure.

### 5.10.2 Field Documentation

5.10.2.1 [MAT\\_MTYPEPQNODE](#) mat\_mtype\_priorityqueue::element

Pointer to priority queue data

#### 5.10.2.2 `int mat_mtype_priorityqueue::length`

Total allocated priority queue length

#### 5.10.2.3 `int mat_mtype_priorityqueue::p`

Current priority queue position

#### 5.10.2.4 `int mat_mtype_priorityqueue::type`

Priority type

The documentation for this struct was generated from the following file:

- [matrix.h](#)

## 5.11 `mat_mtype_queue` Struct Reference

Mtype Queue Structure.

```
#include <matrix.h>
```

### Data Fields

- `int p`
- [MAT\\_QMTYPENODE head](#)
- [MAT\\_QMTYPENODE tail](#)

### 5.11.1 Detailed Description

Mtype Queue Structure.

### 5.11.2 Field Documentation

#### 5.11.2.1 `MAT_QMTYPENODE mat_mtype_queue::head`

Queue head node

#### 5.11.2.2 `int mat_mtype_queue::p`

Current queue position

#### 5.11.2.3 `MAT_QMTYPENODE mat_mtype_queue::tail`

Queue tail node

The documentation for this struct was generated from the following file:

- [matrix.h](#)



## 5.12 `mat_mtype_stack` Struct Reference

Mtype Stack Structure.

```
#include <matrix.h>
```

### Data Fields

- int `p`
- int `length`
- mtype \* `stack`

#### 5.12.1 Detailed Description

Mtype Stack Structure.

#### 5.12.2 Field Documentation

##### 5.12.2.1 int `mat_mtype_stack::length`

Total allocated stack length

##### 5.12.2.2 int `mat_mtype_stack::p`

Current stack position

##### 5.12.2.3 mtype\* `mat_mtype_stack::stack`

Pointer to stack data

The documentation for this struct was generated from the following file:

- [matrix.h](#)

## 5.13 `mat_mtypepqnode` Struct Reference

Mtype Priority Queue Node Structure.

```
#include <matrix.h>
```

### Data Fields

- mtype `data`
- mtype `priority`

#### 5.13.1 Detailed Description

Mtype Priority Queue Node Structure.

### 5.13.2 Field Documentation

#### 5.13.2.1 mtype mat\_mtypepqnode::data

Mtype node data

#### 5.13.2.2 mtype mat\_mtypepqnode::priority

Node priority

The documentation for this struct was generated from the following file:

- [matrix.h](#)

## 5.14 mat\_perceptron Struct Reference

Perceptron Classifier Model Structure.

```
#include <matrix.h>
```

### Data Fields

- int [num\\_of\\_classes](#)
- int [num\\_of\\_features](#)
- [INT\\_VECTOR](#) [class\\_labels](#)
- [MATRIX](#) [class\\_weights](#)
- int [istrained](#)
- int [num\\_of\\_iterations](#)

### 5.14.1 Detailed Description

Perceptron Classifier Model Structure.

### 5.14.2 Field Documentation

#### 5.14.2.1 [INT\\_VECTOR](#) mat\_perceptron::class\_labels

Training data class label vector

#### 5.14.2.2 [MATRIX](#) mat\_perceptron::class\_weights

Trained Classifier Weights

#### 5.14.2.3 int mat\_perceptron::istrained

Is trained

#### 5.14.2.4 int mat\_perceptron::num\_of\_classes

Number of training classes

5.14.2.5 `int mat_perceptron::num_of_features`

Number of training features

5.14.2.6 `int mat_perceptron::num_of_iterations`

Number of training iterations

The documentation for this struct was generated from the following file:

- [matrix.h](#)

## 5.15 `mat_qintnode` Struct Reference

Integer Queue Node Structure.

```
#include <matrix.h>
```

### Data Fields

- `int data`
- `struct mat_qintnode * next`

### 5.15.1 Detailed Description

Integer Queue Node Structure.

### 5.15.2 Field Documentation

5.15.2.1 `int mat_qintnode::data`

Integer node data

5.15.2.2 `struct mat_qintnode* mat_qintnode::next`

Pointer to next node

The documentation for this struct was generated from the following file:

- [matrix.h](#)

## 5.16 `mat_qmtyphenode` Struct Reference

Mtype Queue Node Structure.

```
#include <matrix.h>
```

### Data Fields

- `mtype data`
- `struct mat_qmtyphenode * next`

### 5.16.1 Detailed Description

Mtype Queue Node Structure.

### 5.16.2 Field Documentation

#### 5.16.2.1 mtype mat\_qmtypenode::data

Mtype node data

#### 5.16.2.2 struct mat\_qmtypenode\* mat\_qmtypenode::next

Pointer to next node

The documentation for this struct was generated from the following file:

- [matrix.h](#)

## 5.17 mat\_tree\_node Struct Reference

Search Tree Node Structure.

```
#include <matrix.h>
```

### Data Fields

- mtype [element](#)
- struct [mat\\_tree\\_node](#) \* [left](#)
- struct [mat\\_tree\\_node](#) \* [right](#)

### 5.17.1 Detailed Description

Search Tree Node Structure.

### 5.17.2 Field Documentation

#### 5.17.2.1 mtype mat\_tree\_node::element

Search tree node data

#### 5.17.2.2 struct mat\_tree\_node\* mat\_tree\_node::left

Pointer to left child node

#### 5.17.2.3 struct mat\_tree\_node\* mat\_tree\_node::right

Pointer to right child node

The documentation for this struct was generated from the following file:

- [matrix.h](#)

## Chapter 6

# File Documentation

### 6.1 matabs.c File Reference

#### Functions

- [MATRIX mat\\_abs](#) ([MATRIX A](#), [MATRIX result](#))  
*Computes absolute value of matrix.*

#### 6.1.1 Function Documentation

##### 6.1.1.1 [MATRIX mat\\_abs](#) ( [MATRIX A](#), [MATRIX result](#) )

Computes absolute value of matrix.

#### Parameters

<a href="#">in</a>	<a href="#">A</a>	Input matrix
<a href="#">in</a>	<a href="#">result</a>	Matrix to store the result

#### Returns

*[abs\(A\)](#)*

### 6.2 matadd.c File Reference

#### Functions

- [MATRIX mat\\_add](#) ([MATRIX A](#), [MATRIX B](#), [MATRIX result](#))  
*Adds two matrices.*
- [MATRIX mat\\_adds](#) ([MATRIX A](#), [mtype s](#), [MATRIX result](#))  
*Adds a scalar to a matrix.*
- [INT\\_VECTOR int\\_vec\\_add](#) ([INT\\_VECTOR A](#), [INT\\_VECTOR B](#), [INT\\_VECTOR result](#))  
*Adds two integer vectors.*
- [INT\\_VECTOR int\\_vec\\_adds](#) ([INT\\_VECTOR A](#), [int s](#), [INT\\_VECTOR result](#))  
*Adds an integer to an integer vector.*

## 6.2.1 Function Documentation

### 6.2.1.1 `INT_VECTOR int_vec_add ( INT_VECTOR A, INT_VECTOR B, INT_VECTOR result )`

Adds two integer vectors.

#### Parameters

<code>in</code>	<code>A</code>	Input vector
<code>in</code>	<code>B</code>	Input vector
<code>in</code>	<code>result</code>	Vector to store the result

#### Returns

**$A + B$**

### 6.2.1.2 `INT_VECTOR int_vec_adds ( INT_VECTOR A, int s, INT_VECTOR result )`

Adds an integer to an integer vector.

#### Parameters

<code>in</code>	<code>A</code>	Input vector
<code>in</code>	<code>s</code>	Input scalar
<code>in</code>	<code>result</code>	Vector to store the result

#### Returns

**$A + s\mathbf{1}$**

### 6.2.1.3 `MATRIX mat_add ( MATRIX A, MATRIX B, MATRIX result )`

Adds two matrices.

#### Parameters

<code>in</code>	<code>A</code>	Input matrix
<code>in</code>	<code>B</code>	Input matrix
<code>in</code>	<code>result</code>	Matrix to store the result

#### Returns

**$A + B$**

### 6.2.1.4 `MATRIX mat_adds ( MATRIX A, mtype s, MATRIX result )`

Adds a scalar to a matrix.

#### Parameters

<code>in</code>	<code>A</code>	Input matrix
<code>in</code>	<code>s</code>	Input scalar
<code>in</code>	<code>result</code>	Matrix to store the result

Returns

$$\mathbf{A} + s\mathbf{1}\mathbf{1}^T$$

## 6.3 matcompress.c File Reference

## 6.4 matconcat.c File Reference

### Functions

- [MATRIX mat\\_concat](#) ([MATRIX A](#), [MATRIX B](#), int *dim*)  
*Concatenates two matrices.*
- [INT\\_VECTOR int\\_vec\\_concat](#) ([INT\\_VECTOR a](#), [INT\\_VECTOR b](#), [INT\\_VECTOR result](#))  
*Concatenates two integer vectors.*

### 6.4.1 Function Documentation

#### 6.4.1.1 INT\_VECTOR int\_vec\_concat ( INT\_VECTOR a, INT\_VECTOR b, INT\_VECTOR result )

Concatenates two integer vectors.

#### Parameters

in	<i>a</i>	Input first vector
in	<i>b</i>	Input second vector
in	<i>dim</i>	Concatenation direction (ROWS/COLS)

Returns

$$\begin{bmatrix} a & b \end{bmatrix} \text{ or } \begin{bmatrix} a \\ b \end{bmatrix}$$

#### 6.4.1.2 MATRIX mat\_concat ( MATRIX A, MATRIX B, int dim )

Concatenates two matrices.

#### Parameters

in	<i>A</i>	Input first matrix
in	<i>B</i>	Input second matrix
in	<i>dim</i>	Concatenation direction (ROWS/COLS)

Returns

$$\begin{bmatrix} A & B \end{bmatrix} \text{ or } \begin{bmatrix} A \\ B \end{bmatrix}$$

## 6.5 matconv.c File Reference

### Functions

- [INT\\_VECTOR mat\\_2int\\_vec](#) ([MATRIX A](#))  
*Converts a matrix to an integer vector.*

- **MATRIX** `int_vec2_mat` (**INT\_VECTOR** *a*, int *dir*)  
*Converts an integer vector to a matrix.*
- **MATRIX** `mat_vectorize` (**MATRIX** *A*, **MATRIX** *result*)  
*Reshapes a matrix to a vector.*
- **MATRIX** `mat_vectorize_tr` (**MATRIX** *A*, **MATRIX** *result*)  
*Reshapes transpose of a matrix to a vector.*

## 6.5.1 Function Documentation

### 6.5.1.1 **MATRIX** `int_vec2_mat` ( **INT\_VECTOR** *a*, int *dir* )

Converts an integer vector to a matrix.

#### Parameters

<i>in</i>	<i>a</i>	Input vector
<i>in</i>	<i>dir</i>	Conversion direction

#### Returns

Output matrix

### 6.5.1.2 **INT\_VECTOR** `mat_2int_vec` ( **MATRIX** *A* )

Converts a matrix to an integer vector.

#### Parameters

<i>in</i>	<i>A</i>	Input matrix
<i>out</i>	<i>v</i>	Output vector

#### Returns

Output vector

### 6.5.1.3 **MATRIX** `mat_vectorize` ( **MATRIX** *A*, **MATRIX** *result* )

Reshapes a matrix to a vector.

#### Parameters

<i>in</i>	<i>A</i>	Input matrix
<i>in</i>	<i>result</i>	Matrix to store the result

#### Returns

$vec(\mathbf{A})$

### 6.5.1.4 **MATRIX** `mat_vectorize_tr` ( **MATRIX** *A*, **MATRIX** *result* )

Reshapes transpose of a matrix to a vector.



## Parameters

<code>in</code>	<code>A</code>	Input matrix
<code>in</code>	<code>result</code>	Matrix to store the result

## Returns

$vec(\mathbf{A}^T)$

## 6.6 matcreat.c File Reference

## Functions

- [MATRIX mat\\_creat](#) (int row, int col, int type)  
*Creates a matrix.*
- [MATSTACK matstack\\_creat](#) (int len)  
*Creates a matrix stack.*
- [MATSTACK matstack\\_append](#) (MATSTACK s, MATRIX A)  
*Appends a matrix to a matrix stack.*
- int [matstack\\_free](#) (MATSTACK A)  
*Frees a matrix stack.*
- [MATRIX mat\\_fill](#) (MATRIX A, mtype val)  
*Fills a matrix with a value.*
- [MATRIX mat\\_fill\\_type](#) (MATRIX A, int type)  
*Fills a matrix to a type.*
- int [mat\\_free](#) (MATRIX A)  
*Frees a matrix.*
- [INT\\_VECTOR int\\_vec\\_creat](#) (int len, int type)  
*Creates an integer vector.*
- [INT\\_VECTOR int\\_vec\\_fill](#) (INT\_VECTOR A, int val)  
*Fills an integer vector with a value.*
- [INT\\_VECTOR int\\_vec\\_fill\\_type](#) (INT\_VECTOR A, int type)  
*Fills an integer vector to a type.*
- int [int\\_vec\\_free](#) (INT\_VECTOR A)  
*Frees an integer vector.*
- [INT\\_VECSTACK int\\_vecstack\\_creat](#) (int len)  
*Creates an integer vector stack.*
- int [int\\_vecstack\\_free](#) (INT\_VECSTACK A)  
*Frees an integer vector stack.*
- [MAT\\_BAYES\\_MODEL mat\\_bayes\\_model\\_creat](#) (void)  
*Creates a Bayes model.*
- int [mat\\_bayes\\_model\\_free](#) (MAT\_BAYES\_MODEL a)  
*Frees a Bayes model.*
- [MAT\\_PERCEPTRON mat\\_perceptron\\_creat](#) (void)  
*Creates a perceptron.*
- int [mat\\_perceptron\\_free](#) (MAT\_PERCEPTRON a)  
*Frees a perceptron.*
- [MATVEC\\_DPOINTER matvec\\_creat](#) (void)  
*Creates a matrix-vector pair.*
- int [matvec\\_free](#) (MATVEC\_DPOINTER a)  
*Frees a matrix-vector pair.*

- `INT_VECTOR int_vec_append (INT_VECTOR a, int i)`  
*Appends an integer to an integer vector.*
- `INT_VECTOR int_vec_copy (INT_VECTOR a, INT_VECTOR result)`  
*Copies an integer vector.*
- `MATRIX mat_copy (MATRIX A, MATRIX result)`  
*Copies a matrix.*
- `MATRIX mat_xcopy (MATRIX A, int si, int ei, int sj, int ej, MATRIX result)`  
*Copies a sub-matrix.*
- `MATRIX mat_xjoin (MATRIX A11, MATRIX A12, MATRIX A21, MATRIX A22, MATRIX result)`  
*Copies a sub-matrix.*
- `MATRIX mat_rowcopy (MATRIX A, int rowa, int rowb, MATRIX result)`  
*Copies a row from a matrix.*
- `MATRIX mat_colcopy (MATRIX A, int cola, int colb, MATRIX result)`  
*Copies a column from a matrix.*
- `int mat_fgetmat (MATRIX A, MAT_FILEPOINTER fp)`  
*Gets matrix data from opened file.*
- `MATRIX mat_creat_diag (MATRIX diag_vals, MATRIX result)`  
*Creates a diagonal matrix from a 1-d matrix.*

## 6.6.1 Function Documentation

### 6.6.1.1 `INT_VECTOR int_vec_append ( INT_VECTOR a, int i )`

Appends an integer to an integer vector.

#### Parameters

<code>in</code>	<code>a</code>	Input vector
<code>in</code>	<code>i</code>	Integer to append

#### Returns

Appended vector

### 6.6.1.2 `INT_VECTOR int_vec_copy ( INT_VECTOR a, INT_VECTOR result )`

Copies an integer vector.

#### Parameters

<code>in</code>	<code>a</code>	Input vector
<code>in</code>	<code>result</code>	Vector to store the result

#### Returns

Output vector

### 6.6.1.3 `INT_VECTOR int_vec_creat ( int len, int type )`

Creates an integer vector.

## Parameters

<i>in</i>	<i>len</i>	Length of the vector
<i>in</i>	<i>type</i>	Definition type (UNDEFINED/ZERO_INT_VECTOR/ONES_INT_VECTOR/SERIES_INT_VECTOR)

## Returns

Output vector

6.6.1.4 **INT\_VECTOR** int\_vec\_fill ( **INT\_VECTOR** A, int val )

Fills an integer vector with a value.

## Parameters

<i>in</i>	<i>A</i>	Input vector
<i>in</i>	<i>val</i>	Value to fill with

## Returns

Filled vector

6.6.1.5 **INT\_VECTOR** int\_vec\_fill\_type ( **INT\_VECTOR** A, int type )

Fills an integer vector to a type.

## Parameters

<i>in</i>	<i>A</i>	Input vector
<i>in</i>	<i>type</i>	Definition type (UNDEFINED/ZERO_INT_VECTOR/ONES_INT_VECTOR/SERIES_INT_VECTOR)

## Returns

Filled vector

6.6.1.6 **int** int\_vec\_free ( **INT\_VECTOR** A )

Frees an integer vector.

## Parameters

<i>in</i>	<i>A</i>	Input vector
-----------	----------	--------------

## Returns

Success

6.6.1.7 **INT\_VECSTACK** int\_vecstack\_creat ( int len )

Creates an integer vector stack.

## Parameters

<i>in</i>	<i>len</i>	Length of the stack
-----------	------------	---------------------

## Returns

Output vector stack

### 6.6.1.8 `int int_vecstack_free ( INT_VECSTACK A )`

Frees an integer vector stack.

## Parameters

<i>in</i>	<i>A</i>	Input vector stack
-----------	----------	--------------------

## Returns

Success

### 6.6.1.9 `MAT_BAYES_MODEL mat_bayes_model_creat ( void )`

Creates a Bayes model.

## Returns

Output Bayes model

### 6.6.1.10 `int mat_bayes_model_free ( MAT_BAYES_MODEL a )`

Frees a Bayes model.

## Parameters

<i>in</i>	<i>a</i>	Input Bayes model
-----------	----------	-------------------

## Returns

Success

### 6.6.1.11 `MATRIX mat_colcopy ( MATRIX A, int cola, int colb, MATRIX result )`

Copies a column from a matrix.

## Parameters

<i>in</i>	<i>A</i>	Input matrix
<i>in</i>	<i>cola</i>	Source column
<i>in</i>	<i>colb</i>	Destination column
<i>in</i>	<i>result</i>	Matrix to store the result

## Returns

Copied matrix

6.6.1.12 **MATRIX** mat\_copy ( **MATRIX** *A*, **MATRIX** *result* )

Copies a matrix.

## Parameters

<i>in</i>	<i>A</i>	Input matrix
<i>in</i>	<i>result</i>	Matrix to store the result

## Returns

Output matrix

6.6.1.13 **MATRIX** mat\_creat ( *int row*, *int col*, *int type* )

Creates a matrix.

## Parameters

<i>in</i>	<i>row</i>	Number of rows
<i>in</i>	<i>col</i>	Number of columns
<i>in</i>	<i>type</i>	Definition type (UNDEFINED/ZERO_MATRIX/UNIT_MATRIX/ONES_MATRIX)

## Returns

Output matrix

6.6.1.14 **MATRIX** mat\_creat\_diag ( **MATRIX** *diag\_vals*, **MATRIX** *result* )

Creates a diagonal matrix from a 1-d matrix.

## Parameters

<i>in</i>	<i>diag_vals</i>	Input 1-d diagonal value matrix
<i>in</i>	<i>result</i>	Matrix to store the result

## Returns

Diagonal matrix

6.6.1.15 *int* mat\_fgetmat ( **MATRIX** *A*, *MAT\_FILEPOINTER fp* )

Gets matrix data from opened file.

## Parameters

<i>in</i>	<i>A</i>	Matrix to store the data
<i>in</i>	<i>fp</i>	Pointer to opened file

**Returns**

Number of elements copied

**6.6.1.16 MATRIX mat\_fill ( MATRIX A, mtype val )**

Fills a matrix with a value.

**Parameters**

in	A	Input matrix
in	val	Value to fill with

**Returns**

Filled matrix

**6.6.1.17 MATRIX mat\_fill\_type ( MATRIX A, int type )**

Fills a matrix to a type.

**Parameters**

in	A	Input matrix
in	type	Fill type (UNDEFINED/ZERO_MATRIX/UNIT_MATRIX/ONES_MATRIX)

**Returns**

Filled matrix

**6.6.1.18 int mat\_free ( MATRIX A )**

Frees a matrix.

**Parameters**

in	A	Input matrix
----	---	--------------

**Returns**

Success

**6.6.1.19 MAT\_PERCEPTRON mat\_perceptron\_creat ( void )**

Creates a perceptron.

**Returns**

Output perceptron

**6.6.1.20 int mat\_perceptron\_free ( MAT\_PERCEPTRON a )**

Frees a perceptron.

## Parameters

in	<i>a</i>	Input perceptron
----	----------	------------------

## Returns

Success

### 6.6.1.21 **MATRIX** mat\_rowcopy ( **MATRIX** *A*, int *rowa*, int *rowb*, **MATRIX** *result* )

Copies a row from a matrix.

## Parameters

in	<i>A</i>	Input matrix
in	<i>rowa</i>	Source row
in	<i>rowb</i>	Destination row
in	<i>result</i>	Matrix to store the result

## Returns

Copied matrix

### 6.6.1.22 **MATRIX** mat\_xcopy ( **MATRIX** *A*, int *si*, int *ei*, int *sj*, int *ej*, **MATRIX** *result* )

Copies a sub-matrix.

## Parameters

in	<i>A</i>	Input matrix
in	<i>si</i>	Start of first index, $s_i$
in	<i>ei</i>	End of first index, $e_i$
in	<i>sj</i>	Start of second index, $s_j$
in	<i>ej</i>	End of second index, $e_j$
in	<i>result</i>	Matrix to store the result

## Returns

Extracted matrix  $A_{s_i:e_i,s_j:e_j}$

### 6.6.1.23 **MATRIX** mat\_xjoin ( **MATRIX** *A11*, **MATRIX** *A12*, **MATRIX** *A21*, **MATRIX** *A22*, **MATRIX** *result* )

Copies a sub-matrix.

## Parameters

in	<i>A11</i>	Input matrix, $A_{11}$
in	<i>A12</i>	Input matrix, $A_{12}$
in	<i>A21</i>	Input matrix, $A_{21}$
in	<i>A22</i>	Input matrix, $A_{22}$
in	<i>result</i>	Matrix to store the result

**Returns**

Block matrix  $\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$

**6.6.1.24 MATSTACK matstack\_append ( MATSTACK s, MATRIX A )**

Appends a matrix to a matrix stack.

**Parameters**

<i>in</i>	<i>s</i>	Input matrix stack
<i>in</i>	<i>A</i>	Input matrix to append

**Returns**

Output matrix stack

**6.6.1.25 MATSTACK matstack\_creat ( int len )**

Creates a matrix stack.

**Parameters**

<i>in</i>	<i>len</i>	Length of the stack
-----------	------------	---------------------

**Returns**

Output matrix stack

**6.6.1.26 int matstack\_free ( MATSTACK A )**

Frees a matrix stack.

**Parameters**

<i>in</i>	<i>A</i>	Input matrix stack
-----------	----------	--------------------

**Returns**

Success

**6.6.1.27 MATVEC\_DPOINTER matvec\_creat ( void )**

Creates a matrix-vector pair.

**Returns**

Output matrix-vector pair

**6.6.1.28 int matvec\_free ( MATVEC\_DPOINTER a )**

Frees a matrix-vector pair.



## Parameters

in	a	Input matrix-vector pair
----	---	--------------------------

## Returns

Success

## 6.7 matdatastruct.c File Reference

### Functions

- [MAT\\_TREE mat\\_bs\\_make\\_null](#) (void)
- [MAT\\_TREE mat\\_bs\\_free](#) (MAT\_TREE T)
- [MAT\\_TREE mat\\_bs\\_find](#) (mtype x, MAT\_TREE T)
- [MAT\\_TREE mat\\_bs\\_find\\_min](#) (MAT\_TREE T)
- [MAT\\_TREE mat\\_bs\\_find\\_max](#) (MAT\_TREE T)
- [MAT\\_TREE mat\\_bs\\_insert](#) (mtype x, MAT\_TREE T)
- [MAT\\_TREE mat\\_bs\\_delete](#) (mtype x, MAT\_TREE T)
- [int mat\\_bs\\_inorder](#) (MAT\_TREE T, int index, mtype \*\*p\_ordered)
- [MAT\\_INT\\_STACK mat\\_int\\_stack\\_creat](#) (void)
- [int mat\\_int\\_stack\\_free](#) (MAT\_INT\_STACK s)
- [void mat\\_int\\_stack\\_push](#) (MAT\_INT\_STACK s, int value)
- [int mat\\_int\\_stack\\_pop](#) (MAT\_INT\_STACK s)
- [int mat\\_int\\_stack\\_is\\_empty](#) (MAT\_INT\_STACK s)
- [MAT\\_MTYPE\\_STACK mat\\_mtype\\_stack\\_creat](#) (void)
- [int mat\\_mtype\\_stack\\_free](#) (MAT\_MTYPE\_STACK s)
- [void mat\\_mtype\\_stack\\_push](#) (MAT\_MTYPE\_STACK s, mtype value)
- [mtype mat\\_mtype\\_stack\\_pop](#) (MAT\_MTYPE\_STACK s)
- [int mat\\_mtype\\_stack\\_is\\_empty](#) (MAT\_MTYPE\_STACK s)
- [MAT\\_INT\\_QUEUE mat\\_int\\_queue\\_creat](#) (void)
- [int mat\\_int\\_queue\\_free](#) (MAT\_INT\_QUEUE s)
- [void mat\\_int\\_queue\\_enqueue](#) (MAT\_INT\_QUEUE s, int value)
- [int mat\\_int\\_queue\\_dequeue](#) (MAT\_INT\_QUEUE s)
- [int mat\\_int\\_queue\\_is\\_empty](#) (MAT\_INT\_QUEUE s)
- [MAT\\_MTYPE\\_QUEUE mat\\_mtype\\_queue\\_creat](#) (void)
- [int mat\\_mtype\\_queue\\_free](#) (MAT\_MTYPE\_QUEUE s)
- [void mat\\_mtype\\_queue\\_enqueue](#) (MAT\_MTYPE\_QUEUE s, mtype value)
- [mtype mat\\_mtype\\_queue\\_dequeue](#) (MAT\_MTYPE\_QUEUE s)
- [int mat\\_mtype\\_queue\\_is\\_empty](#) (MAT\_MTYPE\_QUEUE s)
- [MAT\\_INT\\_PRIORITYQUEUE mat\\_int\\_priorityqueue\\_creat](#) (int type)
- [void mat\\_int\\_priorityqueue\\_enqueue](#) (MAT\_INT\_PRIORITYQUEUE H, int data, int priority)
- [mat\\_intpqnode mat\\_int\\_priorityqueue\\_dequeue](#) (MAT\_INT\_PRIORITYQUEUE H)
- [int mat\\_int\\_priorityqueue\\_free](#) (MAT\_INT\_PRIORITYQUEUE H)
- [int mat\\_int\\_priorityqueue\\_update](#) (MAT\_INT\_PRIORITYQUEUE H, int data, int priority, int type)
- [int mat\\_int\\_priorityqueue\\_is\\_empty](#) (MAT\_INT\_PRIORITYQUEUE H)
- [MAT\\_MTYPE\\_PRIORITYQUEUE mat\\_mtype\\_priorityqueue\\_creat](#) (int type)
- [void mat\\_mtype\\_priorityqueue\\_enqueue](#) (MAT\_MTYPE\_PRIORITYQUEUE H, mtype data, mtype priority)
- [mat\\_mtypepqnode mat\\_mtype\\_priorityqueue\\_dequeue](#) (MAT\_MTYPE\_PRIORITYQUEUE H)
- [int mat\\_mtype\\_priorityqueue\\_free](#) (MAT\_MTYPE\_PRIORITYQUEUE H)
- [int mat\\_mtype\\_priorityqueue\\_update](#) (MAT\_MTYPE\_PRIORITYQUEUE H, mtype data, mtype priority, int type)
- [int mat\\_mtype\\_priorityqueue\\_is\\_empty](#) (MAT\_MTYPE\_PRIORITYQUEUE H)

## 6.7.1 Function Documentation

6.7.1.1 **MAT\_TREE** `mat_bs.delete ( mtype x, MAT_TREE T )`

6.7.1.2 **MAT\_TREE** `mat_bs.find ( mtype x, MAT_TREE T )`

6.7.1.3 **MAT\_TREE** `mat_bs.find_max ( MAT_TREE T )`

6.7.1.4 **MAT\_TREE** `mat_bs.find_min ( MAT_TREE T )`

6.7.1.5 **MAT\_TREE** `mat_bs.free ( MAT_TREE T )`

6.7.1.6 **int** `mat_bs.inorder ( MAT_TREE T, int index, mtype ** p_ordered )`

6.7.1.7 **MAT\_TREE** `mat_bs.insert ( mtype x, MAT_TREE T )`

6.7.1.8 **MAT\_TREE** `mat_bs.make_null ( void )`

6.7.1.9 **MAT\_INT\_PRIORITYQUEUE** `mat_int.priorityqueue.creat ( int type )`

6.7.1.10 **mat\_intpqnode** `mat_int.priorityqueue.dequeue ( MAT_INT_PRIORITYQUEUE H )`

6.7.1.11 **void** `mat_int.priorityqueue.enqueue ( MAT_INT_PRIORITYQUEUE H, int data, int priority )`

6.7.1.12 **int** `mat_int.priorityqueue.free ( MAT_INT_PRIORITYQUEUE H )`

6.7.1.13 **int** `mat_int.priorityqueue.is_empty ( MAT_INT_PRIORITYQUEUE H )`

6.7.1.14 **int** `mat_int.priorityqueue.update ( MAT_INT_PRIORITYQUEUE H, int data, int priority, int type )`

6.7.1.15 **MAT\_INT\_QUEUE** `mat_int.queue.creat ( void )`

6.7.1.16 **int** `mat_int.queue.dequeue ( MAT_INT_QUEUE s )`

6.7.1.17 **void** `mat_int.queue.enqueue ( MAT_INT_QUEUE s, int value )`

6.7.1.18 **int** `mat_int.queue.free ( MAT_INT_QUEUE s )`

6.7.1.19 **int** `mat_int.queue.is_empty ( MAT_INT_QUEUE s )`

6.7.1.20 **MAT\_INT\_STACK** `mat_int.stack.creat ( void )`

6.7.1.21 **int** `mat_int.stack.free ( MAT_INT_STACK s )`

6.7.1.22 **int** `mat_int.stack.is_empty ( MAT_INT_STACK s )`

6.7.1.23 **int** `mat_int.stack.pop ( MAT_INT_STACK s )`

6.7.1.24 **void** `mat_int.stack.push ( MAT_INT_STACK s, int value )`

6.7.1.25 **MAT\_MTYPE\_PRIORITYQUEUE** `mat_mtype.priorityqueue.creat ( int type )`

6.7.1.26 **mat\_mtypepqnode** `mat_mtype.priorityqueue.dequeue ( MAT_MTYPE_PRIORITYQUEUE H )`

6.7.1.27 **void** `mat_mtype.priorityqueue.enqueue ( MAT_MTYPE_PRIORITYQUEUE H, mtype data, mtype priority )`

- 6.7.1.28 `int mat_mtype_priorityqueue_free ( MAT_MTYPE_PRIORITYQUEUE H )`
- 6.7.1.29 `int mat_mtype_priorityqueue_is_empty ( MAT_MTYPE_PRIORITYQUEUE H )`
- 6.7.1.30 `int mat_mtype_priorityqueue_update ( MAT_MTYPE_PRIORITYQUEUE H, mtype data, mtype priority, int type )`
- 6.7.1.31 `MAT_MTYPE_QUEUE mat_mtype_queue_creat ( void )`
- 6.7.1.32 `mtype mat_mtype_queue_dequeue ( MAT_MTYPE_QUEUE s )`
- 6.7.1.33 `void mat_mtype_queue_enqueue ( MAT_MTYPE_QUEUE s, mtype value )`
- 6.7.1.34 `int mat_mtype_queue_free ( MAT_MTYPE_QUEUE s )`
- 6.7.1.35 `int mat_mtype_queue_is_empty ( MAT_MTYPE_QUEUE s )`
- 6.7.1.36 `MAT_MTYPE_STACK mat_mtype_stack_creat ( void )`
- 6.7.1.37 `int mat_mtype_stack_free ( MAT_MTYPE_STACK s )`
- 6.7.1.38 `int mat_mtype_stack_is_empty ( MAT_MTYPE_STACK s )`
- 6.7.1.39 `mtype mat_mtype_stack_pop ( MAT_MTYPE_STACK s )`
- 6.7.1.40 `void mat_mtype_stack_push ( MAT_MTYPE_STACK s, mtype value )`

## 6.8 matdet.c File Reference

### Functions

- mtype [mat\\_minor](#) (MATRIX *A*, int *i*, int *j*)  
*Computes a minor of a matrix.*
- mtype [mat\\_cofact](#) (MATRIX *A*, int *i*, int *j*)  
*Computes a cofactor of a matrix.*
- mtype [mat\\_det](#) (MATRIX *A*)  
*Computes the determinant of a matrix.*

### 6.8.1 Function Documentation

#### 6.8.1.1 mtype mat\_cofact ( MATRIX *A*, int *i*, int *j* )

Computes a cofactor of a matrix.

#### Parameters

<code>in</code>	<i>A</i>	Input matrix
<code>in</code>	<i>i</i>	Row index
<code>in</code>	<i>j</i>	Column index

#### Returns

Cofactor  $C_{ij}$

### 6.8.1.2 mtype mat\_det ( MATRIX A )

Computes the determinant of a matrix.

#### Parameters

in	A	Input matrix
----	---	--------------

#### Returns

$\det(A)$

### 6.8.1.3 mtype mat\_minor ( MATRIX A, int i, int j )

Computes a minor of a matrix.

#### Parameters

in	A	Input matrix
in	i	Row index
in	j	Column index

#### Returns

Minor  $M_{ij}$

## 6.9 matdiv.c File Reference

### Functions

- [MATRIX mat\\_div\\_dot \(MATRIX A, MATRIX B, MATRIX result\)](#)  
*Computes element-wise matrix division.*
- [MATRIX mat\\_divs \(MATRIX A, mtype s, MATRIX result\)](#)  
*Divides a matrix by a scalar.*
- [INT\\_VECTOR int\\_vec\\_div \(INT\\_VECTOR A, INT\\_VECTOR B, INT\\_VECTOR result\)](#)  
*Computes element-wise integer vector division.*
- [INT\\_VECTOR int\\_vec\\_divs \(INT\\_VECTOR A, int x, INT\\_VECTOR result\)](#)  
*Divides an integer vector by a scalar.*

### 6.9.1 Function Documentation

#### 6.9.1.1 INT\_VECTOR int\_vec\_div ( INT\_VECTOR A, INT\_VECTOR B, INT\_VECTOR result )

Computes element-wise integer vector division.

#### Parameters

in	A	First input vector
in	B	Second input vector
in	result	Vector to store the result

## Returns

$$A./B$$

## 6.9.1.2 INT\_VECTOR int\_vec\_divs ( INT\_VECTOR A, int x, INT\_VECTOR result )

Divides an integer vector by a scalar.

## Parameters

in	A	Input vector
in	s	Scalar
in	result	Vector to store the result

## Returns

$$\frac{A}{s}$$

## 6.9.1.3 MATRIX mat\_div\_dot ( MATRIX A, MATRIX B, MATRIX result )

Computes element-wise matrix division.

## Parameters

in	A	First input matrix
in	B	Second input matrix
in	result	Matrix to store the result

## Returns

$$A./B$$

## 6.9.1.4 MATRIX mat\_divs ( MATRIX A, mtype s, MATRIX result )

Divides a matrix by a scalar.

## Parameters

in	A	Input matrix
in	s	Scalar
in	result	Matrix to store the result

## Returns

$$\frac{A}{s}$$

## 6.10 matdump.c File Reference

## Functions

- void [mat\\_dump](#) (MATRIX A)  
Dumps a matrix in the stdout.

- void `mat_dumpf` (`MATRIX A`, `const char *s`)  
*Dumps a matrix using a given format specifier in the stdout.*
- void `mat_fdump` (`MATRIX A`, `MAT_FILEPOINTER fp`)  
*Dumps a matrix in an opened file.*
- void `mat_dumpf` (`MATRIX A`, `const char *s`, `MAT_FILEPOINTER fp`)  
*Dumps a matrix using a given format specifier in an opened file.*
- void `int_vec_dump` (`INT_VECTOR A`)  
*Dumps an integer vector in the stdout.*
- void `int_vec_dumpf` (`INT_VECTOR A`, `const char *s`)  
*Dumps an integer vector using a given format specifier in the stdout.*
- void `int_vec_fdump` (`INT_VECTOR A`, `MAT_FILEPOINTER fp`)  
*Dumps an integer vector in an opened file.*
- void `int_vec_dumpf` (`INT_VECTOR A`, `const char *s`, `MAT_FILEPOINTER fp`)  
*Dumps an integer vector using a given format specifier in an opened file.*

## 6.10.1 Function Documentation

### 6.10.1.1 void `int_vec.dump` ( `INT_VECTOR A` )

Dumps an integer vector in the stdout.

#### Parameters

<code>in</code>	<code>A</code>	Input vector
-----------------	----------------	--------------

### 6.10.1.2 void `int_vec.dumpf` ( `INT_VECTOR A`, `const char * s` )

Dumps an integer vector using a given format specifier in the stdout.

#### Parameters

<code>in</code>	<code>A</code>	Input vector
<code>in</code>	<code>s</code>	Format specifier

### 6.10.1.3 void `int_vec.fdump` ( `INT_VECTOR A`, `MAT_FILEPOINTER fp` )

Dumps an integer vector in an opened file.

#### Parameters

<code>in</code>	<code>A</code>	Input vector
<code>in</code>	<code>fp</code>	Pointer to an opened file

### 6.10.1.4 void `int_vec.fdumpf` ( `INT_VECTOR A`, `const char * s`, `MAT_FILEPOINTER fp` )

Dumps an integer vector using a given format specifier in an opened file.

#### Parameters

<code>in</code>	<code>A</code>	Input vector
<code>in</code>	<code>s</code>	Format specifier
<code>in</code>	<code>fp</code>	Pointer to an opened file

## 6.10.1.5 void mat\_dump ( MATRIX A )

Dumps a matrix in the stdout.

## Parameters

in	A	Input matrix
----	---	--------------

## 6.10.1.6 void mat\_dumpf ( MATRIX A, const char \* s )

Dumps a matrix using a given format specifier in the stdout.

## Parameters

in	A	Input matrix
in	s	Format specifier

## 6.10.1.7 void mat\_fdump ( MATRIX A, MAT\_FILEPOINTER fp )

Dumps a matrix in an opened file.

## Parameters

in	A	Input matrix
in	fp	Pointer to an opened file

## 6.10.1.8 void mat\_fdumpf ( MATRIX A, const char \* s, MAT\_FILEPOINTER fp )

Dumps a matrix using a given format specifier in an opened file.

## Parameters

in	A	Input matrix
in	s	Format specifier
in	fp	Pointer to an opened file

## 6.11 matdurbn.c File Reference

## Functions

- [MATRIX mat\\_durbin](#) (MATRIX R, MATRIX result)  
*Runs Levinson-Durbin algorithm.*
- [MATRIX mat\\_ksolve\\_durbin](#) (MATRIX A, MATRIX B, MATRIX result)  
*Runs Levinson-Durbin algorithm.*
- [MATSTACK mat\\_qr](#) (MATRIX A, MATSTACK qr)  
*Computes QR decomposition.*

## 6.11.1 Function Documentation

## 6.11.1.1 MATRIX mat\_durbin ( MATRIX R, MATRIX result )

Runs Levinson-Durbin algorithm.

## Parameters

in	<i>R</i>	Input $n^th$ correlation matrix $(n+1) \times 1$
in	<i>result</i>	Matrix to store the result

## Returns

$$X \text{ where } \tilde{R}X = B, \tilde{R} = \begin{bmatrix} R[0][0] & R[1][0] & \cdots & R[n-1][0] \\ R[1][0] & R[0][0] & \cdots & R[n-2][0] \\ \vdots & \vdots & \ddots & \vdots \\ R[n-1][0] & R[n-2][0] & \cdots & R[0][0] \end{bmatrix} \text{ and } B = [R[1][0] \ R[2][0] \ \cdots \ R[n][0]]$$

## 6.11.1.2 MATRIX mat\_solve\_durbin ( MATRIX A, MATRIX B, MATRIX result )

Runs Levinson-Durbin algorithm.

## Parameters

in	<i>A</i>	Input correlation matrix $A = \begin{bmatrix} r_0 & r_1 & \cdots & r_{n-1} \\ r_1 & r_0 & \cdots & r_{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ r_{n-1} & r_{n-2} & \cdots & r_0 \end{bmatrix}$
in	<i>B</i>	Input correlation matrix $B = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \end{bmatrix}$
in	<i>result</i>	Matrix to store the result

## Returns

$X$  where  $RX = B$

## 6.11.1.3 MATSTACK mat\_qr ( MATRIX A, MATSTACK qr )

Computes QR decomposition.

## Parameters

in	<i>A</i>	Input matrix
in	<i>qr</i>	Matrix stack to store result

## Returns

Output QR Matrix stack

## 6.12 materr.c File Reference

## Functions

- int [gen\\_error](#) (int err\_)  
Generates error message for general errors and exits.
- [MATRIX mat\\_error](#) (int err\_)  
Generates error message for matrix errors and exits.
- [MATSTACK matstack\\_error](#) (int err\_)



- Generates error message for matrix stack errors and exits.*
- `INT_VECTOR int_vec_error (int err_)`  
*Generates error message for integer vector errors and exits.*
- `INT_VECSTACK int_vecstack_error (int err_)`  
*Generates error message for integer vector stack errors and exits.*
- `int stack_error (int err_)`  
*Generates error message for stack errors and exits.*
- `int queue_error (int err_)`  
*Generates error message for queue errors and exits.*
- `int pq_error (int err_)`  
*Generates error message for priority queue errors and exits.*
- `int graph_error (int err_)`  
*Generates error message for graph errors and exits.*

## 6.12.1 Function Documentation

### 6.12.1.1 `int gen_error ( int err_ )`

Generates error message for general errors and exits.

#### Parameters

<code>in</code>	<code>err</code>	Error type (GEN_NOT_CONVERGED/GEN_FNOTOPEN/ GEN_FNOTGETMAT/GEN_SIZEMISMATCH/GEN_MATH_ERROR/GEN_MALLOC/GEN_NOT_FOUND/GEN_SIZE_ERROR/GEN_BAD_TYPE)
-----------------	------------------	--

### 6.12.1.2 `int graph_error ( int err_ )`

Generates error message for graph errors and exits.

#### Parameters

<code>in</code>	<code>err</code>	Error type (GRAPH_MALLOC/GRAPH_READ/GRAPH_ELSE)
-----------------	------------------	---

### 6.12.1.3 `INT_VECTOR int_vec_error ( int err_ )`

Generates error message for integer vector errors and exits.

#### Parameters

<code>in</code>	<code>err</code>	Error type (INT_VEC_MALLOC/INT_VEC_FNOTOPEN/INT_VEC_FNOTGETINT_VEC/INT_VEC_SIZEMISMATCH)
-----------------	------------------	--

### 6.12.1.4 `INT_VECSTACK int_vecstack_error ( int err_ )`

Generates error message for integer vector stack errors and exits.

#### Parameters

<code>in</code>	<code>err</code>	Error type (INT_VECSTACK_MALLOC/INT_VECSTACK_FNOTOPEN/INT_VECSTACK_FNOTGETINT_VEC/INT_VECSTACK_SIZEMISMATCH)
-----------------	------------------	--

### 6.12.1.5 **MATRIX** `mat_error ( int err_ )`

Generates error message for matrix errors and exits.

#### Parameters

<code>in</code>	<code>err</code>	Error type (MAT_MALLOC/MAT_FNOTOPEN/MAT_FNOTGETMAT/MAT_SIZE_MISMATCH/ MAT_INVERSE_ILL_COND/MAT_INVERSE_NOT_SQUARE/MAT_CHOLESKY_FAILED)
-----------------	------------------	--

### 6.12.1.6 **MATSTACK** `matstack_error ( int err_ )`

Generates error message for matrix stack errors and exits.

#### Parameters

<code>in</code>	<code>err</code>	Error type (MATSTACK_MALLOC/MATSTACK_FNOTOPEN/MATSTACK_FNOTGETMAT/MATSTACK_SIZE_MISMATCH/ MATSTACK_INVERSE_ERROR)
-----------------	------------------	---

### 6.12.1.7 `int pq_error ( int err_ )`

Generates error message for priority queue errors and exits.

#### Parameters

<code>in</code>	<code>err</code>	Error type (PQ_MALLOC/PQ_EMPTY)
-----------------	------------------	---------------------------------

### 6.12.1.8 `int queue_error ( int err_ )`

Generates error message for queue errors and exits.

#### Parameters

<code>in</code>	<code>err</code>	Error type (QUEUE_MALLOC/QUEUE_EMPTY)
-----------------	------------------	---------------------------------------

### 6.12.1.9 `int stack_error ( int err_ )`

Generates error message for stack errors and exits.

#### Parameters

<code>in</code>	<code>err</code>	Error type (STACK_MALLOC/STACK_EMPTY)
-----------------	------------------	---------------------------------------

## 6.13 **matfft.c** File Reference

### Functions

- [MATSTACK](#) `mat_fft2` ([MATSTACK](#) `c`, `int` `dir`, [MATSTACK](#) `result`)  
*Computes fast Fourier transform.*

### 6.13.1 Function Documentation

#### 6.13.1.1 MATSTACK mat\_fft2 ( MATSTACK *c*, int *dir*, MATSTACK *result* )

Computes fast Fourier transform.

##### Parameters

in	<i>C</i>	Complex data matrix stack
in	<i>dir</i>	FFT direction (ROWS/COLS)
in	<i>result</i>	Matrix stack to store the result

##### Returns

Transformed matrix stack

## 6.14 matfilter.c File Reference

### Functions

- [MATRIX mat\\_conv2](#) ([MATRIX A](#), [MATRIX mask](#), [MATRIX scratch](#), [MATRIX result](#))  
*Computes 2-D convolution.*

### 6.14.1 Function Documentation

#### 6.14.1.1 MATRIX mat\_conv2 ( MATRIX *A*, MATRIX *mask*, MATRIX *scratch*, MATRIX *result* )

Computes 2-D convolution.

##### Parameters

in	<i>A</i>	Input matrix
in	<i>mask</i>	Input kernel/mask
in	<i>scratch</i>	Scratch matrix for temporary calculations
in	<i>result</i>	Matrix to store the result

##### Returns

Convolved output matrix

## 6.15 matfit.c File Reference

### Functions

- [MATRIX mat\\_linear\\_ls\\_fit](#) ([MATRIX A](#), [MATRIX Y](#), int *deg*, [MATRIX result](#))  
*Polynomial model using least squares.*
- [MATRIX mat\\_least\\_squares](#) ([MATRIX A](#), [MATRIX Y](#), [MATRIX result](#))
- [MATRIX mat\\_w\\_least\\_squares](#) ([MATRIX A](#), [MATRIX Y](#), [MATRIX w](#), [MATRIX result](#))
- [MATRIX mat\\_rob\\_least\\_squares](#) ([MATRIX A](#), [MATRIX Y](#), int *lossfunc*, [MATRIX result](#))
- [MATRIX mat\\_robust\\_fit](#) ([MATRIX A](#), [MATRIX Y](#), int *deg*, int *lossfunc*, [MATRIX result](#))

### 6.15.1 Function Documentation

6.15.1.1 **MATRIX** `mat_least_squares` ( **MATRIX** *A*, **MATRIX** *Y*, **MATRIX** *result* )

6.15.1.2 **MATRIX** `mat_linear_ls_fit` ( **MATRIX** *A*, **MATRIX** *Y*, `int` *deg*, **MATRIX** *result* )

Polynomial model using least squares.

#### Parameters

<code>in</code>	<i>A</i>	Data matrix $N \times 1$
<code>in</code>	<i>Y</i>	Observation matrix $N \times 1$

#### Returns

6.15.1.3 **MATRIX** `mat_rob_least_squares` ( **MATRIX** *A*, **MATRIX** *Y*, `int` *lossfunc*, **MATRIX** *result* )

6.15.1.4 **MATRIX** `mat_robust_fit` ( **MATRIX** *A*, **MATRIX** *Y*, `int` *deg*, `int` *lossfunc*, **MATRIX** *result* )

6.15.1.5 **MATRIX** `mat_w_least_squares` ( **MATRIX** *A*, **MATRIX** *Y*, **MATRIX** *w*, **MATRIX** *result* )

## 6.16 matflip.c File Reference

### Functions

- [MATRIX](#) `mat_fliplr` ([MATRIX](#) *A*, [MATRIX](#) *result*)
- [MATRIX](#) `mat_flipud` ([MATRIX](#) *A*, [MATRIX](#) *result*)

### 6.16.1 Function Documentation

6.16.1.1 **MATRIX** `mat_fliplr` ( **MATRIX** *A*, **MATRIX** *result* )

6.16.1.2 **MATRIX** `mat_flipud` ( **MATRIX** *A*, **MATRIX** *result* )

## 6.17 matfuncs.c File Reference

### Functions

- `mtype` `__mat_addfunc` (`mtype` *x*, `mtype` *y*)  
*Computes addition function.*
- `mtype` `__mat_subfunc` (`mtype` *x*, `mtype` *y*)  
*Computes subtraction function.*
- `mtype` `__mat_mulfunc` (`mtype` *x*, `mtype` *y*)  
*Computes multiplication function.*
- `mtype` `__mat_divfunc` (`mtype` *x*, `mtype` *y*)  
*Computes division function.*
- `mtype` `__mat_sqrfunc` (`mtype` *x*)  
*Computes square function.*
- `mtype` `__mat_sqrtfunc` (`mtype` *x*)  
*Computes square root function.*
- `mtype` `__mat_huber_wt` (`mtype` *x*, `mtype` *k*)  
*Computes Huber weight function.*

- `__mat_bisquare_wt` (mtype x, mtype k)  
*Computes bisquare weight function.*
- `__mat_arcsinh` (mtype x)  
*Computes inverse hyperbolic sine function.*
- `__mat_arccosh` (mtype x)  
*Computes inverse hyperbolic cosine function.*
- `__mat_arctanh` (mtype x)  
*Computes inverse hyperbolic tangent function.*
- `__mat_logplusone` (mtype x)  
*Computes logarithm plus one function.*
- `MATRIX mat_huber_wt` (MATRIX A, mtype k, mtype sigma, MATRIX result)  
*Computes Huber weight function element-wise on a matrix.*
- `MATRIX mat_bisquare_wt` (MATRIX A, mtype k, mtype sigma, MATRIX result)  
*Computes bisquare weight function element-wise on a matrix.*
- `MATRIX mat_gfunc` (MATRIX A, mtype(\*pt2func)(mtype), MATRIX result)  
*Computes a given function element-wise on a matrix.*

## 6.17.1 Function Documentation

### 6.17.1.1 `mtype __mat_addfunc ( mtype x, mtype y )`

Computes addition function.

#### Parameters

in	x	
in	y	

#### Returns

$x + y$

### 6.17.1.2 `mtype __mat_arccosh ( mtype x )`

Computes inverse hyperbolic cosine function.

#### Parameters

in	x	
----	---	--

#### Returns

$\cosh^{-1}(x)$

### 6.17.1.3 `mtype __mat_arcsinh ( mtype x )`

Computes inverse hyperbolic sine function.

#### Parameters

in	x	
----	---	--

**Returns**

$$\sinh^{-1}(x)$$

**6.17.1.4 mtype \_\_mat\_arctanh ( mtype x )**

Computes inverse hyperbolic tangent function.

**Parameters**

in	x	
----	---	--

**Returns**

$$\tanh^{-1}(x)$$

**6.17.1.5 mtype \_\_mat\_bisquare\_wt ( mtype x, mtype k )**

Computes bisquare weight function.

**Parameters**

in	x	
in	k	

**Returns**

$$\begin{cases} \left(1 - \left(\frac{x}{k}\right)^2\right)^2, & \text{for } |x| \leq k, \\ 0, & \text{otherwise.} \end{cases}$$

**6.17.1.6 mtype \_\_mat\_divfunc ( mtype x, mtype y )**

Computes division function.

**Parameters**

in	x	
in	y	

**Returns**

$$\frac{x}{y}$$

**6.17.1.7 mtype \_\_mat\_huber\_wt ( mtype x, mtype k )**

Computes Huber weight function.

**Parameters**

in	x	
in	k	

Returns

$$\begin{cases} 1, & \text{for } |x| \leq k, \\ \frac{k}{|x|}, & \text{otherwise.} \end{cases}$$

## 6.17.1.8 mtype \_\_mat\_logplusone ( mtype x )

Computes logarithm plus one function.

Parameters

in	x	
----	---	--

Returns

$$\log(1+x)$$

## 6.17.1.9 mtype \_\_mat\_mulfunc ( mtype x, mtype y )

Computes multiplication function.

Parameters

in	x	
in	y	

Returns

$$xy$$

## 6.17.1.10 mtype \_\_mat\_sqrtfunc ( mtype x )

Computes square function.

Parameters

in	x	
----	---	--

Returns

$$x^2$$

## 6.17.1.11 mtype \_\_mat\_sqrtfunc ( mtype x )

Computes square root function.

Parameters

in	x	
----	---	--

Returns

$$\sqrt{x}$$

#### 6.17.1.12 `mtypc _mat_subfunc ( mtype x, mtype y )`

Computes subtraction function.

##### Parameters

<code>in</code>	<code>x</code>	
<code>in</code>	<code>y</code>	

##### Returns

$$x - y$$

#### 6.17.1.13 `MATRIX mat_bisquare_wt ( MATRIX A, mtype k, mtype sigma, MATRIX result )`

Computes bisquare weight function element-wise on a matrix.

##### Parameters

<code>in</code>	<code>A</code>	Input matrix
<code>in</code>	<code>k</code>	Bisquare parameter

##### Returns

$$\mathbf{B}, b_{ij} = f_k(a_{ij}) \text{ where } f_k \text{ is the biquare weight function}$$

#### 6.17.1.14 `MATRIX mat_gfunc ( MATRIX A, mtype(*) (mtype) pt2func, MATRIX result )`

Computes a given function element-wise on a matrix.

##### Parameters

<code>in</code>	<code>A</code>	Input matrix
<code>in</code>	<code>f</code>	Given function

##### Returns

$$\mathbf{B}, b_{ij} = f(a_{ij})$$

#### 6.17.1.15 `MATRIX mat_huber_wt ( MATRIX A, mtype k, mtype sigma, MATRIX result )`

Computes Huber weight function element-wise on a matrix.

##### Parameters

<code>in</code>	<code>A</code>	Input matrix
<code>in</code>	<code>k</code>	Huber parameter



## Returns

$\mathbf{B}$ ,  $b_{ij} = f_k(a_{ij})$  where  $f_k$  is the Huber weight function

## 6.18 matgraph.c File Reference

### Functions

- [MAT\\_GRAPH mat\\_graph\\_creat](#) (void)
- void [mat\\_graph\\_adjlist](#) ([MAT\\_GRAPH](#) g, int directed, int weighted, [MAT\\_FILEPOINTER](#) fp)
- [MAT\\_GRAPH mat\\_graph\\_reverse](#) ([MAT\\_GRAPH](#) g, [MAT\\_GRAPH](#) r)
- void [mat\\_graph\\_adjm\\_to\\_adjl](#) ([MAT\\_GRAPH](#) g, [MATRIX](#) a)
- [MAT\\_INT\\_QUEUE mat\\_graph\\_search](#) ([MAT\\_GRAPH](#) g, int connected, int mst)
- void [mat\\_graph\\_visit](#) ([MAT\\_GRAPH](#) g, int k, int connected, int mst, [MAT\\_INT\\_PRIORITYQUEUE](#) pq, [MAT\\_INT\\_QUEUE](#) q)
- void [mat\\_graph\\_dumpf](#) ([MAT\\_GRAPH](#) g, int mst, [MAT\\_FILEPOINTER](#) fp)
- void [mat\\_graph\\_dump](#) ([MAT\\_GRAPH](#) g, int mst)

### 6.18.1 Function Documentation

6.18.1.1 void [mat\\_graph\\_adjlist](#) ( [MAT\\_GRAPH](#) g, int *directed*, int *weighted*, [MAT\\_FILEPOINTER](#) fp )

6.18.1.2 void [mat\\_graph\\_adjm\\_to\\_adjl](#) ( [MAT\\_GRAPH](#) g, [MATRIX](#) a )

6.18.1.3 [MAT\\_GRAPH](#) [mat\\_graph\\_creat](#) ( void )

6.18.1.4 void [mat\\_graph\\_dump](#) ( [MAT\\_GRAPH](#) g, int *mst* )

6.18.1.5 void [mat\\_graph\\_dumpf](#) ( [MAT\\_GRAPH](#) g, int *mst*, [MAT\\_FILEPOINTER](#) fp )

6.18.1.6 [MAT\\_GRAPH](#) [mat\\_graph\\_reverse](#) ( [MAT\\_GRAPH](#) g, [MAT\\_GRAPH](#) r )

6.18.1.7 [MAT\\_INT\\_QUEUE](#) [mat\\_graph\\_search](#) ( [MAT\\_GRAPH](#) g, int *connected*, int *mst* )

6.18.1.8 void [mat\\_graph\\_visit](#) ( [MAT\\_GRAPH](#) g, int *k*, int *connected*, int *mst*, [MAT\\_INT\\_PRIORITYQUEUE](#) pq, [MAT\\_INT\\_QUEUE](#) q )

## 6.19 matinnerprod.c File Reference

### Functions

- mtype [mat\\_innerprod](#) ([MATRIX](#) A, [MATRIX](#) B)
- mtype [mat\\_norm\\_inf](#) ([MATRIX](#) A)
- mtype [mat\\_norm\\_one](#) ([MATRIX](#) A)
- mtype [mat\\_norm\\_p](#) ([MATRIX](#) A, mtype p)

### 6.19.1 Function Documentation

6.19.1.1 mtype [mat\\_innerprod](#) ( [MATRIX](#) A, [MATRIX](#) B )

6.19.1.2 mtype [mat\\_norm\\_inf](#) ( [MATRIX](#) A )

6.19.1.3 mtype [mat\\_norm\\_one](#) ( [MATRIX](#) A )

6.19.1.4 `mtype mat_norm_p ( MATRIX A, mtype p )`

## 6.20 matintegrate.c File Reference

### Functions

- `mtype mat_int_trapezoid (mtype(*func)(mtype), int n, mtype lower, mtype upper)`  
*Computes trapezoid integration.*
- `mtype mat_int_simpson (mtype(*func)(mtype), int n, mtype lower, mtype upper)`  
*Computes Simpson's integration.*
- `mtype mat_int_qadtrat (mtype(*func)(mtype), mtype lower, mtype upper)`  
*Computes Gauss quadrature integration.*

### 6.20.1 Function Documentation

6.20.1.1 `mtype mat_int_qadtrat ( mtype(*) (mtype) func, mtype lower, mtype upper )`

Computes Gauss quadrature integration.

#### Parameters

<code>in</code>	<code>func</code>	Function $f(\cdot)$ to integrate
<code>in</code>	<code>n</code>	Number of subdivisions
<code>in</code>	<code>lower</code>	Lower Limit
<code>in</code>	<code>upper</code>	Upper Limit

#### Returns

$$\int_a^b f(x) dx$$

6.20.1.2 `mtype mat_int_simpson ( mtype(*) (mtype) func, int n, mtype lower, mtype upper )`

Computes Simpson's integration.

#### Parameters

<code>in</code>	<code>func</code>	Function $f(\cdot)$ to integrate
<code>in</code>	<code>n</code>	Number of subdivisions
<code>in</code>	<code>lower</code>	Lower Limit
<code>in</code>	<code>upper</code>	Upper Limit

#### Returns

$$\int_a^b f(x) dx$$

6.20.1.3 `mtype mat_int_trapezoid ( mtype(*) (mtype) func, int n, mtype lower, mtype upper )`

Computes trapezoid integration.

#### Parameters

<code>in</code>	<code>func</code>	Function $f(\cdot)$ to integrate
<code>in</code>	<code>n</code>	Number of subdivisions

<code>in</code>	<i>lower</i>	Lower Limit
<code>in</code>	<i>upper</i>	Upper Limit

Returns

$$\int_a^b f(x) dx$$

## 6.21 matinv.c File Reference

### Functions

- [MATRIX mat\\_inv](#) ([MATRIX A](#), [MATRIX result](#))  
*Computes the inverse of a matrix.*
- [MATRIX mat\\_reg\\_inv](#) ([MATRIX A](#), [mtype r](#), [MATRIX result](#))  
*Computes the regularized inverse of a matrix.*

### 6.21.1 Function Documentation

#### 6.21.1.1 [MATRIX mat\\_inv](#) ( [MATRIX A](#), [MATRIX result](#) )

Computes the inverse of a matrix.

Parameters

<code>in</code>	<i>A</i>	Input matrix
<code>in</code>	<i>result</i>	Matrix to store the result

Returns

$$A^{-1}$$

#### 6.21.1.2 [MATRIX mat\\_reg\\_inv](#) ( [MATRIX A](#), [mtype r](#), [MATRIX result](#) )

Computes the regularized inverse of a matrix.

Parameters

<code>in</code>	<i>A</i>	Input matrix
<code>in</code>	<i>r</i>	Regularizing constant
<code>in</code>	<i>result</i>	Matrix to store the result

Returns

$$(A + rI)^{-1}$$

## 6.22 matkdtree.c File Reference

### Functions

- [MAT\\_KDTREE mat\\_kdtree\\_make\\_tree](#) ([MATRIX A](#), [MAT\\_KDTREE result](#))  
*Creates a k-d tree from a data matrix.*

- `int mat_kdtree_free (MAT_KDTREE t)`  
*Frees a k-d tree.*
- `MATRIX mat_kdtree_nearest (MAT_KDTREE t, MATRIX A, MATRIX result)`  
*Computes nearest neighbors.*
- `MATRIX mat_kdtree_k_nearest (MAT_KDTREE t, MATRIX A, int k, MATRIX result)`  
*Computes k nearest neighbors.*

## 6.22.1 Function Documentation

### 6.22.1.1 `int mat_kdtree_free ( MAT_KDTREE t )`

Frees a k-d tree.

#### Parameters

<code>in</code>	<code>t</code>	Input k-d tree
-----------------	----------------	----------------

#### Returns

Success

### 6.22.1.2 `MATRIX mat_kdtree_k_nearest ( MAT_KDTREE t, MATRIX A, int k, MATRIX result )`

Computes k nearest neighbors.

#### Parameters

<code>in</code>	<code>t</code>	Input k-d tree
<code>in</code>	<code>A</code>	Input data matrix of size $d \times N$
<code>in</code>	<code>k</code>	Number of neighbors
<code>in</code>	<code>result</code>	Matrix to store the result

#### Returns

Output matrix  $B$  with index  $B[0][j]$  and squared distance  $B[1][j]$  for  $j = 1, 2, \dots, N$

### 6.22.1.3 `MAT_KDTREE mat_kdtree_make_tree ( MATRIX A, MAT_KDTREE result )`

Creates a k-d tree from a data matrix.

#### Parameters

<code>in</code>	<code>A</code>	Input data matrix of size $d \times N$
<code>in</code>	<code>result</code>	K-d tree to store the result

#### Returns

Output k-d tree

### 6.22.1.4 `MATRIX mat_kdtree_nearest ( MAT_KDTREE t, MATRIX A, MATRIX result )`

Computes nearest neighbors.

## Parameters

in	$t$	Input k-d tree
in	$A$	Input data matrix of size $d \times N$
in	$result$	Matrix to store the result

## Returns

Output matrix  $B$  with index  $B[0][j]$  and squared distance  $B[1][j]$  for  $j = 1, 2, \dots, N$

## 6.23 matmaxmin.c File Reference

## Functions

- [MATVEC\\_DPOINTER mat\\_max](#) ([MATRIX](#)  $A$ , int  $dim$ )
- [MATVEC\\_DPOINTER mat\\_min](#) ([MATRIX](#)  $A$ , int  $dim$ )

## 6.23.1 Function Documentation

6.23.1.1 [MATVEC\\_DPOINTER mat\\_max](#) ( [MATRIX](#)  $A$ , int  $dim$  )

6.23.1.2 [MATVEC\\_DPOINTER mat\\_min](#) ( [MATRIX](#)  $A$ , int  $dim$  )

## 6.24 matmds.c File Reference

## Functions

- [MATRIX mat\\_mds](#) ([MATRIX](#)  $d$ , int  $dims$ , int  $type$ , [MATRIX](#)  $result$ )
- [MATRIX \\_\\_mat\\_mds\\_metric](#) ([MATRIX](#)  $d$ , int  $dims$ , [MATRIX](#)  $result$ )
- [MATRIX \\_\\_mat\\_mds\\_nonmetric](#) ([MATRIX](#)  $d$ , int  $dims$ , [MATRIX](#)  $result$ )

## 6.24.1 Function Documentation

6.24.1.1 [MATRIX \\_\\_mat\\_mds\\_metric](#) ( [MATRIX](#)  $d$ , int  $dims$ , [MATRIX](#)  $result$  )

6.24.1.2 [MATRIX \\_\\_mat\\_mds\\_nonmetric](#) ( [MATRIX](#)  $d$ , int  $dims$ , [MATRIX](#)  $result$  )

6.24.1.3 [MATRIX mat\\_mds](#) ( [MATRIX](#)  $d$ , int  $dims$ , int  $type$ , [MATRIX](#)  $result$  )

## 6.25 matmean.c File Reference

## Functions

- [mtype mat\\_mean](#) ([MATRIX](#)  $A$ )
- [MATRIX mat\\_mean\\_row](#) ([MATRIX](#)  $A$ , [MATRIX](#)  $result$ )
- [MATRIX mat\\_mean\\_col](#) ([MATRIX](#)  $A$ , [MATRIX](#)  $result$ )

## 6.25.1 Function Documentation

6.25.1.1 [mtype mat\\_mean](#) ( [MATRIX](#)  $A$  )

6.25.1.2 **MATRIX** `mat_mean_col ( MATRIX A, MATRIX result )`

6.25.1.3 **MATRIX** `mat_mean_row ( MATRIX A, MATRIX result )`

## 6.26 matmisc.c File Reference

### Functions

- `int mats_isnan (mtype x)`  
*Checks if scalar is NaN.*
- `int mats_isinf (mtype x)`  
*Checks if scalar is infinite.*
- `void mat_nextline (void)`  
*Prints nextline to stdout.*
- `void mat_fnextline (MAT_FILEPOINTER fp)`  
*Prints nextline to file.*
- **MATRIX** `mat_bsxfun (MATRIX A, MATRIX B, mtype(*func)(mtype, mtype), MATRIX result)`  
*Computes element-wise binary function for two matrices.*
- **INT\_VECTOR** `int_vec_permute_vect (int n, int k, INT_VECTOR result)`  
*Computes a randomly permutation of first k positive integers.*
- **INT\_VECTOR** `mat_get_sub_vector (INT_VECTOR a, INT_VECTOR indices)`  
*Extracts sub-vector from an integer vector.*
- **MATRIX** `mat_get_sub_matrix_from_rows (MATRIX A, INT_VECTOR indices, MATRIX result)`  
*Extracts sub-matrix from rows of a matrix.*
- **MATRIX** `mat_get_sub_matrix_from_cols (MATRIX A, INT_VECTOR indices, MATRIX result)`  
*Extracts sub-matrix from columns of a matrix.*
- **MATRIX** `mat_calc_dist_sq (MATRIX A, MATRIX d, MATRIX result)`  
*Computes the Euclidean distances of points from a given point.*
- **INT\_VECTOR** `mat_find_within_dist (MATRIX A, MATRIX d, mtype range)`  
*Finds points within a neighborhood.*
- **MATRIX** `mat_pick_row (MATRIX A, int r, MATRIX result)`  
*Picks a row from a matrix.*
- **MATRIX** `mat_pick_col (MATRIX A, int c, MATRIX result)`  
*Picks a column from a matrix.*
- `void __mat_cart2pol (mtype x, mtype y, mtype *rho, mtype *th)`
- `void __mat_pol2cart (mtype rho, mtype th, mtype *x, mtype *y)`
- **MATRIX** `mat_cart2pol (MATRIX A, int dim, MATRIX result)`  
*Converts Cartesian co-ordinates to polar co-ordinates.*
- **MATRIX** `mat_pol2cart (MATRIX A, int dim, MATRIX result)`  
*Converts polar co-ordinates to Cartesian co-ordinates.*
- **INT\_VECTOR** `int_vec_unique (INT_VECTOR a)`  
*Extract only the unique integers from an integer vector.*

### 6.26.1 Function Documentation

6.26.1.1 `void __mat_cart2pol ( mtype x, mtype y, mtype * rho, mtype * th )`

6.26.1.2 `void __mat_pol2cart ( mtype rho, mtype th, mtype * x, mtype * y )`

6.26.1.3 **INT\_VECTOR** `int_vec_permute_vect ( int n, int k, INT_VECTOR result )`

Computes a randomly permutation of first k positive integers.

## Parameters

in	<i>n</i>	Number of random permutations to make
in	<i>k</i>	Integer upto which it will consider
in	<i>result</i>	Vector to store the result

## Returns

Permuted vector

6.26.1.4 INT\_VECTOR int\_vec\_unique ( INT\_VECTOR *a* )

Extract only the unique integers from an integer vector.

## Parameters

in	<i>a</i>	Input vector
----	----------	--------------

## Returns

Unique vector

6.26.1.5 MATRIX mat\_bsxfun ( MATRIX *A*, MATRIX *B*, mtype(\*) (mtype, mtype) *func*, MATRIX *result* )

Computes element-wise binary function for two matrices.

## Parameters

in	<i>A</i>	First matrix
in	<i>B</i>	Second matrix
in	<i>func</i>	Pointer to the function
in	<i>result</i>	Matrix to store the result

## Returns

Output matrix

6.26.1.6 MATRIX mat\_calc\_dist\_sq ( MATRIX *A*, MATRIX *d*, MATRIX *result* )

Computes the Euclidean distances of points from a given point.

## Parameters

in	<i>A</i>	Points matrix (d x N)
in	<i>d</i>	Matrix point from which the distance to be computed (d x 1)
in	<i>result</i>	Matrix to store the result

## Returns

Euclidean distance matrix

6.26.1.7 MATRIX mat\_cart2pol ( MATRIX *A*, int *dim*, MATRIX *result* )

Converts Cartesian co-ordinates to polar co-ordinates.

## Parameters

<i>in</i>	<i>A</i>	Input matrix
<i>in</i>	<i>dim</i>	Data order ROWS/COLS

## Returns

Polar co-ordinate matrix

#### 6.26.1.8 INT\_VECTOR mat\_find\_within\_dist ( MATRIX *A*, MATRIX *d*, mtype *range* )

Finds points within a neighborhood.

## Parameters

<i>in</i>	<i>A</i>	Points matrix (d x N)
<i>in</i>	<i>d</i>	Matrix point from which the distance to be computed (d x 1)
<i>in</i>	<i>range</i>	Radius to search within

## Returns

Indices Vector

#### 6.26.1.9 void mat\_fnextline ( MAT\_FILEPOINTER *fp* )

Prints nextline to file.

## Parameters

<i>in</i>	<i>fp</i>	Pointer to opened file
-----------	-----------	------------------------

#### 6.26.1.10 MATRIX mat\_get\_sub\_matrix\_from\_cols ( MATRIX *A*, INT\_VECTOR *indices*, MATRIX *result* )

Extracts sub-matrix from columns of a matrix.

## Parameters

<i>in</i>	<i>A</i>	Input matrix
<i>in</i>	<i>indices</i>	Columns to extract
<i>in</i>	<i>result</i>	Matrix to store the result

## Returns

Extracted matrix

#### 6.26.1.11 MATRIX mat\_get\_sub\_matrix\_from\_rows ( MATRIX *A*, INT\_VECTOR *indices*, MATRIX *result* )

Extracts sub-matrix from rows of a matrix.

## Parameters

<i>in</i>	<i>A</i>	Input matrix
<i>in</i>	<i>indices</i>	Rows to extract
<i>in</i>	<i>result</i>	Matrix to store the result



## Returns

Extracted matrix

6.26.1.12 `INT_VECTOR mat_get_sub_vector ( INT_VECTOR a, INT_VECTOR indices )`

Extracts sub-vector from an integer vector.

## Parameters

<code>in</code>	<code>a</code>	Input vector
<code>in</code>	<code>indices</code>	Indices to extracted

## Returns

Extracted vector

6.26.1.13 `void mat_nextline ( void )`

Prints nextline to stdout.

6.26.1.14 `MATRIX mat_pick_col ( MATRIX A, int c, MATRIX result )`

Picks a column from a matrix.

## Parameters

<code>in</code>	<code>A</code>	Input matrix
<code>in</code>	<code>r</code>	Column index
<code>in</code>	<code>result</code>	Matrix to store the result

## Returns

Column matrix

6.26.1.15 `MATRIX mat_pick_row ( MATRIX A, int r, MATRIX result )`

Picks a row from a matrix.

## Parameters

<code>in</code>	<code>A</code>	Input matrix
<code>in</code>	<code>r</code>	Row index
<code>in</code>	<code>result</code>	Matrix to store the result

## Returns

Row matrix

6.26.1.16 `MATRIX mat_pol2cart ( MATRIX A, int dim, MATRIX result )`

Converts polar co-ordinates to Cartesian co-ordinates.

**Parameters**

<code>in</code>	<code>A</code>	Input matrix
<code>in</code>	<code>dim</code>	Data order ROWS/COLS

**Returns**

Cartesian co-ordinate matrix

**6.26.1.17 int mats\_isinf ( mtype x )**

Checks if scalar is infinite.

**Parameters**

<code>in</code>	<code>x</code>	Input scalar
-----------------	----------------	--------------

**Returns**

Zero/non-zero

**6.26.1.18 int mats\_isnan ( mtype x )**

Checks if scalar is NaN.

**Parameters**

<code>in</code>	<code>x</code>	Input scalar
-----------------	----------------	--------------

**Returns**

Zero/non-zero

**6.27 matmul.c File Reference****Functions**

- [MATRIX mat\\_mul](#) (MATRIX A, MATRIX B, MATRIX result)
- [MATRIX mat\\_mul\\_fast](#) (MATRIX A, MATRIX B, MATRIX result)
- [MATRIX mat\\_muls](#) (MATRIX A, mtype s, MATRIX B)
- [MATRIX mat\\_mul\\_dot](#) (MATRIX A, MATRIX B, MATRIX C)
- mtype [mat\\_diagmul](#) (MATRIX A)
- [INT\\_VECTOR int\\_vec\\_mul](#) (INT\_VECTOR A, INT\_VECTOR B, INT\_VECTOR result)
- [INT\\_VECTOR int\\_vec\\_muls](#) (INT\_VECTOR A, int x, INT\_VECTOR result)

**6.27.1 Function Documentation****6.27.1.1 INT\_VECTOR int\_vec\_mul ( INT\_VECTOR A, INT\_VECTOR B, INT\_VECTOR result )****6.27.1.2 INT\_VECTOR int\_vec\_muls ( INT\_VECTOR A, int x, INT\_VECTOR result )****6.27.1.3 mtype mat\_diagmul ( MATRIX A )**

6.27.1.4 **MATRIX** `mat_mul` ( **MATRIX** *A*, **MATRIX** *B*, **MATRIX** *result* )

6.27.1.5 **MATRIX** `mat_mul_dot` ( **MATRIX** *A*, **MATRIX** *B*, **MATRIX** *C* )

6.27.1.6 **MATRIX** `mat_mul_fast` ( **MATRIX** *A*, **MATRIX** *B*, **MATRIX** *result* )

6.27.1.7 **MATRIX** `mat_muls` ( **MATRIX** *A*, *mtype s*, **MATRIX** *B* )

## 6.28 matpca.c File Reference

### Functions

- **MATSTACK** `mat_pca` (**MATRIX** *data*, *int pca\_type*)
- **MATSTACK** `mat_eig_sym` (**MATRIX** *symmat*, **MATSTACK** *result*)
- **MATSTACK** `mat_corcol` (**MATRIX** *data*)
- **MATSTACK** `mat_covcol` (**MATRIX** *data*)
- **MATRIX** `mat_scpcol` (**MATRIX** *data*)
- void `mat_tred2` (**MATRIX** *a*, **MATRIX** *d*, **MATRIX** *e*)
- void `mat_tqli` (**MATRIX** *d*, **MATRIX** *e*, **MATRIX** *z*)

### 6.28.1 Function Documentation

6.28.1.1 **MATSTACK** `mat_corcol` ( **MATRIX** *data* )

6.28.1.2 **MATSTACK** `mat_covcol` ( **MATRIX** *data* )

6.28.1.3 **MATSTACK** `mat_eig_sym` ( **MATRIX** *symmat*, **MATSTACK** *result* )

6.28.1.4 **MATSTACK** `mat_pca` ( **MATRIX** *data*, *int pca\_type* )

6.28.1.5 **MATRIX** `mat_scpcol` ( **MATRIX** *data* )

6.28.1.6 void `mat_tqli` ( **MATRIX** *d*, **MATRIX** *e*, **MATRIX** *z* )

6.28.1.7 void `mat_tred2` ( **MATRIX** *a*, **MATRIX** *d*, **MATRIX** *e* )

## 6.29 matpinv.c File Reference

### Functions

- **MATRIX** `mat_pinv` (**MATRIX** *A*, **MATRIX** *result*)  
*Computes pseudo-inverse of a matrix.*
- **MATRIX** `mat_wpinv` (**MATRIX** *A*, **MATRIX** *w*, **MATRIX** *result*)  
*Computes weighted pseudo-inverse of a matrix.*

### 6.29.1 Function Documentation

6.29.1.1 **MATRIX** `mat_pinv` ( **MATRIX** *A*, **MATRIX** *result* )

Computes pseudo-inverse of a matrix.

## Parameters

in	A	Input matrix
in	result	Matrix to store the result

## Returns

$$(A^T A)^{-1} A^T$$

6.29.1.2 **MATRIX** mat\_wpinv ( **MATRIX** A, **MATRIX** w, **MATRIX** result )

Computes weighted pseudo-inverse of a matrix.

## Parameters

in	A	Input matrix
in	w	Weight matrix
in	result	Matrix to store the result

## Returns

$$(A^T W A)^{-1} A^T W$$

## 6.30 matpoly.c File Reference

## Functions

- **MATRIX** mat\_poly\_eval (**MATRIX** A, mtype x, int dir, **MATRIX** result)  
*Evaluates polynomial at a point.*
- **MATRIX** mat\_poly\_diff (**MATRIX** A, int dir, **MATRIX** result)  
*Computes derivative polynomial of a polynomial.*
- **MATRIX** mat\_poly\_diff\_eval (**MATRIX** A, mtype x, int dir, **MATRIX** result)  
*Evaluates derivative polynomial at a point.*
- **MATRIX** mat\_poly\_add (**MATRIX** A, **MATRIX** B, **MATRIX** result)  
*Adds two polynomials.*
- **MATRIX** mat\_poly\_mul (**MATRIX** A, **MATRIX** B, **MATRIX** result)  
*Multiplies two polynomials.*
- **MATSTACK** mat\_poly\_div (**MATRIX** A, **MATRIX** B, **MATSTACK** result)  
*Divides two polynomials.*
- **MATRIX** mat\_poly\_scale (**MATRIX** A, mtype s, **MATRIX** result)  
*Multiplies a polynomial with a scalar.*
- **MATRIX** mat\_poly\_shift (**MATRIX** A, int s, **MATRIX** result)  
*Shifts a polynomial.*
- void mat\_cheby\_init ()  
*Initializes the Chebyshev polynomial series.*
- void mat\_legendre\_init ()  
*Initializes the Legendre polynomial series.*
- void mat\_binom\_init ()  
*Initializes the binomial series.*
- **MATRIX** mat\_cheby (int n)  
*Computes the  $n^{th}$  Chebyshev polynomial.*
- **MATRIX** mat\_legendre (int n)

- Computes the  $n^{th}$  Legendre polynomial.
- mtype [mat\\_binom](#) (int  $n$ , int  $k$ )  
Computes a binomial co-efficient.
- **MATRIX** [mat\\_cheby\\_coeffs\\_to\\_poly](#) (**MATRIX** coeffs, **MATRIX** result)  
Converts Chebyshev co-efficients to a single polynomial.
- **MATRIX** [mat\\_cheby\\_approx](#) (mtype(\*))(mtype)  $f$ , mtype  $a$ , mtype  $b$ , int  $n$ , **MATRIX** result)  
Approximates a function using Chebyshev polynomials.

## Variables

- **MATSTACK** [mat\\_cheby\\_series\\_table](#)
- **MATSTACK** [mat\\_legendre\\_series\\_table](#)
- **MATSTACK** [mat\\_binom\\_series\\_table](#)

## 6.30.1 Function Documentation

### 6.30.1.1 mtype mat\_binom ( int $n$ , int $k$ )

Computes a binomial co-efficient.

#### Parameters

in	$n$	1 <sup>st</sup> argument
in	$k$	2 <sup>nd</sup> argument

#### Returns

$$\binom{n}{k}$$

### 6.30.1.2 void mat\_binom\_init ( )

Initializes the binomial series.

### 6.30.1.3 **MATRIX** mat\_cheby ( int $n$ )

Computes the  $n^{th}$  Chebyshev polynomial.

#### Parameters

in	$n$	Polynomial series index
----	-----	-------------------------

#### Returns

Output polynomial matrix

### 6.30.1.4 **MATRIX** mat\_cheby\_approx ( mtype(\*))(mtype) $f$ , mtype $a$ , mtype $b$ , int $n$ , **MATRIX** result )

Approximates a function using Chebyshev polynomials.

**Parameters**

in	$f$	Function to approximate
in	$a$	Lower limit of domain of the function
in	$b$	Upper limit of domain of the function
in	$n$	Degree of the approximate polynomial
in	$result$	Matrix to store the result

**Returns**

Approximate polynomial matrix

**6.30.1.5 MATRIX mat\_cheby\_coefs\_to\_poly ( MATRIX coeffs, MATRIX result )**

Converts Chebyshev co-efficients to a single polynomial.

**Parameters**

in	$coeffs$	Chebyshev polynomial co-efficient matrix
in	$result$	Matrix to store the result

**Returns**

Polynomial matrix

**6.30.1.6 void mat\_cheby\_init ( )**

Initializes the Chebyshev polynomial series.

**6.30.1.7 MATRIX mat\_legendre ( int  $n$  )**

Computes the  $n^{th}$  Legendre polynomial.

**Parameters**

in	$n$	Polynomial series index
----	-----	-------------------------

**Returns**

Output polynomial matrix

**6.30.1.8 void mat\_legendre\_init ( )**

Initializes the Legendre polynomial series.

**6.30.1.9 MATRIX mat\_poly\_add ( MATRIX  $A$ , MATRIX  $B$ , MATRIX result )**

Adds two polynomials.

**Parameters**

in	$A$	First input polynomial matrix
in	$B$	Second input polynomial matrix
in	$result$	Matrix to store the result

## Returns

Output matrix

6.30.1.10 **MATRIX** `mat_poly_diff ( MATRIX A, int dir, MATRIX result )`

Computes derivative polynomial of a polynomial.

## Parameters

<i>in</i>	<i>A</i>	Input polynomial matrix
<i>in</i>	<i>dir</i>	Direction (ROWS/COLS)
<i>in</i>	<i>result</i>	Matrix to store the result

## Returns

Output matrix

6.30.1.11 **MATRIX** `mat_poly_diff_eval ( MATRIX A, mtype x, int dir, MATRIX result )`

Evaluates derivative polynomial at a point.

## Parameters

<i>in</i>	<i>A</i>	Input polynomial matrix
<i>in</i>	<i>x</i>	Value at which to evaluate the derivative
<i>in</i>	<i>dir</i>	Direction (ROWS/COLS)
<i>in</i>	<i>result</i>	Matrix to store the result

## Returns

Output matrix

6.30.1.12 **MATSTACK** `mat_poly_div ( MATRIX A, MATRIX B, MATSTACK result )`

Divides two polynomials.

## Parameters

<i>in</i>	<i>A</i>	First input polynomial matrix
<i>in</i>	<i>B</i>	Second input polynomial matrix
<i>in</i>	<i>result</i>	Matrix to store the result

## Returns

Output matrix

6.30.1.13 **MATRIX** `mat_poly_eval ( MATRIX A, mtype x, int dir, MATRIX result )`

Evaluates polynomial at a point.

**Parameters**

<i>in</i>	<i>A</i>	Input polynomial matrix
<i>in</i>	<i>x</i>	Value at which to evaluate
<i>in</i>	<i>dir</i>	Direction (ROWS/COLS)
<i>in</i>	<i>result</i>	Matrix to store the result

**Returns**

Output matrix

**6.30.1.14 MATRIX mat\_poly\_mul ( MATRIX A, MATRIX B, MATRIX result )**

Multiplies two polynomials.

**Parameters**

<i>in</i>	<i>a</i>	First input polynomial matrix
<i>in</i>	<i>b</i>	Second input polynomial matrix
<i>in</i>	<i>result</i>	Matrix to store the result

**Returns**

Output matrix

**6.30.1.15 MATRIX mat\_poly\_scale ( MATRIX A, mtype s, MATRIX result )**

Multiplies a polynomial with a scalar.

**Parameters**

<i>in</i>	<i>A</i>	Input polynomial matrix
<i>in</i>	<i>s</i>	Scalar
<i>in</i>	<i>result</i>	Matrix to store the result

**Returns**

Output matrix

**6.30.1.16 MATRIX mat\_poly\_shift ( MATRIX A, int s, MATRIX result )**

Shifts a polynomial.

**Parameters**

<i>in</i>	<i>A</i>	Input polynomial matrix
<i>in</i>	<i>s</i>	Scalar shift
<i>in</i>	<i>result</i>	Matrix to store the result

**Returns**

Output matrix



## 6.30.2 Variable Documentation

6.30.2.1 **MATSTACK** `mat_binom_series_table`

6.30.2.2 **MATSTACK** `mat_cheby_series_table`

6.30.2.3 **MATSTACK** `mat_legendre_series_table`

## 6.31 matprec.c File Reference

### Functions

- [MAT\\_BAYES\\_MODEL](#) `mat_bayes_classifier_train` ([MATRIX](#) *data*, [INT\\_VECTOR](#) *labels*)
- [INT\\_VECTOR](#) `mat_bayes_classifier_test` ([MATRIX](#) *data*, [MAT\\_BAYES\\_MODEL](#) *b\_model*)
- [MAT\\_PERCEPTRON](#) `mat_perceptron_train` ([MATRIX](#) *data*, [INT\\_VECTOR](#) *labels*, `int num_of_iterations`)
- [MAT\\_PERCEPTRON](#) `mat_perceptron_train_` ([MATRIX](#) *data1*, [MATRIX](#) *data2*, [MAT\\_PERCEPTRON](#) *p\_model*, `int class_num`)
- [INT\\_VECTOR](#) `mat_perceptron_test` ([MATRIX](#) *data*, [MAT\\_PERCEPTRON](#) *p\_model*)
- [MATVEC\\_DPOINTER](#) `mat_kmeans` ([MATRIX](#) *data*, `int k`, `int iters`, [MATVEC\\_DPOINTER](#) *result*)

### 6.31.1 Function Documentation

6.31.1.1 [INT\\_VECTOR](#) `mat_bayes_classifier_test` ( [MATRIX](#) *data*, [MAT\\_BAYES\\_MODEL](#) *b\_model* )

6.31.1.2 [MAT\\_BAYES\\_MODEL](#) `mat_bayes_classifier_train` ( [MATRIX](#) *data*, [INT\\_VECTOR](#) *labels* )

6.31.1.3 [MATVEC\\_DPOINTER](#) `mat_kmeans` ( [MATRIX](#) *data*, `int k`, `int iters`, [MATVEC\\_DPOINTER](#) *result* )

6.31.1.4 [INT\\_VECTOR](#) `mat_perceptron_test` ( [MATRIX](#) *data*, [MAT\\_PERCEPTRON](#) *p\_model* )

6.31.1.5 [MAT\\_PERCEPTRON](#) `mat_perceptron_train` ( [MATRIX](#) *data*, [INT\\_VECTOR](#) *labels*, `int num_of_iterations` )

6.31.1.6 [MAT\\_PERCEPTRON](#) `mat_perceptron_train_` ( [MATRIX](#) *data1*, [MATRIX](#) *data2*, [MAT\\_PERCEPTRON](#) *p\_model*, `int class_num` )

## 6.32 matpursuit.c File Reference

### Functions

- [MATSTACK](#) `mat_omp` ([MATRIX](#) *A*, [MATRIX](#) *b*, `int k`, `mtyp tol`, [MATSTACK](#) *result*)

### 6.32.1 Function Documentation

6.32.1.1 [MATSTACK](#) `mat_omp` ( [MATRIX](#) *A*, [MATRIX](#) *b*, `int k`, `mtyp tol`, [MATSTACK](#) *result* )

## 6.33 matrand.c File Reference

### Functions

- [MATRIX](#) `mat_rand` (`int n`, `int m`, [MATRIX](#) *result*)
- [MATRIX](#) `mat_randn` (`int n`, `int m`, [MATRIX](#) *result*)
- [MATRIX](#) `mat_randexp` (`int n`, `int m`, `mtyp mu`, [MATRIX](#) *result*)

- [MATRIX mat\\_randfun](#) (int *n*, int *m*, mtype(\*fun)(mtype), mtype *xmin*, mtype *xmax*, [MATRIX](#) *result*)
- void [mat\\_set\\_seed](#) (int *seed*)
- mtype [\\_\\_mat\\_randfun](#) (mtype(\*fun)(mtype), mtype *xmin*, mtype *xmax*)
- mtype [\\_\\_mat\\_rand](#) (void)
- mtype [\\_\\_mat\\_randn](#) (void)
- mtype [\\_\\_mat\\_randexp](#) (mtype *mu*)
- [MATRIX mat\\_randperm](#) (int *m*, int *n*, [MATRIX](#) *result*)
- [MATRIX mat\\_randperm\\_n](#) (int *n*, [MATRIX](#) *result*)
- [INT\\_VECTOR int\\_vec\\_randperm](#) (int *n*, [INT\\_VECTOR](#) *result*)

## Variables

- unsigned int [MAT\\_SEED](#) = 0
- int [MAT\\_SET\\_SEED](#) = 0

## 6.33.1 Function Documentation

6.33.1.1 mtype [\\_\\_mat\\_rand](#) ( void )

6.33.1.2 mtype [\\_\\_mat\\_randexp](#) ( mtype *mu* )

6.33.1.3 mtype [\\_\\_mat\\_randfun](#) ( mtype(\*)(*mtype*) *fun*, mtype *xmin*, mtype *xmax* )

6.33.1.4 mtype [\\_\\_mat\\_randn](#) ( void )

6.33.1.5 [INT\\_VECTOR int\\_vec\\_randperm](#) ( int *n*, [INT\\_VECTOR](#) *result* )

6.33.1.6 [MATRIX mat\\_rand](#) ( int *n*, int *m*, [MATRIX](#) *result* )

6.33.1.7 [MATRIX mat\\_randexp](#) ( int *n*, int *m*, mtype *mu*, [MATRIX](#) *result* )

6.33.1.8 [MATRIX mat\\_randfun](#) ( int *n*, int *m*, mtype(\*)(*mtype*) *fun*, mtype *xmin*, mtype *xmax*, [MATRIX](#) *result* )

6.33.1.9 [MATRIX mat\\_randn](#) ( int *n*, int *m*, [MATRIX](#) *result* )

6.33.1.10 [MATRIX mat\\_randperm](#) ( int *m*, int *n*, [MATRIX](#) *result* )

6.33.1.11 [MATRIX mat\\_randperm\\_n](#) ( int *n*, [MATRIX](#) *result* )

6.33.1.12 void [mat\\_set\\_seed](#) ( int *seed* )

## 6.33.2 Variable Documentation

6.33.2.1 unsigned int [MAT\\_SEED](#) = 0

6.33.2.2 int [MAT\\_SET\\_SEED](#) = 0

## 6.34 matrix.c File Reference

## 6.35 matrix.h File Reference

## Data Structures

- struct [mat\\_int\\_stack](#)

- Integer Stack Structure.*
- struct [mat\\_mtype\\_stack](#)
- Mtype Stack Structure.*
- struct [mat\\_qintnode](#)
- Integer Queue Node Structure.*
- struct [mat\\_int\\_queue](#)
- Integer Queue Structure.*
- struct [mat\\_qmtyphenode](#)
- Mtype Queue Node Structure.*
- struct [mat\\_mtype\\_queue](#)
- Mtype Queue Structure.*
- struct [mat\\_intpqnode](#)
- Integer Priority Queue Node Structure.*
- struct [mat\\_int\\_priorityqueue](#)
- Integer Priority Queue Structure.*
- struct [mat\\_mtypepqnode](#)
- Mtype Priority Queue Node Structure.*
- struct [mat\\_mtype\\_priorityqueue](#)
- Mtype Priority Queue Structure.*
- struct [mat\\_tree\\_node](#)
- Search Tree Node Structure.*
- struct [mat\\_bayes\\_model](#)
- Bayes Classifier Model Structure.*
- struct [mat\\_perceptron](#)
- Perceptron Classifier Model Structure.*
- struct [mat\\_gnode](#)
- Graph Node Structure.*
- struct [mat\\_graph](#)
- Graph Structure.*
- struct [mat\\_kdnode](#)
- struct [mat\\_kdtree](#)

## Typedefs

- typedef struct [mat\\_int\\_stack](#) [mat\\_int\\_stack](#)
- Integer Stack Structure.*
- typedef [mat\\_int\\_stack](#) \* [MAT\\_INT\\_STACK](#)
- typedef struct [mat\\_mtype\\_stack](#) [mat\\_mtype\\_stack](#)
- Mtype Stack Structure.*
- typedef [mat\\_mtype\\_stack](#) \* [MAT\\_MTYPE\\_STACK](#)
- typedef struct [mat\\_qintnode](#) [mat\\_qintnode](#)
- Integer Queue Node Structure.*
- typedef [mat\\_qintnode](#) \* [MAT\\_QINTNODE](#)
- typedef struct [mat\\_int\\_queue](#) [mat\\_int\\_queue](#)
- Integer Queue Structure.*
- typedef [mat\\_int\\_queue](#) \* [MAT\\_INT\\_QUEUE](#)
- typedef struct [mat\\_qmtyphenode](#) [mat\\_qmtyphenode](#)
- Mtype Queue Node Structure.*
- typedef [mat\\_qmtyphenode](#) \* [MAT\\_QMTYPENODE](#)
- typedef struct [mat\\_mtype\\_queue](#) [mat\\_mtype\\_queue](#)
- Mtype Queue Structure.*

- typedef `mat_mtype_queue` \* `MAT_MTYPE_QUEUE`
- typedef struct `mat_intpqnode` `mat_intpqnode`  
*Integer Priority Queue Node Structure.*
- typedef `mat_intpqnode` \* `MAT_INTPQNODE`
- typedef struct `mat_int_priorityqueue` `mat_int_priorityqueue`  
*Integer Priority Queue Structure.*
- typedef `mat_int_priorityqueue` \* `MAT_INT_PRIORITYQUEUE`
- typedef struct `mat_mtypepqnode` `mat_mtypepqnode`  
*Mtype Priority Queue Node Structure.*
- typedef `mat_mtypepqnode` \* `MAT_MTYPEPQNODE`
- typedef struct `mat_mtype_priorityqueue` `mat_mtype_priorityqueue`  
*Mtype Priority Queue Structure.*
- typedef `mat_mtype_priorityqueue` \* `MAT_MTYPE_PRIORITYQUEUE`
- typedef struct `mat_tree_node` `mat_tree_node`  
*Search Tree Node Structure.*
- typedef `mat_tree_node` \* `MAT_TREE_NODE`
- typedef `mat_tree_node` \* `MAT_TREE`
- typedef int \* `INT_VECTOR`
- typedef mtype \*\* `MATRIX`
- typedef `INT_VECTOR` \* `INT_VECSTACK`
- typedef `MATRIX` \* `MATSTACK`
- typedef void \*\* `MATVEC_DPOINTER`
- typedef struct `mat_bayes_model` `mat_bayes_model`  
*Bayes Classifier Model Structure.*
- typedef `mat_bayes_model` \* `MAT_BAYES_MODEL`
- typedef struct `mat_perceptron` `mat_perceptron`  
*Perceptron Classifier Model Structure.*
- typedef `mat_perceptron` \* `MAT_PERCEPTRON`
- typedef struct `mat_gnode` `mat_gnode`  
*Graph Node Structure.*
- typedef `mat_gnode` \* `MAT_GNODE`
- typedef struct `mat_graph` `mat_graph`  
*Graph Structure.*
- typedef `mat_graph` \* `MAT_GRAPH`
- typedef struct `mat_kdnode` `mat_kdnode`
- typedef `mat_kdnode` \* `MAT_KDNODE`
- typedef struct `mat_kdtree` `mat_kdtree`
- typedef `mat_kdtree` \* `MAT_KDTREE`

## Functions

- int `mats_isnan` (mtype x)  
*Checks if scalar is NaN.*
- int `mats_isinf` (mtype x)  
*Checks if scalar is infinite.*
- `INT_VECTOR` `__int_vec_creat` (int len)
- `INT_VECTOR` `int_vec_creat` (int len, int type)  
*Creates an integer vector.*
- `INT_VECTOR` `int_vec_fill` (`INT_VECTOR` A, int val)  
*Fills an integer vector with a value.*

- [INT\\_VECTOR](#) [int\\_vec\\_fill\\_type](#) ([INT\\_VECTOR](#) A, int type)  
*Fills an integer vector to a type.*
- int [int\\_vec\\_free](#) ([INT\\_VECTOR](#) A)  
*Frees an integer vector.*
- [INT\\_VECSTACK](#) [\\_\\_int\\_vecstack\\_creat](#) (int len)
- [INT\\_VECSTACK](#) [int\\_vecstack\\_creat](#) (int len)  
*Creates an integer vector stack.*
- int [int\\_vecstack\\_free](#) ([INT\\_VECSTACK](#) A)  
*Frees an integer vector stack.*
- [MATRIX](#) [\\_\\_mat\\_creat](#) (int r, int c)
- [MATRIX](#) [mat\\_creat](#) (int r, int c, int type)  
*Creates a matrix.*
- [MATRIX](#) [mat\\_creat\\_diag](#) ([MATRIX](#) diag\_vals, [MATRIX](#) result)  
*Creates a diagonal matrix from a 1-d matrix.*
- [MATRIX](#) [mat\\_fill](#) ([MATRIX](#) A, mtype val)  
*Fills a matrix with a value.*
- [MATRIX](#) [mat\\_fill\\_type](#) ([MATRIX](#) A, int type)  
*Fills a matrix to a type.*
- int [mat\\_free](#) ([MATRIX](#) A)  
*Frees a matrix.*
- [MATSTACK](#) [matstack\\_creat](#) (int len)  
*Creates a matrix stack.*
- [MATSTACK](#) [\\_\\_matstack\\_creat](#) (int len)
- int [matstack\\_free](#) ([MATSTACK](#) A)  
*Frees a matrix stack.*
- [MATSTACK](#) [matstack\\_append](#) ([MATSTACK](#) s, [MATRIX](#) a)  
*Appends a matrix to a matrix stack.*
- [MATVEC\\_DPOINTER](#) [matvec\\_creat](#) (void)  
*Creates a matrix-vector pair.*
- int [matvec\\_free](#) ([MATVEC\\_DPOINTER](#) a)  
*Frees a matrix-vector pair.*
- [MATRIX](#) [mat\\_copy](#) ([MATRIX](#) A, [MATRIX](#) result)  
*Copies a matrix.*
- [MATRIX](#) [mat\\_xcopy](#) ([MATRIX](#) A, int si, int ei, int sj, int ej, [MATRIX](#) result)  
*Copies a sub-matrix.*
- [MATRIX](#) [mat\\_xjoin](#) ([MATRIX](#) A11, [MATRIX](#) A12, [MATRIX](#) A21, [MATRIX](#) A22, [MATRIX](#) result)  
*Copies a sub-matrix.*
- [MATRIX](#) [mat\\_rowcopy](#) ([MATRIX](#) A, int rowa, int rowb, [MATRIX](#) result)  
*Copies a row from a matrix.*
- [MATRIX](#) [mat\\_colcopy](#) ([MATRIX](#) A, int cola, int colb, [MATRIX](#) result)  
*Copies a column from a matrix.*
- int [mat\\_fgetmat](#) ([MATRIX](#) A, [MAT\\_FILEPOINTER](#) fp)  
*Gets matrix data from opened file.*
- void [mat\\_dump](#) ([MATRIX](#) A)  
*Dumps a matrix in the stdout.*
- void [mat\\_dumpf](#) ([MATRIX](#) A, const char \*s)  
*Dumps a matrix using a given format specifier in the stdout.*
- void [mat\\_fdump](#) ([MATRIX](#) A, [MAT\\_FILEPOINTER](#) fp)  
*Dumps a matrix in an opened file.*
- void [mat\\_fdumpf](#) ([MATRIX](#) A, const char \*s, [MAT\\_FILEPOINTER](#) fp)  
*Dumps a matrix using a given format specifier in an opened file.*

- void `int_vec_dump` (`INT_VECTOR` a)  
*Dumps an integer vector in the stdout.*
- void `int_vec_dumpf` (`INT_VECTOR` a, const char \*s)  
*Dumps an integer vector using a given format specifier in the stdout.*
- void `int_vec_fdump` (`INT_VECTOR` a, `MAT_FILEPOINTER` fp)  
*Dumps an integer vector in an opened file.*
- void `int_vec_fdumpf` (`INT_VECTOR` a, const char \*s, `MAT_FILEPOINTER` fp)  
*Dumps an integer vector using a given format specifier in an opened file.*
- `INT_VECTOR` `int_vec_copy` (`INT_VECTOR` a, `INT_VECTOR` result)  
*Copies an integer vector.*
- `INT_VECTOR` `int_vec_unique` (`INT_VECTOR` a)  
*Extract only the unique integers from an integer vector.*
- `INT_VECTOR` `int_vec_append` (`INT_VECTOR` a, int i)  
*Appends an integer to an integer vector.*
- `INT_VECTOR` `int_vec_find` (`INT_VECTOR` a, int rel\_type, int n)
- `INT_VECTOR` `int_vec_concat` (`INT_VECTOR` a, `INT_VECTOR` b, `INT_VECTOR` result)  
*Concatenates two integer vectors.*
- `INT_VECTOR` `mat_get_sub_vector` (`INT_VECTOR` a, `INT_VECTOR` indices)  
*Extracts sub-vector from an integer vector.*
- int `gen_error` (int err\_)  
*Generates error message for general errors and exits.*
- `INT_VECTOR` `int_vec_error` (int err\_)  
*Generates error message for integer vector errors and exits.*
- `INT_VECSTACK` `int_vecstack_error` (int err\_)  
*Generates error message for integer vector stack errors and exits.*
- `MATRIX` `mat_error` (int err\_)  
*Generates error message for matrix errors and exits.*
- `MATSTACK` `matstack_error` (int err\_)  
*Generates error message for matrix stack errors and exits.*
- int `stack_error` (int err\_)  
*Generates error message for stack errors and exits.*
- int `queue_error` (int err\_)  
*Generates error message for queue errors and exits.*
- int `pq_error` (int err\_)  
*Generates error message for priority queue errors and exits.*
- int `graph_error` (int err\_)  
*Generates error message for graph errors and exits.*
- mtype `mat_mean` (`MATRIX` A)
- `MATRIX` `mat_mean_row` (`MATRIX` A, `MATRIX` result)
- `MATRIX` `mat_mean_col` (`MATRIX` A, `MATRIX` result)
- mtype `mat_sum` (`MATRIX` A)
- `MATRIX` `mat_sum_row` (`MATRIX` A, `MATRIX` result)
- `MATRIX` `mat_sum_col` (`MATRIX` A, `MATRIX` result)
- `MATRIX` `mat_abs` (`MATRIX` A, `MATRIX` result)  
*Computes absolute value of matrix.*
- `INT_VECTOR` `int_vec_add` (`INT_VECTOR` A, `INT_VECTOR` B, `INT_VECTOR` result)  
*Adds two integer vectors.*
- `INT_VECTOR` `int_vec_adds` (`INT_VECTOR` A, int s, `INT_VECTOR` result)  
*Adds an integer to an integer vector.*
- `INT_VECTOR` `int_vec_sub` (`INT_VECTOR` A, `INT_VECTOR` B, `INT_VECTOR` result)
- `INT_VECTOR` `int_vec_subs` (`INT_VECTOR` A, int s, `INT_VECTOR` result)

- [INT\\_VECTOR int\\_vec\\_mul](#) ([INT\\_VECTOR](#) A, [INT\\_VECTOR](#) B, [INT\\_VECTOR](#) result)
- [INT\\_VECTOR int\\_vec\\_muls](#) ([INT\\_VECTOR](#) A, int s, [INT\\_VECTOR](#) result)
- [INT\\_VECTOR int\\_vec\\_div](#) ([INT\\_VECTOR](#) A, [INT\\_VECTOR](#) B, [INT\\_VECTOR](#) result)  
*Computes element-wise integer vector division.*
- [INT\\_VECTOR int\\_vec\\_divs](#) ([INT\\_VECTOR](#) A, int s, [INT\\_VECTOR](#) result)  
*Divides an integer vector by a scalar.*
- [MATRIX mat\\_add](#) ([MATRIX](#) A, [MATRIX](#) B, [MATRIX](#) result)  
*Adds two matrices.*
- [MATRIX mat\\_adds](#) ([MATRIX](#) A, mtype s, [MATRIX](#) result)  
*Adds a scalar to a matrix.*
- [MATRIX mat\\_sub](#) ([MATRIX](#) A, [MATRIX](#) B, [MATRIX](#) result)
- [MATRIX mat\\_subs](#) ([MATRIX](#) A, mtype s, [MATRIX](#) result)
- [MATRIX mat\\_mul](#) ([MATRIX](#) A, [MATRIX](#) B, [MATRIX](#) result)
- [MATRIX mat\\_mul\\_fast](#) ([MATRIX](#) A, [MATRIX](#) B, [MATRIX](#) result)
- [MATRIX mat\\_mul\\_dot](#) ([MATRIX](#) A, [MATRIX](#) B, [MATRIX](#) result)
- [MATRIX mat\\_muls](#) ([MATRIX](#) A, mtype s, [MATRIX](#) result)
- [MATRIX mat\\_div\\_dot](#) ([MATRIX](#) A, [MATRIX](#) B, [MATRIX](#) result)  
*Computes element-wise matrix division.*
- [MATRIX mat\\_divs](#) ([MATRIX](#) A, mtype s, [MATRIX](#) result)  
*Divides a matrix by a scalar.*
- mtype [mat\\_innerprod](#) ([MATRIX](#) A, [MATRIX](#) B)
- mtype [mat\\_norm\\_inf](#) ([MATRIX](#) A)
- mtype [mat\\_norm\\_one](#) ([MATRIX](#) A)
- mtype [mat\\_norm\\_p](#) ([MATRIX](#) A, mtype p)
- mtype [mat\\_diagmul](#) ([MATRIX](#) A)
- [MATRIX mat\\_tran](#) ([MATRIX](#) A, [MATRIX](#) result)  
*Computes the transpose of a matrix.*
- [MATRIX mat\\_inv](#) ([MATRIX](#) A, [MATRIX](#) result)  
*Computes the inverse of a matrix.*
- [MATRIX mat\\_pinv](#) ([MATRIX](#) A, [MATRIX](#) result)  
*Computes pseudo-inverse of a matrix.*
- [MATRIX mat\\_wpinv](#) ([MATRIX](#) A, [MATRIX](#) w, [MATRIX](#) result)  
*Computes weighted pseudo-inverse of a matrix.*
- [MATRIX mat\\_reg\\_inv](#) ([MATRIX](#) A, mtype r, [MATRIX](#) result)  
*Computes the regularized inverse of a matrix.*
- [MATRIX mat\\_symtoeplz](#) ([MATRIX](#) R, [MATRIX](#) result)
- int [mat\\_lu](#) ([MATRIX](#) A, [MATRIX](#) P)
- void [mat\\_backsubs1](#) ([MATRIX](#) A, [MATRIX](#) B, [MATRIX](#) C, [MATRIX](#) P, int xcol)
- [MATRIX mat\\_lsolve](#) ([MATRIX](#) A, [MATRIX](#) b, [MATRIX](#) result)
- [MATRIX mat\\_cholesky](#) ([MATRIX](#) A, [MATRIX](#) result)  
*Computes Cholesky factor of a matrix.*
- [MATRIX mat\\_conjgrad](#) ([MATRIX](#) A, [MATRIX](#) b, [MATRIX](#) x0, mtype tol, int miters, [MATRIX](#) result)  
*Solves a linear system with conjugate gradients method.*
- [MATRIX mat\\_submat](#) ([MATRIX](#) A, int i, int j, [MATRIX](#) result)  
*Deletes a row and a column of a matrix.*
- mtype [mat\\_cofact](#) ([MATRIX](#) A, int i, int j)  
*Computes a cofactor of a matrix.*
- mtype [mat\\_det](#) ([MATRIX](#) A)  
*Computes the determinant of a matrix.*
- mtype [mat\\_minor](#) ([MATRIX](#) A, int i, int j)  
*Computes a minor of a matrix.*
- [MATSTACK mat\\_qr](#) ([MATRIX](#) A, [MATSTACK](#) qr)

- Computes QR decomposition.*
- [MATRIX mat\\_durbin](#) ([MATRIX](#) R, [MATRIX](#) result)
- Runs Levinson-Durbin algorithm.*
- [MATRIX mat\\_solve\\_durbin](#) ([MATRIX](#) A, [MATRIX](#) B, [MATRIX](#) result)
- Runs Levinson-Durbin algorithm.*
- [mtype mat\\_median](#) ([MATRIX](#) A)
- Computes the median of elements of a given matrix.*
- [mtype mat\\_order\\_statistic](#) ([MATRIX](#) A, int k)
- Computes the  $k^{th}$  order statistic of elements of a given matrix.*
- [void \\_\\_mat\\_quicksort](#) ([MATRIX](#) A, int l, int r, int offset, [MATRIX](#) ind)
- [MATSTACK mat\\_qsort](#) ([MATRIX](#) A, int dim, [MATSTACK](#) result)
- Sorts elements of a given matrix.*
- [MATVEC\\_DPOINTER mat\\_max](#) ([MATRIX](#) A, int dim)
- [MATVEC\\_DPOINTER mat\\_min](#) ([MATRIX](#) A, int dim)
- [MATRIX mat\\_rand](#) (int r, int c, [MATRIX](#) result)
- [MATRIX mat\\_randn](#) (int r, int c, [MATRIX](#) result)
- [MATRIX mat\\_randexp](#) (int r, int c, [mtype](#) mu, [MATRIX](#) result)
- [INT\\_VECTOR int\\_vec\\_permute\\_vect](#) (int n, int k, [INT\\_VECTOR](#) result)
- Computes a randomly permutation of first k positive integers.*
- [MATRIX mat\\_randfun](#) (int r, int c, [mtype](#)(\*fun)([mtype](#)), [mtype](#) xmin, [mtype](#) xmax, [MATRIX](#) result)
- [void mat\\_set\\_seed](#) (int seed)
- [mtype \\_\\_mat\\_randfun](#) ([mtype](#)(\*fun)([mtype](#)), [mtype](#) xmin, [mtype](#) xmax)
- [mtype \\_\\_mat\\_rand](#) ([void](#))
- [mtype \\_\\_mat\\_randn](#) ([void](#))
- [mtype \\_\\_mat\\_randexp](#) ([mtype](#) mu)
- [MATRIX mat\\_randperm](#) (int m, int n, [MATRIX](#) result)
- [MATRIX mat\\_randperm\\_n](#) (int n, [MATRIX](#) result)
- [INT\\_VECTOR int\\_vec\\_randperm](#) (int n, [INT\\_VECTOR](#) result)
- [MATRIX mat\\_least\\_squares](#) ([MATRIX](#) A, [MATRIX](#) Y, [MATRIX](#) result)
- [MATRIX mat\\_w\\_least\\_squares](#) ([MATRIX](#) A, [MATRIX](#) Y, [MATRIX](#) w, [MATRIX](#) result)
- [MATRIX mat\\_rob\\_least\\_squares](#) ([MATRIX](#) A, [MATRIX](#) Y, int lossfunc, [MATRIX](#) result)
- [MATRIX mat\\_linear\\_ls\\_fit](#) ([MATRIX](#) A, [MATRIX](#) Y, int deg, [MATRIX](#) result)
- Polynomial model using least squares.*
- [MATRIX mat\\_robust\\_fit](#) ([MATRIX](#) A, [MATRIX](#) Y, int deg, int lossfunc, [MATRIX](#) result)
- [MATRIX mat\\_concat](#) ([MATRIX](#) A, [MATRIX](#) B, int dim)
- Concatenates two matrices.*
- [MATRIX mat\\_get\\_sub\\_matrix\\_from\\_rows](#) ([MATRIX](#) A, [INT\\_VECTOR](#) indices, [MATRIX](#) result)
- Extracts sub-matrix from rows of a matrix.*
- [MATRIX mat\\_get\\_sub\\_matrix\\_from\\_cols](#) ([MATRIX](#) A, [INT\\_VECTOR](#) indices, [MATRIX](#) result)
- Extracts sub-matrix from columns of a matrix.*
- [MATRIX mat\\_pick\\_row](#) ([MATRIX](#) A, int r, [MATRIX](#) result)
- Picks a row from a matrix.*
- [MATRIX mat\\_pick\\_col](#) ([MATRIX](#) A, int c, [MATRIX](#) result)
- Picks a column from a matrix.*
- [INT\\_VECSTACK mat\\_find](#) ([MATRIX](#) A, int rel\_type, [mtype](#) x)
- [MATRIX mat\\_fliplr](#) ([MATRIX](#) A, [MATRIX](#) result)
- [MATRIX mat\\_flipud](#) ([MATRIX](#) A, [MATRIX](#) result)
- [MATRIX mat\\_calc\\_dist\\_sq](#) ([MATRIX](#) A, [MATRIX](#) d, [MATRIX](#) result)
- Computes the Euclidean distances of points from a given point.*
- [INT\\_VECTOR mat\\_find\\_within\\_dist](#) ([MATRIX](#) A, [MATRIX](#) d, [mtype](#) range)
- Finds points within a neighborhood.*
- [void \\_\\_mat\\_cart2pol](#) ([mtype](#) x, [mtype](#) y, [mtype](#) \*rho, [mtype](#) \*th)



- void `__mat_pol2cart` (mtype rho, mtype th, mtype \*x, mtype \*y)
- `MATRIX mat_cart2pol` (`MATRIX` A, int dim, `MATRIX` result)  
*Converts Cartesian co-ordinates to polar co-ordinates.*
- `MATRIX mat_pol2cart` (`MATRIX` A, int dim, `MATRIX` result)  
*Converts polar co-ordinates to Cartesian co-ordinates.*
- mtype `__mat_addfunc` (mtype x, mtype y)  
*Computes addition function.*
- mtype `__mat_subfunc` (mtype x, mtype y)  
*Computes subtraction function.*
- mtype `__mat_mulfunc` (mtype x, mtype y)  
*Computes multiplication function.*
- mtype `__mat_divfunc` (mtype x, mtype y)  
*Computes division function.*
- mtype `__mat_sqrfunc` (mtype x)  
*Computes square function.*
- mtype `__mat_sqrtfunc` (mtype x)  
*Computes square root function.*
- mtype `__mat_huber_wt` (mtype x, mtype k)  
*Computes Huber weight function.*
- mtype `__mat_bisquare_wt` (mtype x, mtype k)  
*Computes bisquare weight function.*
- mtype `__mat_logplusone` (mtype x)  
*Computes logarithm plus one function.*
- mtype `__mat_arcsinh` (mtype x)  
*Computes inverse hyperbolic sine function.*
- mtype `__mat_arccosh` (mtype x)  
*Computes inverse hyperbolic cosine function.*
- mtype `__mat_arctanh` (mtype x)  
*Computes inverse hyperbolic tangent function.*
- `MATRIX mat_bisquare_wt` (`MATRIX` A, mtype k, mtype sigma, `MATRIX` result)  
*Computes bisquare weight function element-wise on a matrix.*
- `MATRIX mat_huber_wt` (`MATRIX` A, mtype k, mtype sigma, `MATRIX` result)  
*Computes Huber weight function element-wise on a matrix.*
- `MATRIX mat_gfunc` (`MATRIX` A, mtype(\*pt2func)(mtype), `MATRIX` result)  
*Computes a given function element-wise on a matrix.*
- `MATRIX mat_bsxfun` (`MATRIX` A, `MATRIX` B, mtype(\*func)(mtype, mtype), `MATRIX` result)  
*Computes element-wise binary function for two matrices.*
- `MATSTACK mat_corcol` (`MATRIX` data)
- `MATSTACK mat_covcol` (`MATRIX` data)
- `MATRIX mat_scpcol` (`MATRIX` data)
- void `mat_tred2` (`MATRIX` a, `MATRIX` d, `MATRIX` e)
- void `mat_tqli` (`MATRIX` d, `MATRIX` e, `MATRIX` z)
- `MATSTACK mat_pca` (`MATRIX` data, int pca\_type)
- `MATSTACK mat_eig_sym` (`MATRIX` symmat, `MATSTACK` result)
- void `mat_nextline` (void)  
*Prints nextline to stdout.*
- void `mat_fnextline` (`MAT_FILEPOINTER` fp)  
*Prints nextline to file.*
- int `__mat_powerof2` (int width, int \*m, int \*twopm)
- `MATSTACK mat_fft2` (`MATSTACK` c, int dir, `MATSTACK` result)  
*Computes fast Fourier transform.*

- `int __mat_fft (int dir, int m, mtype *x, mtype *y)`
- `MATRIX mat_conv2 (MATRIX A, MATRIX mask, MATRIX scratch, MATRIX result)`  
*Computes 2-D convolution.*
- `INT_VECTOR mat_2int_vec (MATRIX a)`  
*Converts a matrix to an integer vector.*
- `MATRIX int_vec2_mat (INT_VECTOR a, int dir)`  
*Converts an integer vector to a matrix.*
- `MATRIX mat_vectorize (MATRIX a, MATRIX result)`  
*Reshapes a matrix to a vector.*
- `MATRIX mat_vectorize_tr (MATRIX a, MATRIX result)`  
*Reshapes transpose of a matrix to a vector.*
- `mtype mat_int_trapezoid (mtype(*func)(mtype), int n, mtype lower, mtype upper)`  
*Computes trapezoid integration.*
- `mtype mat_int_simpson (mtype(*func)(mtype), int n, mtype lower, mtype upper)`  
*Computes Simpson's integration.*
- `mtype __mat_lint (mtype *x, mtype(*func)(mtype), mtype x0, mtype xn, mtype f0, mtype f2, mtype f3, mtype f5, mtype f6, mtype f7, mtype f9, mtype fl4, mtype hmin, mtype hmax, mtype re, mtype ae)`
- `mtype mat_int_qadtrat (mtype(*func)(mtype), mtype lower, mtype upper)`  
*Computes Gauss quadrature integration.*
- `MATRIX mat_poly_eval (MATRIX A, mtype x, int dir, MATRIX result)`  
*Evaluates polynomial at a point.*
- `MATRIX mat_poly_diff (MATRIX A, int dir, MATRIX result)`  
*Computes derivative polynomial of a polynomial.*
- `MATRIX mat_poly_diff_eval (MATRIX A, mtype x, int dir, MATRIX result)`  
*Evaluates derivative polynomial at a point.*
- `MATRIX mat_poly_add (MATRIX A, MATRIX B, MATRIX result)`  
*Adds two polynomials.*
- `MATRIX mat_poly_mul (MATRIX A, MATRIX B, MATRIX result)`  
*Multiplies two polynomials.*
- `MATSTACK mat_poly_div (MATRIX A, MATRIX B, MATSTACK result)`  
*Divides two polynomials.*
- `MATRIX mat_poly_scale (MATRIX A, mtype s, MATRIX result)`  
*Multiplies a polynomial with a scalar.*
- `MATRIX mat_poly_shift (MATRIX A, int s, MATRIX result)`  
*Shifts a polynomial.*
- `void mat_cheby_init ()`  
*Initializes the Chebyshev polynomial series.*
- `void mat_legendre_init ()`  
*Initializes the Legendre polynomial series.*
- `void mat_binom_init ()`  
*Initializes the binomial series.*
- `MATRIX mat_cheby (int n)`  
*Computes the  $n^{th}$  Chebyshev polynomial.*
- `MATRIX mat_legendre (int n)`  
*Computes the  $n^{th}$  Legendre polynomial.*
- `mtype mat_binom (int n, int k)`  
*Computes a binomial co-efficient.*
- `MATRIX mat_cheby_coeffs_to_poly (MATRIX coeffs, MATRIX result)`  
*Converts Chebyshev co-efficients to a single polynomial.*
- `MATRIX mat_cheby_approx (mtype(*f)(mtype), mtype a, mtype b, int n, MATRIX result)`  
*Approximates a function using Chebyshev polynomials.*

- [MAT\\_BAYES\\_MODEL mat\\_bayes\\_model\\_creat](#) (void)  
*Creates a Bayes model.*
- int [mat\\_bayes\\_model\\_free](#) (MAT\_BAYES\_MODEL a)  
*Frees a Bayes model.*
- [MAT\\_PERCEPTRON mat\\_perceptron\\_creat](#) (void)  
*Creates a perceptron.*
- int [mat\\_perceptron\\_free](#) (MAT\_PERCEPTRON a)  
*Frees a perceptron.*
- [MAT\\_BAYES\\_MODEL mat\\_bayes\\_classifier\\_train](#) (MATRIX data, INT\_VECTOR labels)
- [INT\\_VECTOR mat\\_bayes\\_classifier\\_test](#) (MATRIX data, MAT\_BAYES\_MODEL b\_model)
- [MAT\\_PERCEPTRON mat\\_perceptron\\_train](#) (MATRIX data, INT\_VECTOR labels, int num\_of\_iterations)
- [INT\\_VECTOR mat\\_perceptron\\_test](#) (MATRIX data, MAT\_PERCEPTRON p\_model)
- [MAT\\_PERCEPTRON mat\\_perceptron\\_train\\_](#) (MATRIX data1, MATRIX data2, MAT\_PERCEPTRON p\_model, int class\_num)
- [MATVEC\\_DPOINTER mat\\_kmeans](#) (MATRIX data, int k, int iters, MATVEC\_DPOINTER result)
- [MAT\\_TREE mat\\_bs\\_make\\_null](#) (void)
- [MAT\\_TREE mat\\_bs\\_free](#) (MAT\_TREE T)
- [MAT\\_TREE mat\\_bs\\_find](#) (mtype x, MAT\_TREE T)
- [MAT\\_TREE mat\\_bs\\_find\\_min](#) (MAT\_TREE T)
- [MAT\\_TREE mat\\_bs\\_find\\_max](#) (MAT\_TREE T)
- [MAT\\_TREE mat\\_bs\\_insert](#) (mtype x, MAT\_TREE T)
- [MAT\\_TREE mat\\_bs\\_delete](#) (mtype x, MAT\_TREE T)
- int [mat\\_bs\\_inorder](#) (MAT\_TREE T, int index, mtype \*\*ordered)
- int [gen\\_gt](#) (mtype a)
- int [gen\\_lt](#) (mtype a)
- int [gen\\_eq](#) (mtype a)
- mtype [gen\\_abs\\_ceil](#) (mtype a)
- int [mat\\_isnumeric](#) (MAT\_FILEPOINTER fp)  
*Checks if current word in an opened file is numeric or not.*
- int [mat\\_go\\_next\\_word](#) (MAT\_FILEPOINTER fp)  
*Moves to next word in an opened file.*
- int [mat\\_count\\_words\\_in\\_line](#) (MAT\_FILEPOINTER fp, int \*count)  
*Count words in current line in an opened file.*
- int [mat\\_read\\_word](#) (MAT\_FILEPOINTER fp, char \*c\_word)  
*Reads current word from an opened file.*
- [MATRIX mat\\_dlmread](#) (const char \*fname)  
*Reads a matrix from a file.*
- void [mat\\_dlmwrite](#) (const char \*fname, MATRIX A)  
*Writes a matrix to a file.*
- void [mat\\_tic](#) (void)  
*Starts stopwatch timer.*
- double [mat\\_toc](#) (void)  
*Computes elapsed time from last start of timer.*
- void [mat\\_toc\\_print](#) (void)  
*Computes and prints elapsed time from last start of timer on the stdout.*
- [MAT\\_INT\\_STACK mat\\_int\\_stack\\_creat](#) (void)
- int [mat\\_int\\_stack\\_free](#) (MAT\_INT\_STACK s)
- void [mat\\_int\\_stack\\_push](#) (MAT\_INT\_STACK s, int value)
- int [mat\\_int\\_stack\\_pop](#) (MAT\_INT\_STACK s)
- int [mat\\_int\\_stack\\_is\\_empty](#) (MAT\_INT\_STACK s)
- [MAT\\_MTYPE\\_STACK mat\\_mtype\\_stack\\_creat](#) (void)
- int [mat\\_mtype\\_stack\\_free](#) (MAT\_MTYPE\_STACK s)
- void [mat\\_mtype\\_stack\\_push](#) (MAT\_MTYPE\_STACK s, mtype value)

- `mtype mat_mtype_stack_pop (MAT_MTYPE_STACK s)`
- `int mat_mtype_stack_is_empty (MAT_MTYPE_STACK s)`
- `MAT_INT_QUEUE mat_int_queue_creat (void)`
- `int mat_int_queue_free (MAT_INT_QUEUE s)`
- `void mat_int_queue_enqueue (MAT_INT_QUEUE s, int value)`
- `int mat_int_queue_dequeue (MAT_INT_QUEUE s)`
- `int mat_int_queue_is_empty (MAT_INT_QUEUE s)`
- `MAT_MTYPE_QUEUE mat_mtype_queue_creat (void)`
- `int mat_mtype_queue_free (MAT_MTYPE_QUEUE s)`
- `void mat_mtype_queue_enqueue (MAT_MTYPE_QUEUE s, mtype value)`
- `mtype mat_mtype_queue_dequeue (MAT_MTYPE_QUEUE s)`
- `int mat_mtype_queue_is_empty (MAT_MTYPE_QUEUE s)`
- `MAT_INT_PRIORITYQUEUE mat_int_priorityqueue_creat (int type)`
- `void mat_int_priorityqueue_enqueue (MAT_INT_PRIORITYQUEUE H, int data, int priority)`
- `mat_intpqnode mat_int_priorityqueue_dequeue (MAT_INT_PRIORITYQUEUE H)`
- `int mat_int_priorityqueue_free (MAT_INT_PRIORITYQUEUE H)`
- `int mat_int_priorityqueue_update (MAT_INT_PRIORITYQUEUE H, int data, int priority, int type)`
- `int mat_int_priorityqueue_is_empty (MAT_INT_PRIORITYQUEUE H)`
- `MAT_MTYPE_PRIORITYQUEUE mat_mtype_priorityqueue_creat (int type)`
- `void mat_mtype_priorityqueue_enqueue (MAT_MTYPE_PRIORITYQUEUE H, mtype data, mtype priority)`
- `mat_mtypepqnode mat_mtype_priorityqueue_dequeue (MAT_MTYPE_PRIORITYQUEUE H)`
- `int mat_mtype_priorityqueue_free (MAT_MTYPE_PRIORITYQUEUE H)`
- `int mat_mtype_priorityqueue_update (MAT_MTYPE_PRIORITYQUEUE H, mtype data, mtype priority, int type)`
- `int mat_mtype_priorityqueue_is_empty (MAT_MTYPE_PRIORITYQUEUE H)`
- `MATRIX mat_mds (MATRIX d, int dims, int type, MATRIX result)`
- `MATRIX __mat_mds_metric (MATRIX d, int dims, MATRIX result)`
- `MATRIX __mat_mds_nonmetric (MATRIX d, int dims, MATRIX result)`
- `MAT_GRAPH mat_graph_creat (void)`
- `void mat_graph_adjlist (MAT_GRAPH g, int directed, int weighted, MAT_FILEPOINTER fp)`
- `MAT_INT_QUEUE mat_graph_search (MAT_GRAPH g, int connected, int mst)`
- `void mat_graph_visit (MAT_GRAPH g, int k, int connected, int mst, MAT_INT_PRIORITYQUEUE pq, MAT_INT_QUEUE q)`
- `void mat_graph_dumpf (MAT_GRAPH g, int mst, MAT_FILEPOINTER fp)`
- `void mat_graph_dump (MAT_GRAPH g, int mst)`
- `void mat_graph_adjm_to_adjl (MAT_GRAPH g, MATRIX a)`
- `MAT_GRAPH mat_graph_reverse (MAT_GRAPH g, MAT_GRAPH r)`
- `MAT_KDTREE mat_kdtree_make_tree (MATRIX A, MAT_KDTREE result)`  
*Creates a k-d tree from a data matrix.*
- `int mat_kdtree_free (MAT_KDTREE t)`  
*Frees a k-d tree.*
- `MATRIX mat_kdtree_nearest (MAT_KDTREE t, MATRIX A, MATRIX result)`  
*Computes nearest neighbors.*
- `MATRIX mat_kdtree_k_nearest (MAT_KDTREE t, MATRIX A, int k, MATRIX result)`  
*Computes k nearest neighbors.*
- `MAT_KDNODE __mat_kdtree_make_tree (MAT_KDNODE t, int len, int i, int dim)`
- `MAT_KDNODE __mat_kd_find_median (MAT_KDNODE kd_start, MAT_KDNODE kd_end, int idx)`
- `void __mat_kdtree_nearest (MAT_KDNODE root, MAT_KDNODE nd, int i, int dim, MAT_KDNODE *best, mtype *best_dist)`
- `void __mat_kdtree_k_nearest (MAT_KDNODE root, MAT_KDNODE nd, int i, int dim, MAT_MTYPE_PRIORITYQUEUE pq, MATRIX bmax, MATRIX bmin)`
- `MATSTACK mat_omp (MATRIX A, MATRIX b, int k, mtype tol, MATSTACK result)`

## Variables

- clock\_t [MAT\\_CLOCK\\_TIME](#)
- unsigned int [MAT\\_SEED](#)
- int [MAT\\_SET\\_SEED](#)
- MATSTACK [mat\\_cheby\\_series\\_table](#)
- MATSTACK [mat\\_legendre\\_series\\_table](#)
- MATSTACK [mat\\_binom\\_series\\_table](#)

### 6.35.1 Typedef Documentation

#### 6.35.1.1 typedef INT\_VECTOR\* INT\_VECSTACK

Integer Vector Stack

#### 6.35.1.2 typedef int\* INT\_VECTOR

Integer Vector

#### 6.35.1.3 typedef struct mat\_bayes\_model mat\_bayes\_model

Bayes Classifier Model Structure.

Bayes Classifier Model

#### 6.35.1.4 typedef mat\_bayes\_model\* MAT\_BAYES\_MODEL

Bayes Classifier Model Pointer

#### 6.35.1.5 typedef struct mat\_gnode mat\_gnode

Graph Node Structure.

Graph Node

#### 6.35.1.6 typedef mat\_gnode\* MAT\_GNODE

Graph Node Pointer

#### 6.35.1.7 typedef struct mat\_graph mat\_graph

Graph Structure.

#### 6.35.1.8 typedef mat\_graph\* MAT\_GRAPH

#### 6.35.1.9 typedef struct mat\_int\_priorityqueue mat\_int\_priorityqueue

Integer Priority Queue Structure.

Integer Priority Queue

6.35.1.10 `typedef mat_int_priorityqueue* MAT_INT_PRIORITYQUEUE`

Integer Priority Queue Pointer

6.35.1.11 `typedef struct mat_int_queue mat_int_queue`

Integer Queue Structure.

Integer Queue

6.35.1.12 `typedef mat_int_queue* MAT_INT_QUEUE`

Integer Queue Pointer

6.35.1.13 `typedef struct mat_int_stack mat_int_stack`

Integer Stack Structure.

Integer Stack

6.35.1.14 `typedef mat_int_stack* MAT_INT_STACK`

Integer Stack Pointer

6.35.1.15 `typedef struct mat_intpqnode mat_intpqnode`

Integer Priority Queue Node Structure.

Integer Priority Queue Node

6.35.1.16 `typedef mat_intpqnode* MAT_INTPQNODE`

Integer Priority Queue Node Pointer

6.35.1.17 `typedef struct mat_kdnode mat_kdnode`

6.35.1.18 `typedef mat_kdnode* MAT_KDNODE`

6.35.1.19 `typedef struct mat_kdtree mat_kdtree`

6.35.1.20 `typedef mat_kdtree* MAT_KDTREE`

6.35.1.21 `typedef struct mat_mtype_priorityqueue mat_mtype_priorityqueue`

Mtype Priority Queue Structure.

Mtype Priority Queue

6.35.1.22 `typedef mat_mtype_priorityqueue* MAT_MTYPE_PRIORITYQUEUE`

Mtype Priority Queue Pointer

6.35.1.23 `typedef struct mat_mtype_queue mat_mtype_queue`

Mtype Queue Structure.

Mtype Queue

6.35.1.24 `typedef mat_mtype_queue* MAT_MTYPE_QUEUE`

Mtype Queue Pointer

6.35.1.25 `typedef struct mat_mtype_stack mat_mtype_stack`

Mtype Stack Structure.

Mtype Stack

6.35.1.26 `typedef mat_mtype_stack* MAT_MTYPE_STACK`

Mtype Stack Pointer

6.35.1.27 `typedef struct mat_mtypepqnode mat_mtypepqnode`

Mtype Priority Queue Node Structure.

Mtype Priority Queue Node

6.35.1.28 `typedef mat_mtypepqnode* MAT_MTYPEPQNODE`

Mtype Priority Queue Node Pointer

6.35.1.29 `typedef struct mat_perceptron mat_perceptron`

Perceptron Classifier Model Structure.

Perceptron Classifier Model

6.35.1.30 `typedef mat_perceptron* MAT_PERCEPTRON`

Perceptron Classifier Model Pointer

6.35.1.31 `typedef struct mat_qintnode mat_qintnode`

Integer Queue Node Structure.

Integer Queue Node

6.35.1.32 `typedef mat_qintnode* MAT_QINTNODE`

Integer Queue Node Pointer

6.35.1.33 `typedef struct mat_qmtypenode mat_qmtypenode`

Mtype Queue Node Structure.

Mtype Queue Node

6.35.1.34 `typedef mat_qmtypenode* MAT_QMTYPENODE`

Mtype Queue Node Pointer

6.35.1.35 `typedef mat_tree_node* MAT_TREE`

Search Tree Pointer

6.35.1.36 `typedef struct mat_tree_node mat_tree_node`

Search Tree Node Structure.

Search Tree Node

6.35.1.37 `typedef mat_tree_node* MAT_TREE_NODE`

Search Tree Node Pointer

6.35.1.38 `typedef mtype** MATRIX`

Mtype Matrix

6.35.1.39 `typedef MATRIX* MATSTACK`

Mtype Matrix Stack

6.35.1.40 `typedef void** MATVEC_DPOINTER`

Mtype Matrix - Integer Vector Pair

## 6.35.2 Function Documentation

6.35.2.1 `INT_VECTOR __int_vec_creat ( int len )`

6.35.2.2 `INT_VECSTACK __int_vecstack_creat ( int len )`

6.35.2.3 `mtype __mat_addfunc ( mtype x, mtype y )`

Computes addition function.

### Parameters

<code>in</code>	<code>x</code>	
<code>in</code>	<code>y</code>	



## Returns

$$x + y$$

## 6.35.2.4 mtype \_\_mat\_arccosh ( mtype x )

Computes inverse hyperbolic cosine function.

## Parameters

in	x	
----	---	--

## Returns

$$\cosh^{-1}(x)$$

## 6.35.2.5 mtype \_\_mat\_arcsinh ( mtype x )

Computes inverse hyperbolic sine function.

## Parameters

in	x	
----	---	--

## Returns

$$\sinh^{-1}(x)$$

## 6.35.2.6 mtype \_\_mat\_arctanh ( mtype x )

Computes inverse hyperbolic tangent function.

## Parameters

in	x	
----	---	--

## Returns

$$\tanh^{-1}(x)$$

## 6.35.2.7 mtype \_\_mat\_bisquare\_wt ( mtype x, mtype k )

Computes bisquare weight function.

## Parameters

in	x	
in	k	

## Returns

$$\begin{cases} \left(1 - \left(\frac{x}{k}\right)^2\right)^2, & \text{for } |x| \leq k, \\ 0, & \text{otherwise.} \end{cases}$$

6.35.2.8 void \_\_mat\_cart2pol ( mtype *x*, mtype *y*, mtype \* *rho*, mtype \* *th* )

6.35.2.9 MATRIX \_\_mat\_creat ( int *r*, int *c* )

6.35.2.10 mtype \_\_mat\_divfunc ( mtype *x*, mtype *y* )

Computes division function.

#### Parameters

in	<i>x</i>	
in	<i>y</i>	

#### Returns

$$\frac{x}{y}$$

6.35.2.11 int \_\_mat\_fft ( int *dir*, int *m*, mtype \* *x*, mtype \* *y* )

6.35.2.12 mtype \_\_mat\_huber\_wt ( mtype *x*, mtype *k* )

Computes Huber weight function.

#### Parameters

in	<i>x</i>	
in	<i>k</i>	

#### Returns

$$\begin{cases} 1, & \text{for } |x| \leq k, \\ \frac{k}{|x|}, & \text{otherwise.} \end{cases}$$

6.35.2.13 MAT\_KDNODE \_\_mat\_kd.find\_median ( MAT\_KDNODE *kd\_start*, MAT\_KDNODE *kd\_end*, int *idx* )

6.35.2.14 void \_\_mat\_kdtree\_k\_nearest ( MAT\_KDNODE *root*, MAT\_KDNODE *nd*, int *i*, int *dim*,  
MAT\_MTYPE\_PRIORITYQUEUE *pq*, MATRIX *bmax*, MATRIX *bmin* )

6.35.2.15 MAT\_KDNODE \_\_mat\_kdtree.make.tree ( MAT\_KDNODE *t*, int *len*, int *i*, int *dim* )

6.35.2.16 void \_\_mat\_kdtree\_nearest ( MAT\_KDNODE *root*, MAT\_KDNODE *nd*, int *i*, int *dim*, MAT\_KDNODE \* *best*,  
mtype \* *best\_dist* )

6.35.2.17 mtype \_\_mat\_lint ( mtype \* *x*, mtype(\*) (mtype) *func*, mtype *x0*, mtype *xn*, mtype *f0*, mtype *f2*, mtype *f3*, mtype *f5*,  
mtype *f6*, mtype *f7*, mtype *f9*, mtype *f14*, mtype *hmin*, mtype *hmax*, mtype *re*, mtype *ae* )

6.35.2.18 mtype \_\_mat\_logplusone ( mtype *x* )

Computes logarithm plus one function.

#### Parameters

in	<i>x</i>	
----	----------	--

## Returns

 $\log(1+x)$ 

6.35.2.19 **MATRIX** \_\_mat\_mds\_metric ( **MATRIX** *d*, int *dims*, **MATRIX** *result* )

6.35.2.20 **MATRIX** \_\_mat\_mds\_nonmetric ( **MATRIX** *d*, int *dims*, **MATRIX** *result* )

6.35.2.21 **mtype** \_\_mat\_mulfunc ( **mtype** *x*, **mtype** *y* )

Computes multiplication function.

## Parameters

<i>in</i>	<i>x</i>	
<i>in</i>	<i>y</i>	

## Returns

 $xy$ 

6.35.2.22 **void** \_\_mat\_pol2cart ( **mtype** *rho*, **mtype** *th*, **mtype** \* *x*, **mtype** \* *y* )

6.35.2.23 **int** \_\_mat\_powerof2 ( int *width*, int \* *m*, int \* *twopm* )

6.35.2.24 **void** \_\_mat\_quicksort ( **MATRIX** *A*, int *l*, int *r*, int *offset*, **MATRIX** *ind* )

6.35.2.25 **mtype** \_\_mat\_rand ( **void** )

6.35.2.26 **mtype** \_\_mat\_randexp ( **mtype** *mu* )

6.35.2.27 **mtype** \_\_mat\_randfun ( **mtype**(\*)(**mtype**) *fun*, **mtype** *xmin*, **mtype** *xmax* )

6.35.2.28 **mtype** \_\_mat\_randn ( **void** )

6.35.2.29 **mtype** \_\_mat\_sqrfunc ( **mtype** *x* )

Computes square function.

## Parameters

<i>in</i>	<i>x</i>	
-----------	----------	--

## Returns

 $x^2$ 

6.35.2.30 **mtype** \_\_mat\_sqrtfunc ( **mtype** *x* )

Computes square root function.

## Parameters

<i>in</i>	<i>x</i>	
-----------	----------	--

## Returns

$$\sqrt{x}$$

6.35.2.31 `mtype __mat_subfunc ( mtype x, mtype y )`

Computes subtraction function.

## Parameters

<i>in</i>	<i>x</i>	
<i>in</i>	<i>y</i>	

## Returns

$$x - y$$

6.35.2.32 **MATSTACK** `__matstack_creat ( int len )`

6.35.2.33 `mtype gen_abs_ceil ( mtype a )`

6.35.2.34 `int gen_eq ( mtype a )`

6.35.2.35 `int gen_error ( int err_ )`

Generates error message for general errors and exits.

## Parameters

<i>in</i>	<i>err</i>	Error type (GEN_NOT_CONVERGED/GEN_FNOTOPEN/ GEN_FNOTGETMAT/GEN_SIZEMISMATCH/GEN_MATH_ERROR/GEN_MALLOC/GEN_NOT_FOUND/GEN_SIZE_ERROR/GEN_BAD_TYPE)
-----------	------------	--

6.35.2.36 `int gen_gt ( mtype a )`

6.35.2.37 `int gen_lt ( mtype a )`

6.35.2.38 `int graph_error ( int err_ )`

Generates error message for graph errors and exits.

## Parameters

<i>in</i>	<i>err</i>	Error type (GRAPH_MALLOC/GRAPH_READ/GRAPH_ELSE)
-----------	------------	---

6.35.2.39 **MATRIX** `int_vec2_mat ( INT_VECTOR a, int dir )`

Converts an integer vector to a matrix.

## Parameters

<i>in</i>	<i>a</i>	Input vector
<i>in</i>	<i>dir</i>	Conversion direction

## Returns

Output matrix

6.35.2.40 `INT_VECTOR int_vec.add ( INT_VECTOR A, INT_VECTOR B, INT_VECTOR result )`

Adds two integer vectors.

## Parameters

<code>in</code>	<code>A</code>	Input vector
<code>in</code>	<code>B</code>	Input vector
<code>in</code>	<code>result</code>	Vector to store the result

## Returns

$\mathbf{A} + \mathbf{B}$

6.35.2.41 `INT_VECTOR int_vec.adds ( INT_VECTOR A, int s, INT_VECTOR result )`

Adds an integer to an integer vector.

## Parameters

<code>in</code>	<code>A</code>	Input vector
<code>in</code>	<code>s</code>	Input scalar
<code>in</code>	<code>result</code>	Vector to store the result

## Returns

$\mathbf{A} + s\mathbf{1}$

6.35.2.42 `INT_VECTOR int_vec.append ( INT_VECTOR a, int i )`

Appends an integer to an integer vector.

## Parameters

<code>in</code>	<code>a</code>	Input vector
<code>in</code>	<code>i</code>	Integer to append

## Returns

Appended vector

6.35.2.43 `INT_VECTOR int_vec.concat ( INT_VECTOR a, INT_VECTOR b, INT_VECTOR result )`

Concatenates two integer vectors.

## Parameters

<code>in</code>	<code>a</code>	Input first vector
<code>in</code>	<code>b</code>	Input second vector
<code>in</code>	<code>dim</code>	Concatenation direction (ROWS/COLS)

**Returns**

$$\begin{bmatrix} a & b \end{bmatrix} \text{ or } \begin{bmatrix} a \\ b \end{bmatrix}$$

**6.35.2.44 INT\_VECTOR int\_vec\_copy ( INT\_VECTOR *a*, INT\_VECTOR *result* )**

Copies an integer vector.

**Parameters**

<i>in</i>	<i>a</i>	Input vector
<i>in</i>	<i>result</i>	Vector to store the result

**Returns**

Output vector

**6.35.2.45 INT\_VECTOR int\_vec\_creat ( int *len*, int *type* )**

Creates an integer vector.

**Parameters**

<i>in</i>	<i>len</i>	Length of the vector
<i>in</i>	<i>type</i>	Definition type (UNDEFINED/ZERO_INT_VECTOR/ONES_INT_VECTOR/SERIES_INT_VECTOR)

**Returns**

Output vector

**6.35.2.46 INT\_VECTOR int\_vec\_div ( INT\_VECTOR *A*, INT\_VECTOR *B*, INT\_VECTOR *result* )**

Computes element-wise integer vector division.

**Parameters**

<i>in</i>	<i>A</i>	First input vector
<i>in</i>	<i>B</i>	Second input vector
<i>in</i>	<i>result</i>	Vector to store the result

**Returns**

$$A./B$$

**6.35.2.47 INT\_VECTOR int\_vec\_divs ( INT\_VECTOR *A*, int *x*, INT\_VECTOR *result* )**

Divides an integer vector by a scalar.

**Parameters**

<i>in</i>	<i>A</i>	Input vector
<i>in</i>	<i>s</i>	Scalar
<i>in</i>	<i>result</i>	Vector to store the result

Returns

$$\frac{A}{s}$$
**6.35.2.48** void int\_vec\_dump ( INT\_VECTOR *A* )

Dumps an integer vector in the stdout.

Parameters

in	<i>A</i>	Input vector
----	----------	--------------

**6.35.2.49** void int\_vec\_dumpf ( INT\_VECTOR *A*, const char \* *s* )

Dumps an integer vector using a given format specifier in the stdout.

Parameters

in	<i>A</i>	Input vector
in	<i>s</i>	Format specifier

**6.35.2.50** INT\_VECTOR int\_vec\_error ( int *err\_* )

Generates error message for integer vector errors and exits.

Parameters

in	<i>err</i>	Error type (INT_VEC_MALLOC/INT_VEC_FNOTOPEN/INT_VEC_FNOTGET-INT_VEC/INT_VEC_SIZEMISMATCH)
----	------------	---

**6.35.2.51** void int\_vec\_fdump ( INT\_VECTOR *A*, MAT\_FILEPOINTER *fp* )

Dumps an integer vector in an opened file.

Parameters

in	<i>A</i>	Input vector
in	<i>fp</i>	Pointer to an opened file

**6.35.2.52** void int\_vec\_dumpf ( INT\_VECTOR *A*, const char \* *s*, MAT\_FILEPOINTER *fp* )

Dumps an integer vector using a given format specifier in an opened file.

Parameters

in	<i>A</i>	Input vector
in	<i>s</i>	Format specifier
in	<i>fp</i>	Pointer to an opened file

**6.35.2.53 INT\_VECTOR int\_vec\_fill ( INT\_VECTOR A, int val )**

Fills an integer vector with a value.

**Parameters**

in	A	Input vector
in	val	Value to fill with

**Returns**

Filled vector

**6.35.2.54 INT\_VECTOR int\_vec\_fill\_type ( INT\_VECTOR A, int type )**

Fills an integer vector to a type.

**Parameters**

in	A	Input vector
in	type	Definition type (UNDEFINED/ZERO_INT_VECTOR/ONES_INT_VECTOR/SERIES_INT_VECTOR)

**Returns**

Filled vector

**6.35.2.55 INT\_VECTOR int\_vec\_find ( INT\_VECTOR a, int rel\_type, int n )****6.35.2.56 int int\_vec\_free ( INT\_VECTOR A )**

Frees an integer vector.

**Parameters**

in	A	Input vector
----	---	--------------

**Returns**

Success

**6.35.2.57 INT\_VECTOR int\_vec\_mul ( INT\_VECTOR A, INT\_VECTOR B, INT\_VECTOR result )****6.35.2.58 INT\_VECTOR int\_vec\_muls ( INT\_VECTOR A, int s, INT\_VECTOR result )****6.35.2.59 INT\_VECTOR int\_vec\_permute\_vect ( int n, int k, INT\_VECTOR result )**

Computes a randomly permutation of first k positive integers.

**Parameters**

in	n	Number of random permutations to make
in	k	Integer upto which it will consider
in	result	Vector to store the result



## Returns

Permuted vector

6.35.2.60 **INT\_VECTOR** `int_vec_randperm ( int n, INT_VECTOR result )`

6.35.2.61 **INT\_VECTOR** `int_vec_sub ( INT_VECTOR A, INT_VECTOR B, INT_VECTOR result )`

6.35.2.62 **INT\_VECTOR** `int_vec_subs ( INT_VECTOR A, int s, INT_VECTOR result )`

6.35.2.63 **INT\_VECTOR** `int_vec_unique ( INT_VECTOR a )`

Extract only the unique integers from an integer vector.

## Parameters

<i>in</i>	<i>a</i>	Input vector
-----------	----------	--------------

## Returns

Unique vector

6.35.2.64 **INT\_VECSTACK** `int_vecstack_creat ( int len )`

Creates an integer vector stack.

## Parameters

<i>in</i>	<i>len</i>	Length of the stack
-----------	------------	---------------------

## Returns

Output vector stack

6.35.2.65 **INT\_VECSTACK** `int_vecstack_error ( int err_ )`

Generates error message for integer vector stack errors and exits.

## Parameters

<i>in</i>	<i>err</i>	Error type (INT_VECSTACK_MALLOC/INT_VECSTACK_FNOTOPEN/INT_VECSTACK_FNOTGETINT_VEC/INT_VECSTACK_SIZEMISMATCH)
-----------	------------	--

6.35.2.66 **int** `int_vecstack_free ( INT_VECSTACK A )`

Frees an integer vector stack.

## Parameters

<i>in</i>	<i>A</i>	Input vector stack
-----------	----------	--------------------

## Returns

Success

### 6.35.2.67 INT\_VECTOR mat\_2int\_vec ( MATRIX A )

Converts a matrix to an integer vector.

## Parameters

in	A	Input matrix
out	v	Output vector

## Returns

Output vector

### 6.35.2.68 MATRIX mat\_abs ( MATRIX A, MATRIX result )

Computes absolute value of matrix.

## Parameters

in	A	Input matrix
in	result	Matrix to store the result

## Returns

$abs(A)$

### 6.35.2.69 MATRIX mat\_add ( MATRIX A, MATRIX B, MATRIX result )

Adds two matrices.

## Parameters

in	A	Input matrix
in	B	Input matrix
in	result	Matrix to store the result

## Returns

$A + B$

### 6.35.2.70 MATRIX mat\_adds ( MATRIX A, mtype s, MATRIX result )

Adds a scalar to a matrix.

## Parameters

in	A	Input matrix
in	s	Input scalar
in	result	Matrix to store the result

Returns

$$\mathbf{A} + s\mathbf{1}\mathbf{1}^T$$

6.35.2.71 void mat\_backsubs1 ( MATRIX *A*, MATRIX *B*, MATRIX *C*, MATRIX *P*, int *xcol* )

6.35.2.72 INT\_VECTOR mat\_bayes\_classifier\_test ( MATRIX *data*, MAT\_BAYES\_MODEL *b\_model* )

6.35.2.73 MAT\_BAYES\_MODEL mat\_bayes\_classifier\_train ( MATRIX *data*, INT\_VECTOR *labels* )

6.35.2.74 MAT\_BAYES\_MODEL mat\_bayes\_model\_creat ( void )

Creates a Bayes model.

Returns

Output Bayes model

6.35.2.75 int mat\_bayes\_model\_free ( MAT\_BAYES\_MODEL *a* )

Frees a Bayes model.

Parameters

in	<i>a</i>	Input Bayes model
----	----------	-------------------

Returns

Success

6.35.2.76 mtype mat\_binom ( int *n*, int *k* )

Computes a binomial co-efficient.

Parameters

in	<i>n</i>	1 <sup>st</sup> argument
in	<i>k</i>	2 <sup>nd</sup> argument

Returns

$$\binom{n}{k}$$

6.35.2.77 void mat\_binom\_init ( )

Initializes the binomial series.

6.35.2.78 MATRIX mat\_bisquare\_wt ( MATRIX *A*, mtype *k*, mtype *sigma*, MATRIX *result* )

Computes bisquare weight function element-wise on a matrix.

## Parameters

<code>in</code>	<code>A</code>	Input matrix
<code>in</code>	<code>k</code>	Bisquare parameter

## Returns

$\mathbf{B}$ ,  $b_{ij} = f_k(a_{ij})$  where  $f_k$  is the biquare weight function

6.35.2.79 **MAT\_TREE** `mat_bs_delete ( mtype x, MAT_TREE T )`

6.35.2.80 **MAT\_TREE** `mat_bs_find ( mtype x, MAT_TREE T )`

6.35.2.81 **MAT\_TREE** `mat_bs_find_max ( MAT_TREE T )`

6.35.2.82 **MAT\_TREE** `mat_bs_find_min ( MAT_TREE T )`

6.35.2.83 **MAT\_TREE** `mat_bs_free ( MAT_TREE T )`

6.35.2.84 `int` `mat_bs_inorder ( MAT_TREE T, int index, mtype ** ordered )`

6.35.2.85 **MAT\_TREE** `mat_bs_insert ( mtype x, MAT_TREE T )`

6.35.2.86 **MAT\_TREE** `mat_bs_make_null ( void )`

6.35.2.87 **MATRIX** `mat_bsxfun ( MATRIX A, MATRIX B, mtype(*) (mtype, mtype) func, MATRIX result )`

Computes element-wise binary function for two matrices.

## Parameters

<code>in</code>	<code>A</code>	First matrix
<code>in</code>	<code>B</code>	Second matrix
<code>in</code>	<code>func</code>	Pointer to the function
<code>in</code>	<code>result</code>	Matrix to store the result

## Returns

Output matrix

6.35.2.88 **MATRIX** `mat_calc_dist_sq ( MATRIX A, MATRIX d, MATRIX result )`

Computes the Euclidean distances of points from a given point.

## Parameters

<code>in</code>	<code>A</code>	Points matrix (d x N)
<code>in</code>	<code>d</code>	Matrix point from which the distance to be computed (d x 1)
<code>in</code>	<code>result</code>	Matrix to store the result

## Returns

Euclidean distance matrix

**6.35.2.89 MATRIX mat\_cart2pol ( MATRIX *A*, int *dim*, MATRIX *result* )**

Converts Cartesian co-ordinates to polar co-ordinates.

**Parameters**

in	<i>A</i>	Input matrix
in	<i>dim</i>	Data order ROWS/COLS

**Returns**

Polar co-ordinate matrix

**6.35.2.90 MATRIX mat\_cheby ( int *n* )**

Computes the  $n^{th}$  Chebyshev polynomial.

**Parameters**

in	<i>n</i>	Polynomial series index
----	----------	-------------------------

**Returns**

Output polynomial matrix

**6.35.2.91 MATRIX mat\_cheby\_approx ( mtype(\*)(*mtype*) *f*, mtype *a*, mtype *b*, int *n*, MATRIX *result* )**

Approximates a function using Chebyshev polynomials.

**Parameters**

in	<i>f</i>	Function to approximate
in	<i>a</i>	Lower limit of domain of the function
in	<i>b</i>	Upper limit of domain of the function
in	<i>n</i>	Degree of the approximate polynomial
in	<i>result</i>	Matrix to store the result

**Returns**

Approximate polynomial matrix

**6.35.2.92 MATRIX mat\_cheby\_coeffs\_to\_poly ( MATRIX *coeffs*, MATRIX *result* )**

Converts Chebyshev co-efficients to a single polynomial.

**Parameters**

in	<i>coeffs</i>	Chebyshev polynomial co-efficient matrix
in	<i>result</i>	Matrix to store the result

**Returns**

Polynomial matrix

**6.35.2.93 void mat\_cheby\_init ( )**

Initializes the Chebyshev polynomial series.

**6.35.2.94 MATRIX mat\_cholesky ( MATRIX *A*, MATRIX *result* )**

Computes Cholesky factor of a matrix.

**Parameters**

<i>in</i>	<i>A</i>	Input matrix
<i>in</i>	<i>result</i>	Matrix to store the result

**Returns**

Cholesky factor

**6.35.2.95 mtype mat\_cofact ( MATRIX *A*, int *i*, int *j* )**

Computes a cofactor of a matrix.

**Parameters**

<i>in</i>	<i>A</i>	Input matrix
<i>in</i>	<i>i</i>	Row index
<i>in</i>	<i>j</i>	Column index

**Returns**

Cofactor  $C_{ij}$

**6.35.2.96 MATRIX mat\_colcopy ( MATRIX *A*, int *cola*, int *colb*, MATRIX *result* )**

Copies a column from a matrix.

**Parameters**

<i>in</i>	<i>A</i>	Input matrix
<i>in</i>	<i>cola</i>	Source column
<i>in</i>	<i>colb</i>	Destination column
<i>in</i>	<i>result</i>	Matrix to store the result

**Returns**

Copied matrix

**6.35.2.97 MATRIX mat\_concat ( MATRIX *A*, MATRIX *B*, int *dim* )**

Concatenates two matrices.

**Parameters**

<i>in</i>	<i>A</i>	Input first matrix
<i>in</i>	<i>B</i>	Input second matrix
<i>in</i>	<i>dim</i>	Concatenation direction (ROWS/COLS)

Returns

$$\begin{bmatrix} A & B \end{bmatrix} \text{ or } \begin{bmatrix} A \\ B \end{bmatrix}$$

**6.35.2.98** **MATRIX** mat\_conjgrad ( **MATRIX** *A*, **MATRIX** *b*, **MATRIX** *x0*, *mtype tol*, *int miters*, **MATRIX** *result* )

Solves a linear system with conjugate gradients method.

Parameters

<i>in</i>	<i>A</i>	Input matrix
<i>in</i>	<i>b</i>	Observed matrix
<i>in</i>	<i>result</i>	Matrix to store the result

Returns

*x*

**6.35.2.99** **MATRIX** mat\_conv2 ( **MATRIX** *A*, **MATRIX** *mask*, **MATRIX** *scratch*, **MATRIX** *result* )

Computes 2-D convolution.

Parameters

<i>in</i>	<i>A</i>	Input matrix
<i>in</i>	<i>mask</i>	Input kernel/mask
<i>in</i>	<i>scratch</i>	Scratch matrix for temporary calculations
<i>in</i>	<i>result</i>	Matrix to store the result

Returns

Convolved output matrix

**6.35.2.100** **MATRIX** mat\_copy ( **MATRIX** *A*, **MATRIX** *result* )

Copies a matrix.

Parameters

<i>in</i>	<i>A</i>	Input matrix
<i>in</i>	<i>result</i>	Matrix to store the result

Returns

Output matrix

**6.35.2.101** **MATSTACK** mat\_corcol ( **MATRIX** *data* )

**6.35.2.102** **int** mat\_count\_words\_in\_line ( **MAT\_FILEPOINTER** *fp*, **int** \* *count* )

Count words in current line in an opened file.

## Parameters

in	<i>fp</i>	Pointer to an opened file
out	<i>count</i>	Pointer to output count

## Returns

EOF reached

6.35.2.103 **MATSTACK** `mat_covcol ( MATRIX data )`

6.35.2.104 **MATRIX** `mat_creat ( int row, int col, int type )`

Creates a matrix.

## Parameters

in	<i>row</i>	Number of rows
in	<i>col</i>	Number of columns
in	<i>type</i>	Definition type (UNDEFINED/ZERO_MATRIX/UNIT_MATRIX/ONES_MATRIX)

## Returns

Output matrix

6.35.2.105 **MATRIX** `mat_creat_diag ( MATRIX diag_vals, MATRIX result )`

Creates a diagonal matrix from a 1-d matrix.

## Parameters

in	<i>diag_vals</i>	Input 1-d diagonal value matrix
in	<i>result</i>	Matrix to store the result

## Returns

Diagonal matrix

6.35.2.106 **mtypes** `mat_det ( MATRIX A )`

Computes the determinant of a matrix.

## Parameters

in	<i>A</i>	Input matrix
----	----------	--------------

## Returns

$\det(A)$

6.35.2.107 **mtypes** `mat_diagmul ( MATRIX A )`



**6.35.2.108 MATRIX mat\_div\_dot ( MATRIX *A*, MATRIX *B*, MATRIX *result* )**

Computes element-wise matrix division.

**Parameters**

in	<i>A</i>	First input matrix
in	<i>B</i>	Second input matrix
in	<i>result</i>	Matrix to store the result

**Returns**

$$A./B$$

**6.35.2.109 MATRIX mat\_divs ( MATRIX *A*, mtype *s*, MATRIX *result* )**

Divides a matrix by a scalar.

**Parameters**

in	<i>A</i>	Input matrix
in	<i>s</i>	Scalar
in	<i>result</i>	Matrix to store the result

**Returns**

$$\frac{A}{s}$$

**6.35.2.110 MATRIX mat\_dlmread ( const char \* *fname* )**

Reads a matrix from a file.

**Parameters**

in	<i>fname</i>	Filename to read from
----	--------------	-----------------------

**Returns**

Output matrix

**6.35.2.111 void mat\_dlmwrite ( const char \* *fname*, MATRIX *A* )**

Writes a matrix to a file.

**Parameters**

in	<i>fname</i>	Filename to write into
in	<i>A</i>	Input matrix

**6.35.2.112 void mat\_dump ( MATRIX *A* )**

Dumps a matrix in the stdout.

## Parameters

in	A	Input matrix
----	---	--------------

## 6.35.2.113 void mat\_dumpf ( MATRIX A, const char \* s )

Dumps a matrix using a given format specifier in the stdout.

## Parameters

in	A	Input matrix
in	s	Format specifier

## 6.35.2.114 MATRIX mat\_durbin ( MATRIX R, MATRIX result )

Runs Levinson-Durbin algorithm.

## Parameters

in	R	Input $n^th$ correlation matrix $(n+1) \times 1$
in	result	Matrix to store the result

## Returns

$$X \text{ where } \tilde{R}X = B, \tilde{R} = \begin{bmatrix} R[0][0] & R[1][0] & \cdots & R[n-1][0] \\ R[1][0] & R[0][0] & \cdots & R[n-2][0] \\ \vdots & \vdots & \ddots & \vdots \\ R[n-1][0] & R[n-2][0] & \cdots & R[0][0] \end{bmatrix} \text{ and } B = [R[1][0] \ R[2][0] \ \cdots \ R[n][0]]$$

## 6.35.2.115 MATSTACK mat\_eig\_sym ( MATRIX symmat, MATSTACK result )

## 6.35.2.116 MATRIX mat\_error ( int err\_ )

Generates error message for matrix errors and exits.

## Parameters

in	err	Error type (MAT_MALLOC/MAT_FNOTOPEN/MAT_FNOTGETMAT/MAT_SIZEMISMATCH/ MAT_INVERSE_ILL_COND/MAT_INVERSE_NOT_SQUARE/MAT_CHOLESKY_FAILED)
----	-----	---

## 6.35.2.117 void mat\_fdump ( MATRIX A, MAT\_FILEPOINTER fp )

Dumps a matrix in an opened file.

## Parameters

in	A	Input matrix
in	fp	Pointer to an opened file

## 6.35.2.118 void mat\_fdumpf ( MATRIX A, const char \* s, MAT\_FILEPOINTER fp )

Dumps a matrix using a given format specifier in an opened file.

## Parameters

<i>in</i>	<i>A</i>	Input matrix
<i>in</i>	<i>s</i>	Format specifier
<i>in</i>	<i>fp</i>	Pointer to an opened file

6.35.2.119 **MATSTACK** `mat_fft2 ( MATSTACK c, int dir, MATSTACK result )`

Computes fast Fourier transform.

## Parameters

<i>in</i>	<i>C</i>	Complex data matrix stack
<i>in</i>	<i>dir</i>	FFT direction (ROWS/COLS)
<i>in</i>	<i>result</i>	Matrix stack to store the result

## Returns

Transformed matrix stack

6.35.2.120 **int** `mat_fgetmat ( MATRIX A, MAT_FILEPOINTER fp )`

Gets matrix data from opened file.

## Parameters

<i>in</i>	<i>A</i>	Matrix to store the data
<i>in</i>	<i>fp</i>	Pointer to opened file

## Returns

Number of elements copied

6.35.2.121 **MATRIX** `mat_fill ( MATRIX A, mtype val )`

Fills a matrix with a value.

## Parameters

<i>in</i>	<i>A</i>	Input matrix
<i>in</i>	<i>val</i>	Value to fill with

## Returns

Filled matrix

6.35.2.122 **MATRIX** `mat_fill_type ( MATRIX A, int type )`

Fills a matrix to a type.

## Parameters

<i>in</i>	<i>A</i>	Input matrix
<i>in</i>	<i>type</i>	Fill type (UNDEFINED/ZERO_MATRIX/UNIT_MATRIX/ONES_MATRIX)

## Returns

Filled matrix

6.35.2.123 **INT\_VECSTACK** `mat_find ( MATRIX A, int rel_type, mtype x )`

6.35.2.124 **INT\_VECTOR** `mat_find_within_dist ( MATRIX A, MATRIX d, mtype range )`

Finds points within a neighborhood.

## Parameters

<i>in</i>	<i>A</i>	Points matrix (d x N)
<i>in</i>	<i>d</i>	Matrix point from which the distance to be computed (d x 1)
<i>in</i>	<i>range</i>	Radius to search within

## Returns

Indices Vector

6.35.2.125 **MATRIX** `mat_fliplr ( MATRIX A, MATRIX result )`

6.35.2.126 **MATRIX** `mat_flipud ( MATRIX A, MATRIX result )`

6.35.2.127 **void** `mat_fnextline ( MAT_FILEPOINTER fp )`

Prints nextline to file.

## Parameters

<i>in</i>	<i>fp</i>	Pointer to opened file
-----------	-----------	------------------------

6.35.2.128 **int** `mat_free ( MATRIX A )`

Frees a matrix.

## Parameters

<i>in</i>	<i>A</i>	Input matrix
-----------	----------	--------------

## Returns

Success

6.35.2.129 **MATRIX** `mat_get_sub_matrix_from_cols ( MATRIX A, INT_VECTOR indices, MATRIX result )`

Extracts sub-matrix from columns of a matrix.

## Parameters

<i>in</i>	<i>A</i>	Input matrix
<i>in</i>	<i>indices</i>	Columns to extract
<i>in</i>	<i>result</i>	Matrix to store the result

## Returns

Extracted matrix

**6.35.2.130** **MATRIX** `mat_get_sub_matrix_from_rows ( MATRIX A, INT_VECTOR indices, MATRIX result )`

Extracts sub-matrix from rows of a matrix.

## Parameters

<i>in</i>	<i>A</i>	Input matrix
<i>in</i>	<i>indices</i>	Rows to extract
<i>in</i>	<i>result</i>	Matrix to store the result

## Returns

Extracted matrix

**6.35.2.131** **INT\_VECTOR** `mat_get_sub_vector ( INT_VECTOR a, INT_VECTOR indices )`

Extracts sub-vector from an integer vector.

## Parameters

<i>in</i>	<i>a</i>	Input vector
<i>in</i>	<i>indices</i>	Indices to extracted

## Returns

Extracted vector

**6.35.2.132** **MATRIX** `mat_gfunc ( MATRIX A, mtype(*)(mtype) pt2func, MATRIX result )`

Computes a given function element-wise on a matrix.

## Parameters

<i>in</i>	<i>A</i>	Input matrix
<i>in</i>	<i>f</i>	Given function

## Returns

$\mathbf{B}, b_{ij} = f(a_{ij})$

**6.35.2.133** **int** `mat_go_next_word ( MAT_FILEPOINTER fp )`

Moves to next word in an opened file.

## Parameters

<i>in</i>	<i>fp</i>	Pointer to an opened file
-----------	-----------	---------------------------

## Returns

EOF reached

- 6.35.2.134 void `mat_graph_adjlist` ( `MAT_GRAPH` *g*, int *directed*, int *weighted*, `MAT_FILEPOINTER` *fp* )
- 6.35.2.135 void `mat_graph_adjm_to_adjl` ( `MAT_GRAPH` *g*, `MATRIX` *a* )
- 6.35.2.136 `MAT_GRAPH` `mat_graph_creat` ( void )
- 6.35.2.137 void `mat_graph_dump` ( `MAT_GRAPH` *g*, int *mst* )
- 6.35.2.138 void `mat_graph_dumpf` ( `MAT_GRAPH` *g*, int *mst*, `MAT_FILEPOINTER` *fp* )
- 6.35.2.139 `MAT_GRAPH` `mat_graph_reverse` ( `MAT_GRAPH` *g*, `MAT_GRAPH` *r* )
- 6.35.2.140 `MAT_INT_QUEUE` `mat_graph_search` ( `MAT_GRAPH` *g*, int *connected*, int *mst* )
- 6.35.2.141 void `mat_graph_visit` ( `MAT_GRAPH` *g*, int *k*, int *connected*, int *mst*, `MAT_INT_PRIORITYQUEUE` *pq*, `MAT_INT_QUEUE` *q* )
- 6.35.2.142 `MATRIX` `mat_huber_wt` ( `MATRIX` *A*, mtype *k*, mtype *sigma*, `MATRIX` *result* )

Computes Huber weight function element-wise on a matrix.

## Parameters

<code>in</code>	<i>A</i>	Input matrix
<code>in</code>	<i>k</i>	Huber parameter

## Returns

$\mathbf{B}$ ,  $b_{ij} = f_k(a_{ij})$  where  $f_k$  is the Huber weight function

- 6.35.2.143 mtype `mat_innerprod` ( `MATRIX` *A*, `MATRIX` *B* )
- 6.35.2.144 `MAT_INT_PRIORITYQUEUE` `mat_int_priorityqueue_creat` ( int *type* )
- 6.35.2.145 `mat_intpqnode` `mat_int_priorityqueue_dequeue` ( `MAT_INT_PRIORITYQUEUE` *H* )
- 6.35.2.146 void `mat_int_priorityqueue_enqueue` ( `MAT_INT_PRIORITYQUEUE` *H*, int *data*, int *priority* )
- 6.35.2.147 int `mat_int_priorityqueue_free` ( `MAT_INT_PRIORITYQUEUE` *H* )
- 6.35.2.148 int `mat_int_priorityqueue_is_empty` ( `MAT_INT_PRIORITYQUEUE` *H* )
- 6.35.2.149 int `mat_int_priorityqueue_update` ( `MAT_INT_PRIORITYQUEUE` *H*, int *data*, int *priority*, int *type* )
- 6.35.2.150 mtype `mat_int_qadtrat` ( mtype(\*)mtype *func*, mtype *lower*, mtype *upper* )

Computes Gauss quadrature integration.

## Parameters

in	<i>func</i>	Function $f(\cdot)$ to integrate
in	<i>n</i>	Number of subdivisions
in	<i>lower</i>	Lower Limit
in	<i>upper</i>	Upper Limit

## Returns

$$\int_a^b f(x) dx$$

6.35.2.151 **MAT\_INT\_QUEUE** mat\_int\_queue\_creat ( void )

6.35.2.152 int mat\_int\_queue\_dequeue ( MAT\_INT\_QUEUE s )

6.35.2.153 void mat\_int\_queue\_enqueue ( MAT\_INT\_QUEUE s, int value )

6.35.2.154 int mat\_int\_queue\_free ( MAT\_INT\_QUEUE s )

6.35.2.155 int mat\_int\_queue\_is\_empty ( MAT\_INT\_QUEUE s )

6.35.2.156 mtype mat\_int\_simpson ( mtype(\*) (mtype) func, int n, mtype lower, mtype upper )

Computes Simpson's integration.

## Parameters

in	<i>func</i>	Function $f(\cdot)$ to integrate
in	<i>n</i>	Number of subdivisions
in	<i>lower</i>	Lower Limit
in	<i>upper</i>	Upper Limit

## Returns

$$\int_a^b f(x) dx$$

6.35.2.157 **MAT\_INT\_STACK** mat\_int\_stack\_creat ( void )

6.35.2.158 int mat\_int\_stack\_free ( MAT\_INT\_STACK s )

6.35.2.159 int mat\_int\_stack\_is\_empty ( MAT\_INT\_STACK s )

6.35.2.160 int mat\_int\_stack\_pop ( MAT\_INT\_STACK s )

6.35.2.161 void mat\_int\_stack\_push ( MAT\_INT\_STACK s, int value )

6.35.2.162 mtype mat\_int\_trapezoid ( mtype(\*) (mtype) func, int n, mtype lower, mtype upper )

Computes trapezoid integration.

## Parameters

in	<i>func</i>	Function $f(\cdot)$ to integrate
in	<i>n</i>	Number of subdivisions
in	<i>lower</i>	Lower Limit
in	<i>upper</i>	Upper Limit

## Returns

$$\int_a^b f(x) dx$$

6.35.2.163 **MATRIX** mat\_inv ( **MATRIX** *A*, **MATRIX** *result* )

Computes the inverse of a matrix.

## Parameters

in	<i>A</i>	Input matrix
in	<i>result</i>	Matrix to store the result

## Returns

$$A^{-1}$$

6.35.2.164 **int** mat\_isnumeric ( **MAT\_FILEPOINTER** *fp* )

Checks if current word in an opened file is numeric or not.

## Parameters

in	<i>fp</i>	Pointer to an opened file
----	-----------	---------------------------

## Returns

Zero/non-zero

6.35.2.165 **int** mat\_kdtree\_free ( **MAT\_KDTREE** *t* )

Frees a k-d tree.

## Parameters

in	<i>t</i>	Input k-d tree
----	----------	----------------

## Returns

Success

6.35.2.166 **MATRIX** mat\_kdtree\_k\_nearest ( **MAT\_KDTREE** *t*, **MATRIX** *A*, **int** *k*, **MATRIX** *result* )

Computes k nearest neighbors.

## Parameters

in	<i>t</i>	Input k-d tree
in	<i>A</i>	Input data matrix of size $d \times N$
in	<i>k</i>	Number of neighbors
in	<i>result</i>	Matrix to store the result



## Returns

Output matrix  $B$  with index  $B[0][j]$  and squared distance  $B[1][j]$  for  $j = 1, 2, \dots, N$

6.35.2.167 **MAT\_KDTREE** `mat.kdtree_make_tree ( MATRIX  $A$ , MAT_KDTREE  $result$  )`

Creates a k-d tree from a data matrix.

## Parameters

in	$A$	Input data matrix of size $d \times N$
in	$result$	K-d tree to store the result

## Returns

Output k-d tree

6.35.2.168 **MATRIX** `mat.kdtree_nearest ( MAT_KDTREE  $t$ , MATRIX  $A$ , MATRIX  $result$  )`

Computes nearest neighbors.

## Parameters

in	$t$	Input k-d tree
in	$A$	Input data matrix of size $d \times N$
in	$result$	Matrix to store the result

## Returns

Output matrix  $B$  with index  $B[0][j]$  and squared distance  $B[1][j]$  for  $j = 1, 2, \dots, N$

6.35.2.169 **MATVEC\_DPOINTER** `mat.kmeans ( MATRIX  $data$ , int  $k$ , int  $iters$ , MATVEC_DPOINTER  $result$  )`6.35.2.170 **MATRIX** `mat.least_squares ( MATRIX  $A$ , MATRIX  $Y$ , MATRIX  $result$  )`6.35.2.171 **MATRIX** `mat.legendre ( int  $n$  )`

Computes the  $n^{th}$  Legendre polynomial.

## Parameters

in	$n$	Polynomial series index
----	-----	-------------------------

## Returns

Output polynomial matrix

6.35.2.172 **void** `mat.legendre_init ( )`

Initializes the Legendre polynomial series.

6.35.2.173 **MATRIX** `mat_linear_ls_fit ( MATRIX A, MATRIX Y, int deg, MATRIX result )`

Polynomial model using least squares.

#### Parameters

<code>in</code>	<i>A</i>	Data matrix $N \times 1$
<code>in</code>	<i>Y</i>	Observation matrix $N \times 1$

#### Returns

6.35.2.174 **MATRIX** `mat_ksolve ( MATRIX A, MATRIX b, MATRIX result )`

6.35.2.175 **MATRIX** `mat_ksolve_durbin ( MATRIX A, MATRIX B, MATRIX result )`

Runs Levinson-Durbin algorithm.

#### Parameters

<code>in</code>	<i>A</i>	Input correlation matrix $A = \begin{bmatrix} r_0 & r_1 & \cdots & r_{n-1} \\ r_1 & r_0 & \cdots & r_{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ r_{n-1} & r_{n-2} & \cdots & r_0 \end{bmatrix}$
<code>in</code>	<i>B</i>	Input correlation matrix $B = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \end{bmatrix}$
<code>in</code>	<i>result</i>	Matrix to store the result

#### Returns

$X$  where  $RX = B$

6.35.2.176 **int** `mat_lu ( MATRIX A, MATRIX P )`

6.35.2.177 **MATVEC\_DPOINTER** `mat_max ( MATRIX A, int dim )`

6.35.2.178 **MATRIX** `mat_mds ( MATRIX d, int dims, int type, MATRIX result )`

6.35.2.179 **mtyp** `mat_mean ( MATRIX A )`

6.35.2.180 **MATRIX** `mat_mean_col ( MATRIX A, MATRIX result )`

6.35.2.181 **MATRIX** `mat_mean_row ( MATRIX A, MATRIX result )`

6.35.2.182 **mtyp** `mat_median ( MATRIX A )`

Computes the median of elements of a given matrix.

#### Parameters

<code>in</code>	<i>A</i>	Input matrix
-----------------	----------	--------------

## Returns

$$\text{median}(\{a_{ij}\})$$

6.35.2.183 **MATVEC\_DPOINTER** `mat_min ( MATRIX A, int dim )`

6.35.2.184 `mtype mat_minor ( MATRIX A, int i, int j )`

Computes a minor of a matrix.

## Parameters

<code>in</code>	<i>A</i>	Input matrix
<code>in</code>	<i>i</i>	Row index
<code>in</code>	<i>j</i>	Column index

## Returns

Minor  $M_{ij}$

6.35.2.185 **MAT\_MTYPE\_PRIORITYQUEUE** `mat_mtype_priorityqueue_creat ( int type )`

6.35.2.186 `mat_mtypepqnode mat_mtype_priorityqueue_dequeue ( MAT_MTYPE_PRIORITYQUEUE H )`

6.35.2.187 `void mat_mtype_priorityqueue_enqueue ( MAT_MTYPE_PRIORITYQUEUE H, mtype data, mtype priority )`

6.35.2.188 `int mat_mtype_priorityqueue_free ( MAT_MTYPE_PRIORITYQUEUE H )`

6.35.2.189 `int mat_mtype_priorityqueue_is_empty ( MAT_MTYPE_PRIORITYQUEUE H )`

6.35.2.190 `int mat_mtype_priorityqueue_update ( MAT_MTYPE_PRIORITYQUEUE H, mtype data, mtype priority, int type )`

6.35.2.191 **MAT\_MTYPE\_QUEUE** `mat_mtype_queue_creat ( void )`

6.35.2.192 `mtype mat_mtype_queue_dequeue ( MAT_MTYPE_QUEUE s )`

6.35.2.193 `void mat_mtype_queue_enqueue ( MAT_MTYPE_QUEUE s, mtype value )`

6.35.2.194 `int mat_mtype_queue_free ( MAT_MTYPE_QUEUE s )`

6.35.2.195 `int mat_mtype_queue_is_empty ( MAT_MTYPE_QUEUE s )`

6.35.2.196 **MAT\_MTYPE\_STACK** `mat_mtype_stack_creat ( void )`

6.35.2.197 `int mat_mtype_stack_free ( MAT_MTYPE_STACK s )`

6.35.2.198 `int mat_mtype_stack_is_empty ( MAT_MTYPE_STACK s )`

6.35.2.199 `mtype mat_mtype_stack_pop ( MAT_MTYPE_STACK s )`

6.35.2.200 `void mat_mtype_stack_push ( MAT_MTYPE_STACK s, mtype value )`

6.35.2.201 **MATRIX** `mat_mul ( MATRIX A, MATRIX B, MATRIX result )`

6.35.2.202 **MATRIX** `mat_mul_dot ( MATRIX A, MATRIX B, MATRIX result )`

6.35.2.203 **MATRIX** `mat_mul_fast ( MATRIX A, MATRIX B, MATRIX result )`

6.35.2.204 **MATRIX** `mat_muls ( MATRIX A, mtype s, MATRIX result )`

6.35.2.205 `void mat_nextline ( void )`

Prints nextline to stdout.

6.35.2.206 `mtype mat_norm_inf ( MATRIX A )`

6.35.2.207 `mtype mat_norm_one ( MATRIX A )`

6.35.2.208 `mtype mat_norm_p ( MATRIX A, mtype p )`

6.35.2.209 **MATSTACK** `mat_omp ( MATRIX A, MATRIX b, int k, mtype tol, MATSTACK result )`

6.35.2.210 `mtype mat_order_statistic ( MATRIX A, int k )`

Computes the  $k^{th}$  order statistic of elements of a given matrix.

#### Parameters

<code>in</code>	<code>A</code>	Input matrix
<code>in</code>	<code>k</code>	Order

#### Returns

$$O_k(\{a_{ij}\})$$

6.35.2.211 **MATSTACK** `mat_pca ( MATRIX data, int pca_type )`

6.35.2.212 **MAT\_PERCEPTRON** `mat_perceptron_creat ( void )`

Creates a perceptron.

#### Returns

Output perceptron

6.35.2.213 `int mat_perceptron_free ( MAT_PERCEPTRON a )`

Frees a perceptron.

#### Parameters

<code>in</code>	<code>a</code>	Input perceptron
-----------------	----------------	------------------

#### Returns

Success

6.35.2.214 **INT\_VECTOR** `mat_perceptron_test ( MATRIX data, MAT_PERCEPTRON p_model )`

6.35.2.215 **MAT\_PERCEPTRON** `mat_perceptron_train ( MATRIX data, INT_VECTOR labels, int num_of_iterations )`

6.35.2.216 **MAT\_PERCEPTRON** `mat_perceptron_train_ ( MATRIX data1, MATRIX data2, MAT_PERCEPTRON p_model, int class_num )`

6.35.2.217 **MATRIX** `mat_pick_col ( MATRIX A, int c, MATRIX result )`

Picks a column from a matrix.

#### Parameters

<i>in</i>	<i>A</i>	Input matrix
<i>in</i>	<i>r</i>	Column index
<i>in</i>	<i>result</i>	Matrix to store the result

#### Returns

Column matrix

6.35.2.218 **MATRIX** `mat_pick_row ( MATRIX A, int r, MATRIX result )`

Picks a row from a matrix.

#### Parameters

<i>in</i>	<i>A</i>	Input matrix
<i>in</i>	<i>r</i>	Row index
<i>in</i>	<i>result</i>	Matrix to store the result

#### Returns

Row matrix

6.35.2.219 **MATRIX** `mat_pinv ( MATRIX A, MATRIX result )`

Computes pseudo-inverse of a matrix.

#### Parameters

<i>in</i>	<i>A</i>	Input matrix
<i>in</i>	<i>result</i>	Matrix to store the result

#### Returns

$$(A^T A)^{-1} A^T$$

6.35.2.220 **MATRIX** `mat_pol2cart ( MATRIX A, int dim, MATRIX result )`

Converts polar co-ordinates to Cartesian co-ordinates.

#### Parameters

<i>in</i>	<i>A</i>	Input matrix
<i>in</i>	<i>dim</i>	Data order ROWS/COLS

**Returns**

Cartesian co-ordinate matrix

**6.35.2.221 MATRIX mat\_poly\_add ( MATRIX A, MATRIX B, MATRIX result )**

Adds two polynomials.

**Parameters**

<i>in</i>	<i>A</i>	First input polynomial matrix
<i>in</i>	<i>B</i>	Second input polynomial matrix
<i>in</i>	<i>result</i>	Matrix to store the result

**Returns**

Output matrix

**6.35.2.222 MATRIX mat\_poly\_diff ( MATRIX A, int dir, MATRIX result )**

Computes derivative polynomial of a polynomial.

**Parameters**

<i>in</i>	<i>A</i>	Input polynomial matrix
<i>in</i>	<i>dir</i>	Direction (ROWS/COLS)
<i>in</i>	<i>result</i>	Matrix to store the result

**Returns**

Output matrix

**6.35.2.223 MATRIX mat\_poly\_diff\_eval ( MATRIX A, mtype x, int dir, MATRIX result )**

Evaluates derivative polynomial at a point.

**Parameters**

<i>in</i>	<i>A</i>	Input polynomial matrix
<i>in</i>	<i>x</i>	Value at which to evaluate the derivative
<i>in</i>	<i>dir</i>	Direction (ROWS/COLS)
<i>in</i>	<i>result</i>	Matrix to store the result

**Returns**

Output matrix

**6.35.2.224 MATSTACK mat\_poly\_div ( MATRIX A, MATRIX B, MATSTACK result )**

Divides two polynomials.

## Parameters

in	<i>A</i>	First input polynomial matrix
in	<i>B</i>	Second input polynomial matrix
in	<i>result</i>	Matrix to store the result

## Returns

Output matrix

6.35.2.225 **MATRIX** `mat_poly_eval ( MATRIX A, mtype x, int dir, MATRIX result )`

Evaluates polynomial at a point.

## Parameters

in	<i>A</i>	Input polynomial matrix
in	<i>x</i>	Value at which to evaluate
in	<i>dir</i>	Direction (ROWS/COLS)
in	<i>result</i>	Matrix to store the result

## Returns

Output matrix

6.35.2.226 **MATRIX** `mat_poly_mul ( MATRIX A, MATRIX B, MATRIX result )`

Multiplies two polynomials.

## Parameters

in	<i>a</i>	First input polynomial matrix
in	<i>b</i>	Second input polynomial matrix
in	<i>result</i>	Matrix to store the result

## Returns

Output matrix

6.35.2.227 **MATRIX** `mat_poly_scale ( MATRIX A, mtype s, MATRIX result )`

Multiplies a polynomial with a scalar.

## Parameters

in	<i>A</i>	Input polynomial matrix
in	<i>s</i>	Scalar
in	<i>result</i>	Matrix to store the result

## Returns

Output matrix

**6.35.2.228 MATRIX mat\_poly\_shift ( MATRIX *A*, int *s*, MATRIX *result* )**

Shifts a polynomial.

**Parameters**

in	<i>A</i>	Input polynomial matrix
in	<i>s</i>	Scalar shift
in	<i>result</i>	Matrix to store the result

**Returns**

Output matrix

**6.35.2.229 MATSTACK mat\_qr ( MATRIX *A*, MATSTACK *qr* )**

Computes QR decomposition.

**Parameters**

in	<i>A</i>	Input matrix
in	<i>qr</i>	Matrix stack to store result

**Returns**

Output QR Matrix stack

**6.35.2.230 MATSTACK mat\_qsort ( MATRIX *A*, int *dim*, MATSTACK *result* )**

Sorts elements of a given matrix.

**Parameters**

in	<i>A</i>	Input matrix
in	<i>dim</i>	Direction of sort (ROWS/COLS)
out	<i>result</i>	Output matrix stack

**Returns**

Output matrix stack of sorted *A* and their positions

**6.35.2.231 MATRIX mat\_rand ( int *r*, int *c*, MATRIX *result* )****6.35.2.232 MATRIX mat\_randexp ( int *r*, int *c*, mtype *mu*, MATRIX *result* )****6.35.2.233 MATRIX mat\_randfun ( int *r*, int *c*, mtype(\*) (mtype) *fun*, mtype *xmin*, mtype *xmax*, MATRIX *result* )****6.35.2.234 MATRIX mat\_randn ( int *r*, int *c*, MATRIX *result* )****6.35.2.235 MATRIX mat\_randperm ( int *m*, int *n*, MATRIX *result* )****6.35.2.236 MATRIX mat\_randperm.n ( int *n*, MATRIX *result* )**



6.35.2.237 `int mat_read_word ( MAT_FILEPOINTER fp, char * c_word )`

Reads current word from an opened file.

#### Parameters

<i>in</i>	<i>fp</i>	Pointer to an opened file
<i>out</i>	<i>c_word</i>	Pointer to word read

#### Returns

EOF reached

6.35.2.238 `MATRIX mat_reg_inv ( MATRIX A, mtype r, MATRIX result )`

Computes the regularized inverse of a matrix.

#### Parameters

<i>in</i>	<i>A</i>	Input matrix
<i>in</i>	<i>r</i>	Regularizing constant
<i>in</i>	<i>result</i>	Matrix to store the result

#### Returns

$$(A + rI)^{-1}$$

6.35.2.239 `MATRIX mat_rob_least_squares ( MATRIX A, MATRIX Y, int lossfunc, MATRIX result )`

6.35.2.240 `MATRIX mat_robust_fit ( MATRIX A, MATRIX Y, int deg, int lossfunc, MATRIX result )`

6.35.2.241 `MATRIX mat_rowcopy ( MATRIX A, int rowa, int rowb, MATRIX result )`

Copies a row from a matrix.

#### Parameters

<i>in</i>	<i>A</i>	Input matrix
<i>in</i>	<i>rowa</i>	Source row
<i>in</i>	<i>rowb</i>	Destination row
<i>in</i>	<i>result</i>	Matrix to store the result

#### Returns

Copied matrix

6.35.2.242 `MATRIX mat_scpcol ( MATRIX data )`

6.35.2.243 `void mat_set_seed ( int seed )`

6.35.2.244 `MATRIX mat_sub ( MATRIX A, MATRIX B, MATRIX result )`

6.35.2.245 `MATRIX mat_submat ( MATRIX A, int i, int j, MATRIX result )`

Deletes a row and a column of a matrix.

## Parameters

<code>in</code>	<code>A</code>	Input matrix
<code>in</code>	<code>i</code>	Row index
<code>in</code>	<code>j</code>	Column index
<code>in</code>	<code>result</code>	Matrix to store the result

## Returns

Extracted matrix

6.35.2.246 **MATRIX** `mat_subs ( MATRIX A, mtype s, MATRIX result )`

6.35.2.247 `mtype mat_sum ( MATRIX A )`

6.35.2.248 **MATRIX** `mat_sum_col ( MATRIX A, MATRIX result )`

6.35.2.249 **MATRIX** `mat_sum_row ( MATRIX A, MATRIX result )`

6.35.2.250 **MATRIX** `mat_syntoeplz ( MATRIX R, MATRIX result )`

6.35.2.251 `void mat_tic ( void )`

Starts stopwatch timer.

6.35.2.252 `double mat_toc ( void )`

Computes elapsed time from last start of timer.

## Returns

Elapsed time

6.35.2.253 `void mat_toc_print ( void )`

Computes and prints elapsed time from last start of timer on the stdout.

6.35.2.254 `void mat_tqli ( MATRIX d, MATRIX e, MATRIX z )`

6.35.2.255 **MATRIX** `mat_tran ( MATRIX A, MATRIX result )`

Computes the transpose of a matrix.

## Parameters

<code>in</code>	<code>A</code>	Input matrix
-----------------	----------------	--------------

## Returns

$A^T$

6.35.2.256 `void mat_tred2 ( MATRIX a, MATRIX d, MATRIX e )`

**6.35.2.257 MATRIX mat\_vectorize ( MATRIX *A*, MATRIX *result* )**

Reshapes a matrix to a vector.

**Parameters**

in	<i>A</i>	Input matrix
in	<i>result</i>	Matrix to store the result

**Returns**

$vec(\mathbf{A})$

**6.35.2.258 MATRIX mat\_vectorize\_tr ( MATRIX *A*, MATRIX *result* )**

Reshapes transpose of a matrix to a vector.

**Parameters**

in	<i>A</i>	Input matrix
in	<i>result</i>	Matrix to store the result

**Returns**

$vec(\mathbf{A}^T)$

**6.35.2.259 MATRIX mat\_w\_least\_squares ( MATRIX *A*, MATRIX *Y*, MATRIX *w*, MATRIX *result* )****6.35.2.260 MATRIX mat\_wpinv ( MATRIX *A*, MATRIX *w*, MATRIX *result* )**

Computes weighted pseudo-inverse of a matrix.

**Parameters**

in	<i>A</i>	Input matrix
in	<i>w</i>	Weight matrix
in	<i>result</i>	Matrix to store the result

**Returns**

$(A^T W A)^{-1} A^T W$

**6.35.2.261 MATRIX mat\_xcopy ( MATRIX *A*, int *si*, int *ei*, int *sj*, int *ej*, MATRIX *result* )**

Copies a sub-matrix.

**Parameters**

in	<i>A</i>	Input matrix
in	<i>si</i>	Start of first index, $s_i$
in	<i>ei</i>	End of first index, $e_i$
in	<i>sj</i>	Start of second index, $s_j$
in	<i>ej</i>	End of second index, $e_j$
in	<i>result</i>	Matrix to store the result

## Returns

Extracted matrix  $A_{s_i:e_i,s_j:e_j}$

### 6.35.2.262 MATRIX mat\_xjoin ( MATRIX A11, MATRIX A12, MATRIX A21, MATRIX A22, MATRIX result )

Copies a sub-matrix.

## Parameters

in	A11	Input matrix, $A_{11}$
in	A12	Input matrix, $A_{12}$
in	A21	Input matrix, $A_{21}$
in	A22	Input matrix, $A_{22}$
in	result	Matrix to store the result

## Returns

Block matrix  $\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$

### 6.35.2.263 int mats\_isinf ( mtype x )

Checks if scalar is infinite.

## Parameters

in	x	Input scalar
----	---	--------------

## Returns

Zero/non-zero

### 6.35.2.264 int mats\_isnan ( mtype x )

Checks if scalar is NaN.

## Parameters

in	x	Input scalar
----	---	--------------

## Returns

Zero/non-zero

### 6.35.2.265 MATSTACK matstack\_append ( MATSTACK s, MATRIX A )

Appends a matrix to a matrix stack.

## Parameters

in	s	Input matrix stack
in	A	Input matrix to append

## Returns

Output matrix stack

6.35.2.266 **MATSTACK** `matstack_creat ( int len )`

Creates a matrix stack.

## Parameters

<i>in</i>	<i>len</i>	Length of the stack
-----------	------------	---------------------

## Returns

Output matrix stack

6.35.2.267 **MATSTACK** `matstack_error ( int err_ )`

Generates error message for matrix stack errors and exits.

## Parameters

<i>in</i>	<i>err</i>	Error type (MATSTACK_MALLOC/MATSTACK_FNOTOPEN/MATSTACK_FNOTGETMAT/MATSTACK_SIZEMISMATCH/ MATSTACK_INVERSE_ERROR)
-----------	------------	--

6.35.2.268 `int` `matstack_free ( MATSTACK A )`

Frees a matrix stack.

## Parameters

<i>in</i>	<i>A</i>	Input matrix stack
-----------	----------	--------------------

## Returns

Success

6.35.2.269 **MATVEC\_DPOINTER** `matvec_creat ( void )`

Creates a matrix-vector pair.

## Returns

Output matrix-vector pair

6.35.2.270 `int` `matvec_free ( MATVEC_DPOINTER a )`

Frees a matrix-vector pair.

## Parameters

<i>in</i>	<i>a</i>	Input matrix-vector pair
-----------	----------	--------------------------

## Returns

Success

6.35.2.271 `int pq_error ( int err_ )`

Generates error message for priority queue errors and exits.

## Parameters

<i>in</i>	<i>err</i>	Error type (PQ_MALLOC/PQ_EMPTY)
-----------	------------	---------------------------------

6.35.2.272 `int queue_error ( int err_ )`

Generates error message for queue errors and exits.

## Parameters

<i>in</i>	<i>err</i>	Error type (QUEUE_MALLOC/QUEUE_EMPTY)
-----------	------------	---------------------------------------

6.35.2.273 `int stack_error ( int err_ )`

Generates error message for stack errors and exits.

## Parameters

<i>in</i>	<i>err</i>	Error type (STACK_MALLOC/STACK_EMPTY)
-----------	------------	---------------------------------------

## 6.35.3 Variable Documentation

6.35.3.1 **MATSTACK** `mat_binom_series_table`

6.35.3.2 **MATSTACK** `mat_cheby_series_table`

6.35.3.3 `clock_t` **MAT\_CLOCK\_TIME**

6.35.3.4 **MATSTACK** `mat_legendre_series_table`

6.35.3.5 `unsigned int` **MAT\_SEED**

6.35.3.6 `int` **MAT\_SET\_SEED**

## 6.36 matsearch.c File Reference

### Functions

- [INT\\_VECTOR](#) `int_vec_find` ([INT\\_VECTOR](#) *a*, `int` *rel\_type*, `int` *n*)
- [INT\\_VECSTACK](#) `mat_find` ([MATRIX](#) *A*, `int` *rel\_type*, `mtype` *x*)

### 6.36.1 Function Documentation

6.36.1.1 `INT_VECTOR` `int_vec_find` ( `INT_VECTOR` *a*, `int` *rel\_type*, `int` *n* )

6.36.1.2 `INT_VECSTACK mat_find ( MATRIX A, int rel_type, mtype x )`

## 6.37 matsolve.c File Reference

### Functions

- `int mat_lu (MATRIX A, MATRIX P)`
- `void mat_backsubs1 (MATRIX A, MATRIX B, MATRIX X, MATRIX P, int xcol)`
- `MATRIX mat_lsolve (MATRIX A, MATRIX b, MATRIX result)`
- `MATRIX mat_cholesky (MATRIX A, MATRIX result)`  
*Computes Cholesky factor of a matrix.*
- `MATRIX mat_conjgrad (MATRIX A, MATRIX b, MATRIX x0, mtype tol, int miters, MATRIX result)`  
*Solves a linear system with conjugate gradients method.*

### 6.37.1 Function Documentation

6.37.1.1 `void mat_backsubs1 ( MATRIX A, MATRIX B, MATRIX X, MATRIX P, int xcol )`

6.37.1.2 `MATRIX mat_cholesky ( MATRIX A, MATRIX result )`

Computes Cholesky factor of a matrix.

#### Parameters

<code>in</code>	<code>A</code>	Input matrix
<code>in</code>	<code>result</code>	Matrix to store the result

#### Returns

Cholesky factor

6.37.1.3 `MATRIX mat_conjgrad ( MATRIX A, MATRIX b, MATRIX x0, mtype tol, int miters, MATRIX result )`

Solves a linear system with conjugate gradients method.

#### Parameters

<code>in</code>	<code>A</code>	Input matrix
<code>in</code>	<code>b</code>	Observed matrix
<code>in</code>	<code>result</code>	Matrix to store the result

#### Returns

`x`

6.37.1.4 `MATRIX mat_lsolve ( MATRIX A, MATRIX b, MATRIX result )`

6.37.1.5 `int mat_lu ( MATRIX A, MATRIX P )`

## 6.38 matsort.c File Reference

## Functions

- mtype `mat_median` (**MATRIX** A)  
*Computes the median of elements of a given matrix.*
- mtype `mat_order_statistic` (**MATRIX** A, int k)  
*Computes the  $k^{th}$  order statistic of elements of a given matrix.*
- **MATSTACK** `mat_qsort` (**MATRIX** A, int dim, **MATSTACK** result)  
*Sorts elements of a given matrix.*

### 6.38.1 Function Documentation

#### 6.38.1.1 mtype `mat_median` ( **MATRIX** A )

Computes the median of elements of a given matrix.

##### Parameters

in	A	Input matrix
----	---	--------------

##### Returns

$\text{median}(\{a_{ij}\})$

#### 6.38.1.2 mtype `mat_order_statistic` ( **MATRIX** A, int k )

Computes the  $k^{th}$  order statistic of elements of a given matrix.

##### Parameters

in	A	Input matrix
in	k	Order

##### Returns

$O_k(\{a_{ij}\})$

#### 6.38.1.3 **MATSTACK** `mat_qsort` ( **MATRIX** A, int dim, **MATSTACK** result )

Sorts elements of a given matrix.

##### Parameters

in	A	Input matrix
in	dim	Direction of sort (ROWS/COLS)
out	result	Output matrix stack

##### Returns

Output matrix stack of sorted A and their positions

## 6.39 matstdrels.c File Reference



## Functions

- int [gen\\_gt](#) (mtype a)
- int [gen\\_lt](#) (mtype a)
- int [gen\\_eq](#) (mtype a)
- mtype [gen\\_abs\\_ceil](#) (mtype a)

### 6.39.1 Function Documentation

6.39.1.1 mtype [gen\\_abs\\_ceil](#) ( mtype a )

6.39.1.2 int [gen\\_eq](#) ( mtype a )

6.39.1.3 int [gen\\_gt](#) ( mtype a )

6.39.1.4 int [gen\\_lt](#) ( mtype a )

## 6.40 matsub.c File Reference

## Functions

- [MATRIX mat\\_sub](#) ([MATRIX A](#), [MATRIX B](#), [MATRIX result](#))
- [MATRIX mat\\_subs](#) ([MATRIX A](#), mtype s, [MATRIX result](#))
- [INT\\_VECTOR int\\_vec\\_sub](#) ([INT\\_VECTOR A](#), [INT\\_VECTOR B](#), [INT\\_VECTOR result](#))
- [INT\\_VECTOR int\\_vec\\_subs](#) ([INT\\_VECTOR A](#), int x, [INT\\_VECTOR result](#))

### 6.40.1 Function Documentation

6.40.1.1 [INT\\_VECTOR int\\_vec\\_sub](#) ( [INT\\_VECTOR A](#), [INT\\_VECTOR B](#), [INT\\_VECTOR result](#) )

6.40.1.2 [INT\\_VECTOR int\\_vec\\_subs](#) ( [INT\\_VECTOR A](#), int x, [INT\\_VECTOR result](#) )

6.40.1.3 [MATRIX mat\\_sub](#) ( [MATRIX A](#), [MATRIX B](#), [MATRIX result](#) )

6.40.1.4 [MATRIX mat\\_subs](#) ( [MATRIX A](#), mtype s, [MATRIX result](#) )

## 6.41 matsubx.c File Reference

## Functions

- [MATRIX mat\\_submat](#) ([MATRIX A](#), int i, int j, [MATRIX result](#))  
*Deletes a row and a column of a matrix.*

### 6.41.1 Function Documentation

6.41.1.1 [MATRIX mat\\_submat](#) ( [MATRIX A](#), int i, int j, [MATRIX result](#) )

Deletes a row and a column of a matrix.

## Parameters

in	<i>A</i>	Input matrix
in	<i>i</i>	Row index
in	<i>j</i>	Column index
in	<i>result</i>	Matrix to store the result

## Returns

Extracted matrix

## 6.42 matsum.c File Reference

### Functions

- mtype [mat\\_sum](#) ([MATRIX A](#))
- [MATRIX mat\\_sum\\_row](#) ([MATRIX A](#), [MATRIX result](#))
- [MATRIX mat\\_sum\\_col](#) ([MATRIX A](#), [MATRIX result](#))

#### 6.42.1 Function Documentation

6.42.1.1 mtype [mat\\_sum](#) ( [MATRIX A](#) )

6.42.1.2 [MATRIX mat\\_sum\\_col](#) ( [MATRIX A](#), [MATRIX result](#) )

6.42.1.3 [MATRIX mat\\_sum\\_row](#) ( [MATRIX A](#), [MATRIX result](#) )

## 6.43 mattext.c File Reference

### Functions

- int [mat\\_isnumeric](#) ([MAT\\_FILEPOINTER fp](#))  
*Checks if current word in an opened file is numeric or not.*
- int [mat\\_go\\_next\\_word](#) ([MAT\\_FILEPOINTER fp](#))  
*Moves to next word in an opened file.*
- int [mat\\_count\\_words\\_in\\_line](#) ([MAT\\_FILEPOINTER fp](#), int \*count)  
*Count words in current line in an opened file.*
- [MATRIX mat\\_dlmread](#) (const char \*fname)  
*Reads a matrix from a file.*
- int [mat\\_read\\_word](#) ([MAT\\_FILEPOINTER fp](#), char \*c\_word)  
*Reads current word from an opened file.*
- void [mat\\_dlmwrite](#) (const char \*fname, [MATRIX A](#))  
*Writes a matrix to a file.*

#### 6.43.1 Function Documentation

6.43.1.1 int [mat\\_count\\_words\\_in\\_line](#) ( [MAT\\_FILEPOINTER fp](#), int \* *count* )

Count words in current line in an opened file.

## Parameters

in	<i>fp</i>	Pointer to an opened file
out	<i>count</i>	Pointer to output count

## Returns

EOF reached

6.43.1.2 **MATRIX** mat\_dlmread ( const char \* *fname* )

Reads a matrix from a file.

## Parameters

in	<i>fname</i>	Filename to read from
----	--------------	-----------------------

## Returns

Output matrix

6.43.1.3 void mat\_dlmwrite ( const char \* *fname*, **MATRIX** *A* )

Writes a matrix to a file.

## Parameters

in	<i>fname</i>	Filename to write into
in	<i>A</i>	Input matrix

6.43.1.4 int mat\_go\_next\_word ( **MAT\_FILEPOINTER** *fp* )

Moves to next word in an opened file.

## Parameters

in	<i>fp</i>	Pointer to an opened file
----	-----------	---------------------------

## Returns

EOF reached

6.43.1.5 int mat\_isnumeric ( **MAT\_FILEPOINTER** *fp* )

Checks if current word in an opened file is numeric or not.

## Parameters

in	<i>fp</i>	Pointer to an opened file
----	-----------	---------------------------

## Returns

Zero/non-zero

6.43.1.6 int mat\_read\_word ( **MAT\_FILEPOINTER** *fp*, char \* *c\_word* )

Reads current word from an opened file.

## Parameters

in	<i>fp</i>	Pointer to an opened file
out	<i>c_word</i>	Pointer to word read

## Returns

EOF reached

## 6.44 mattimers.c File Reference

### Functions

- void [mat\\_tic](#) (void)  
*Starts stopwatch timer.*
- double [mat\\_toc](#) (void)  
*Computes elapsed time from last start of timer.*
- void [mat\\_toc\\_print](#) (void)  
*Computes and prints elapsed time from last start of timer on the stdout.*

### Variables

- clock\_t [MAT\\_CLOCK\\_TIME](#)

#### 6.44.1 Function Documentation

##### 6.44.1.1 void mat\_tic ( void )

Starts stopwatch timer.

##### 6.44.1.2 double mat\_toc ( void )

Computes elapsed time from last start of timer.

##### Returns

Elapsed time

##### 6.44.1.3 void mat\_toc\_print ( void )

Computes and prints elapsed time from last start of timer on the stdout.

#### 6.44.2 Variable Documentation

##### 6.44.2.1 clock\_t MAT\_CLOCK\_TIME

## 6.45 mattoepz.c File Reference

### Functions

- [MATRIX mat\\_symtoeplz](#) ([MATRIX](#) R, [MATRIX](#) result)

### 6.45.1 Function Documentation

6.45.1.1 **MATRIX** `mat_symtoeplz ( MATRIX R, MATRIX result )`

## 6.46 mattran.c File Reference

### Functions

- **MATRIX** `mat_tran (MATRIX A, MATRIX result)`  
*Computes the transpose of a matrix.*

### 6.46.1 Function Documentation

6.46.1.1 **MATRIX** `mat_tran ( MATRIX A, MATRIX result )`

Computes the transpose of a matrix.

#### Parameters

<code>in</code>	<code>A</code>	Input matrix
-----------------	----------------	--------------

#### Returns

$A^T$

## 6.47 README.md File Reference

# Index

- `__int_vec_creat`
  - `matrix.h`, [80](#)
- `__int_vecstack_creat`
  - `matrix.h`, [80](#)
- `__mat_addfunc`
  - `matfuncs.c`, [45](#)
  - `matrix.h`, [80](#)
- `__mat_arccosh`
  - `matfuncs.c`, [45](#)
  - `matrix.h`, [81](#)
- `__mat_arcsinh`
  - `matfuncs.c`, [45](#)
  - `matrix.h`, [81](#)
- `__mat_arctanh`
  - `matfuncs.c`, [46](#)
  - `matrix.h`, [81](#)
- `__mat_bisquare_wt`
  - `matfuncs.c`, [46](#)
  - `matrix.h`, [81](#)
- `__mat_cart2pol`
  - `matmisc.c`, [54](#)
  - `matrix.h`, [81](#)
- `__mat_creat`
  - `matrix.h`, [82](#)
- `__mat_divfunc`
  - `matfuncs.c`, [46](#)
  - `matrix.h`, [82](#)
- `__mat_fft`
  - `matrix.h`, [82](#)
- `__mat_huber_wt`
  - `matfuncs.c`, [46](#)
  - `matrix.h`, [82](#)
- `__mat_kd_find_median`
  - `matrix.h`, [82](#)
- `__mat_kdtree_k_nearest`
  - `matrix.h`, [82](#)
- `__mat_kdtree_make_tree`
  - `matrix.h`, [82](#)
- `__mat_kdtree_nearest`
  - `matrix.h`, [82](#)
- `__mat_lint`
  - `matrix.h`, [82](#)
- `__mat_logplusone`
  - `matfuncs.c`, [47](#)
  - `matrix.h`, [82](#)
- `__mat_mds_metric`
  - `matmds.c`, [53](#)
  - `matrix.h`, [83](#)
- `__mat_mds_nonmetric`
  - `matmds.c`, [53](#)
  - `matrix.h`, [83](#)
- `__mat_mulfunc`
  - `matfuncs.c`, [47](#)
  - `matrix.h`, [83](#)
- `__mat_pol2cart`
  - `matmisc.c`, [54](#)
  - `matrix.h`, [83](#)
- `__mat_powerof2`
  - `matrix.h`, [83](#)
- `__mat_quicksort`
  - `matrix.h`, [83](#)
- `__mat_rand`
  - `matrand.c`, [66](#)
  - `matrix.h`, [83](#)
- `__mat_randexp`
  - `matrand.c`, [66](#)
  - `matrix.h`, [83](#)
- `__mat_randfun`
  - `matrand.c`, [66](#)
  - `matrix.h`, [83](#)
- `__mat_randn`
  - `matrand.c`, [66](#)
  - `matrix.h`, [83](#)
- `__mat_sqrfunc`
  - `matfuncs.c`, [47](#)
  - `matrix.h`, [83](#)
- `__mat_sqrtfunc`
  - `matfuncs.c`, [47](#)
  - `matrix.h`, [83](#)
- `__mat_subfunc`
  - `matfuncs.c`, [47](#)
  - `matrix.h`, [84](#)
- `__matstack_creat`
  - `matrix.h`, [84](#)
- `_is_allocated`
  - `mat_kdtree`, [15](#)
- `adj`
  - `mat_graph`, [11](#)
- `class_covars`
  - `mat_bayes_model`, [9](#)
- `class_labels`
  - `mat_bayes_model`, [9](#)
  - `mat_perceptron`, [18](#)
- `class_means`
  - `mat_bayes_model`, [9](#)
- `class_priors`
  - `mat_bayes_model`, [9](#)

- class\_weights
  - mat\_perceptron, 18
- dad
  - mat\_graph, 11
- data
  - mat\_intpqnode, 14
  - mat\_kdtree, 15
  - mat\_mtypepqnode, 18
  - mat\_qintnode, 19
  - mat\_qmtyphenode, 20
- element
  - mat\_int\_priorityqueue, 12
  - mat\_mtype\_priorityqueue, 15
  - mat\_tree\_node, 20
- gen\_abs\_ceil
  - matrix.h, 84
  - matstdrels.c, 121
- gen\_eq
  - matrix.h, 84
  - matstdrels.c, 121
- gen\_error
  - materr.c, 41
  - matrix.h, 84
- gen\_gt
  - matrix.h, 84
  - matstdrels.c, 121
- gen\_lt
  - matrix.h, 84
  - matstdrels.c, 121
- graph\_error
  - materr.c, 41
  - matrix.h, 84
- head
  - mat\_int\_queue, 13
  - mat\_mtype\_queue, 16
- INT\_VECSTACK
  - matrix.h, 77
- INT\_VECTOR
  - matrix.h, 77
- id
  - mat\_graph, 11
- idx
  - mat\_kdnode, 14
- int\_vec2\_mat
  - matconv.c, 24
  - matrix.h, 84
- int\_vec\_add
  - matadd.c, 22
  - matrix.h, 85
- int\_vec\_adds
  - matadd.c, 22
  - matrix.h, 85
- int\_vec\_append
  - matcreat.c, 26
- matrix.h, 85
- int\_vec\_concat
  - matconcat.c, 23
  - matrix.h, 85
- int\_vec\_copy
  - matcreat.c, 26
  - matrix.h, 86
- int\_vec\_creat
  - matcreat.c, 26
  - matrix.h, 86
- int\_vec\_div
  - matdiv.c, 36
  - matrix.h, 86
- int\_vec\_divs
  - matdiv.c, 37
  - matrix.h, 86
- int\_vec\_dump
  - matdump.c, 38
  - matrix.h, 87
- int\_vec\_dumpf
  - matdump.c, 38
  - matrix.h, 87
- int\_vec\_error
  - materr.c, 41
  - matrix.h, 87
- int\_vec\_fdump
  - matdump.c, 38
  - matrix.h, 87
- int\_vec\_fdumpf
  - matdump.c, 38
  - matrix.h, 87
- int\_vec\_fill
  - matcreat.c, 27
  - matrix.h, 87
- int\_vec\_fill\_type
  - matcreat.c, 27
  - matrix.h, 88
- int\_vec\_find
  - matrix.h, 88
  - matsearch.c, 118
- int\_vec\_free
  - matcreat.c, 27
  - matrix.h, 88
- int\_vec\_mul
  - matmul.c, 58
  - matrix.h, 88
- int\_vec\_muls
  - matmul.c, 58
  - matrix.h, 88
- int\_vec\_permute\_vect
  - matmisc.c, 54
  - matrix.h, 88
- int\_vec\_randperm
  - matrand.c, 66
  - matrix.h, 89
- int\_vec\_sub
  - matrix.h, 89
  - matsub.c, 121

- int\_vec\_subs
  - matrix.h, 89
  - matsub.c, 121
- int\_vec\_unique
  - matmisc.c, 55
  - matrix.h, 89
- int\_vecstack\_creat
  - matcreat.c, 27
  - matrix.h, 89
- int\_vecstack\_error
  - materr.c, 41
  - matrix.h, 89
- int\_vecstack\_free
  - matcreat.c, 28
  - matrix.h, 89
- istrained
  - mat\_perceptron, 18
- kdtree
  - mat\_kdtree, 15
- left
  - mat\_kdnode, 14
  - mat\_tree\_node, 20
- length
  - mat\_int\_priorityqueue, 12
  - mat\_int\_stack, 13
  - mat\_kdtree, 15
  - mat\_mtype\_priorityqueue, 15
  - mat\_mtype\_stack, 17
- MAT\_BAYES\_MODEL
  - matrix.h, 77
- MAT\_CLOCK\_TIME
  - matrix.h, 118
  - mattimers.c, 124
- MAT\_GNODE
  - matrix.h, 77
- MAT\_GRAPH
  - matrix.h, 77
- MAT\_INT\_QUEUE
  - matrix.h, 78
- MAT\_INT\_STACK
  - matrix.h, 78
- MAT\_INTPQNODE
  - matrix.h, 78
- MAT\_KDNODE
  - matrix.h, 78
- MAT\_KDTREE
  - matrix.h, 78
- MAT\_MTYPE\_QUEUE
  - matrix.h, 79
- MAT\_MTYPE\_STACK
  - matrix.h, 79
- MAT\_MTYPEPQNODE
  - matrix.h, 79
- MAT\_PERCEPTRON
  - matrix.h, 79
- MAT\_QINTNODE
  - matrix.h, 79
- MAT\_QMTYPENODE
  - matrix.h, 80
- MAT\_SEED
  - matrand.c, 66
  - matrix.h, 118
- MAT\_SET\_SEED
  - matrand.c, 66
  - matrix.h, 118
- MAT\_TREE
  - matrix.h, 80
- MAT\_TREE\_NODE
  - matrix.h, 80
- MATRIX
  - matrix.h, 80
- MATSTACK
  - matrix.h, 80
- MATVEC\_DPOINTER
  - matrix.h, 80
- mat\_2int\_vec
  - matconv.c, 24
  - matrix.h, 90
- mat\_abs
  - matabs.c, 21
  - matrix.h, 90
- mat\_add
  - matadd.c, 22
  - matrix.h, 90
- mat\_adds
  - matadd.c, 22
  - matrix.h, 90
- mat\_backsubs1
  - matrix.h, 91
  - matsolve.c, 119
- mat\_bayes\_classifier\_test
  - matprec.c, 65
  - matrix.h, 91
- mat\_bayes\_classifier\_train
  - matprec.c, 65
  - matrix.h, 91
- mat\_bayes\_model, 9
  - class\_covars, 9
  - class\_labels, 9
  - class\_means, 9
  - class\_priors, 9
  - matrix.h, 77
  - num\_of\_classes, 9
  - num\_of\_features, 10
- mat\_bayes\_model\_creat
  - matcreat.c, 28
  - matrix.h, 91
- mat\_bayes\_model\_free
  - matcreat.c, 28
  - matrix.h, 91
- mat\_binom
  - matpoly.c, 61
  - matrix.h, 91
- mat\_binom\_init



- matpoly.c, 61
- matrix.h, 91
- mat\_binom\_series\_table
  - matpoly.c, 65
  - matrix.h, 118
- mat\_bisquare\_wt
  - matfuncs.c, 48
  - matrix.h, 91
- mat\_bs\_delete
  - matdatastruct.c, 34
  - matrix.h, 92
- mat\_bs\_find
  - matdatastruct.c, 34
  - matrix.h, 92
- mat\_bs\_find\_max
  - matdatastruct.c, 34
  - matrix.h, 92
- mat\_bs\_find\_min
  - matdatastruct.c, 34
  - matrix.h, 92
- mat\_bs\_free
  - matdatastruct.c, 34
  - matrix.h, 92
- mat\_bs\_inorder
  - matdatastruct.c, 34
  - matrix.h, 92
- mat\_bs\_insert
  - matdatastruct.c, 34
  - matrix.h, 92
- mat\_bs\_make\_null
  - matdatastruct.c, 34
  - matrix.h, 92
- mat\_bsxfun
  - matmisc.c, 55
  - matrix.h, 92
- mat\_calc\_dist\_sq
  - matmisc.c, 55
  - matrix.h, 92
- mat\_cart2pol
  - matmisc.c, 55
  - matrix.h, 92
- mat\_cheby
  - matpoly.c, 61
  - matrix.h, 93
- mat\_cheby\_approx
  - matpoly.c, 61
  - matrix.h, 93
- mat\_cheby\_coeffs\_to\_poly
  - matpoly.c, 62
  - matrix.h, 93
- mat\_cheby\_init
  - matpoly.c, 62
  - matrix.h, 93
- mat\_cheby\_series\_table
  - matpoly.c, 65
  - matrix.h, 118
- mat\_cholesky
  - matrix.h, 94
- matsolve.c, 119
- mat\_cofact
  - matdet.c, 35
  - matrix.h, 94
- mat\_colcopy
  - matcreat.c, 28
  - matrix.h, 94
- mat\_concat
  - matconcat.c, 23
  - matrix.h, 94
- mat\_conjgrad
  - matrix.h, 95
  - matsolve.c, 119
- mat\_conv2
  - matfilter.c, 43
  - matrix.h, 95
- mat\_copy
  - matcreat.c, 29
  - matrix.h, 95
- mat\_corcol
  - matpca.c, 59
  - matrix.h, 95
- mat\_count\_words\_in\_line
  - matrix.h, 95
  - mattext.c, 122
- mat\_covcol
  - matpca.c, 59
  - matrix.h, 96
- mat\_creat
  - matcreat.c, 29
  - matrix.h, 96
- mat\_creat\_diag
  - matcreat.c, 29
  - matrix.h, 96
- mat\_det
  - matdet.c, 35
  - matrix.h, 96
- mat\_diagmul
  - matmul.c, 58
  - matrix.h, 96
- mat\_div\_dot
  - matdiv.c, 37
  - matrix.h, 96
- mat\_divs
  - matdiv.c, 37
  - matrix.h, 97
- mat\_dlmread
  - matrix.h, 97
  - mattext.c, 123
- mat\_dlmwrite
  - matrix.h, 97
  - mattext.c, 123
- mat\_dump
  - matdump.c, 39
  - matrix.h, 97
- mat\_dumpf
  - matdump.c, 39
  - matrix.h, 98

- mat\_durbin
  - matdurbn.c, 39
  - matrix.h, 98
- mat\_eig\_sym
  - matpca.c, 59
  - matrix.h, 98
- mat\_error
  - materr.c, 41
  - matrix.h, 98
- mat\_fdump
  - matdump.c, 39
  - matrix.h, 98
- mat\_fdumpf
  - matdump.c, 39
  - matrix.h, 98
- mat\_fft2
  - matfft.c, 43
  - matrix.h, 99
- mat\_fgetmat
  - matcreat.c, 29
  - matrix.h, 99
- mat\_fill
  - matcreat.c, 30
  - matrix.h, 99
- mat\_fill\_type
  - matcreat.c, 30
  - matrix.h, 99
- mat\_find
  - matrix.h, 100
  - matsearch.c, 118
- mat\_find\_within\_dist
  - matmisc.c, 56
  - matrix.h, 100
- mat\_fliplr
  - matflip.c, 44
  - matrix.h, 100
- mat\_flipud
  - matflip.c, 44
  - matrix.h, 100
- mat\_fnextline
  - matmisc.c, 56
  - matrix.h, 100
- mat\_free
  - matcreat.c, 30
  - matrix.h, 100
- mat\_get\_sub\_matrix\_from\_cols
  - matmisc.c, 56
  - matrix.h, 100
- mat\_get\_sub\_matrix\_from\_rows
  - matmisc.c, 56
  - matrix.h, 101
- mat\_get\_sub\_vector
  - matmisc.c, 57
  - matrix.h, 101
- mat\_gfunc
  - matfuncs.c, 48
  - matrix.h, 101
- mat\_gnode, 10
- matrix.h, 77
- next, 10
- v, 10
- weight, 10
- mat\_go\_next\_word
  - matrix.h, 101
  - mattext.c, 123
- mat\_graph, 10
  - adj, 11
  - dad, 11
  - id, 11
  - matrix.h, 77
  - nedges, 11
  - nvertices, 11
  - pq, 11
  - val, 11
  - vseq, 11
  - weighted, 11
  - z, 11
- mat\_graph\_adjlist
  - matgraph.c, 49
  - matrix.h, 102
- mat\_graph\_adjm\_to\_adjl
  - matgraph.c, 49
  - matrix.h, 102
- mat\_graph\_creat
  - matgraph.c, 49
  - matrix.h, 102
- mat\_graph\_dump
  - matgraph.c, 49
  - matrix.h, 102
- mat\_graph\_dumpf
  - matgraph.c, 49
  - matrix.h, 102
- mat\_graph\_reverse
  - matgraph.c, 49
  - matrix.h, 102
- mat\_graph\_search
  - matgraph.c, 49
  - matrix.h, 102
- mat\_graph\_visit
  - matgraph.c, 49
  - matrix.h, 102
- mat\_huber\_wt
  - matfuncs.c, 48
  - matrix.h, 102
- mat\_innerprod
  - matinnerprod.c, 49
  - matrix.h, 102
- mat\_int\_priorityqueue, 11
  - element, 12
  - length, 12
  - matrix.h, 77
  - p, 12
  - type, 12
- mat\_int\_priorityqueue\_creat
  - matdatastruct.c, 34
  - matrix.h, 102

- mat\_int\_priorityqueue\_dequeue
  - matdatastruct.c, 34
  - matrix.h, 102
- mat\_int\_priorityqueue\_enqueue
  - matdatastruct.c, 34
  - matrix.h, 102
- mat\_int\_priorityqueue\_free
  - matdatastruct.c, 34
  - matrix.h, 102
- mat\_int\_priorityqueue\_is\_empty
  - matdatastruct.c, 34
  - matrix.h, 102
- mat\_int\_priorityqueue\_update
  - matdatastruct.c, 34
  - matrix.h, 102
- mat\_int\_qadrat
  - matintegrate.c, 50
  - matrix.h, 102
- mat\_int\_queue, 12
  - head, 13
  - matrix.h, 78
  - p, 13
  - tail, 13
- mat\_int\_queue\_creat
  - matdatastruct.c, 34
  - matrix.h, 103
- mat\_int\_queue\_dequeue
  - matdatastruct.c, 34
  - matrix.h, 103
- mat\_int\_queue\_enqueue
  - matdatastruct.c, 34
  - matrix.h, 103
- mat\_int\_queue\_free
  - matdatastruct.c, 34
  - matrix.h, 103
- mat\_int\_queue\_is\_empty
  - matdatastruct.c, 34
  - matrix.h, 103
- mat\_int\_simpson
  - matintegrate.c, 50
  - matrix.h, 103
- mat\_int\_stack, 13
  - length, 13
  - matrix.h, 78
  - p, 13
  - stack, 13
- mat\_int\_stack\_creat
  - matdatastruct.c, 34
  - matrix.h, 103
- mat\_int\_stack\_free
  - matdatastruct.c, 34
  - matrix.h, 103
- mat\_int\_stack\_is\_empty
  - matdatastruct.c, 34
  - matrix.h, 103
- mat\_int\_stack\_pop
  - matdatastruct.c, 34
  - matrix.h, 103
- mat\_int\_stack\_push
  - matdatastruct.c, 34
  - matrix.h, 103
- mat\_int\_trapezoid
  - matintegrate.c, 50
  - matrix.h, 103
- mat\_intpqnode, 14
  - data, 14
  - matrix.h, 78
  - priority, 14
- mat\_inv
  - matinv.c, 51
  - matrix.h, 104
- mat\_isnumeric
  - matrix.h, 104
  - mattext.c, 123
- mat\_kdnode, 14
  - idx, 14
  - left, 14
  - matrix.h, 78
  - right, 14
  - x, 14
- mat\_kdtree, 15
  - \_is\_allocated, 15
  - data, 15
  - kdtree, 15
  - length, 15
  - matrix.h, 78
  - ndims, 15
- mat\_kdtree\_free
  - matkdtree.c, 52
  - matrix.h, 104
- mat\_kdtree\_k\_nearest
  - matkdtree.c, 52
  - matrix.h, 104
- mat\_kdtree\_make\_tree
  - matkdtree.c, 52
  - matrix.h, 105
- mat\_kdtree\_nearest
  - matkdtree.c, 52
  - matrix.h, 105
- mat\_kmeans
  - matprec.c, 65
  - matrix.h, 105
- mat\_least\_squares
  - matfit.c, 43
  - matrix.h, 105
- mat\_legendre
  - matpoly.c, 62
  - matrix.h, 105
- mat\_legendre\_init
  - matpoly.c, 62
  - matrix.h, 105
- mat\_legendre\_series\_table
  - matpoly.c, 65
  - matrix.h, 118
- mat\_linear\_ls\_fit
  - matfit.c, 44

- matrix.h, 105
- mat\_solve
  - matrix.h, 106
  - matsolve.c, 119
- mat\_solve\_durbin
  - matdurbn.c, 40
  - matrix.h, 106
- mat\_lu
  - matrix.h, 106
  - matsolve.c, 119
- mat\_max
  - matmaxmin.c, 53
  - matrix.h, 106
- mat\_mds
  - matmds.c, 53
  - matrix.h, 106
- mat\_mean
  - matmean.c, 53
  - matrix.h, 106
- mat\_mean\_col
  - matmean.c, 53
  - matrix.h, 106
- mat\_mean\_row
  - matmean.c, 54
  - matrix.h, 106
- mat\_median
  - matrix.h, 106
  - matsort.c, 120
- mat\_min
  - matmaxmin.c, 53
  - matrix.h, 107
- mat\_minor
  - matdet.c, 36
  - matrix.h, 107
- mat\_mtype\_priorityqueue, 15
  - element, 15
  - length, 15
  - matrix.h, 78
  - p, 16
  - type, 16
- mat\_mtype\_priorityqueue\_creat
  - matdatastruct.c, 34
  - matrix.h, 107
- mat\_mtype\_priorityqueue\_dequeue
  - matdatastruct.c, 34
  - matrix.h, 107
- mat\_mtype\_priorityqueue\_enqueue
  - matdatastruct.c, 34
  - matrix.h, 107
- mat\_mtype\_priorityqueue\_free
  - matdatastruct.c, 34
  - matrix.h, 107
- mat\_mtype\_priorityqueue\_is\_empty
  - matdatastruct.c, 35
  - matrix.h, 107
- mat\_mtype\_priorityqueue\_update
  - matdatastruct.c, 35
  - matrix.h, 107
- mat\_mtype\_queue, 16
  - head, 16
  - matrix.h, 78
  - p, 16
  - tail, 16
- mat\_mtype\_queue\_creat
  - matdatastruct.c, 35
  - matrix.h, 107
- mat\_mtype\_queue\_dequeue
  - matdatastruct.c, 35
  - matrix.h, 107
- mat\_mtype\_queue\_enqueue
  - matdatastruct.c, 35
  - matrix.h, 107
- mat\_mtype\_queue\_free
  - matdatastruct.c, 35
  - matrix.h, 107
- mat\_mtype\_queue\_is\_empty
  - matdatastruct.c, 35
  - matrix.h, 107
- mat\_mtype\_stack, 17
  - length, 17
  - matrix.h, 79
  - p, 17
  - stack, 17
- mat\_mtype\_stack\_creat
  - matdatastruct.c, 35
  - matrix.h, 107
- mat\_mtype\_stack\_free
  - matdatastruct.c, 35
  - matrix.h, 107
- mat\_mtype\_stack\_is\_empty
  - matdatastruct.c, 35
  - matrix.h, 107
- mat\_mtype\_stack\_pop
  - matdatastruct.c, 35
  - matrix.h, 107
- mat\_mtype\_stack\_push
  - matdatastruct.c, 35
  - matrix.h, 107
- mat\_mtypepqnode, 17
  - data, 18
  - matrix.h, 79
  - priority, 18
- mat\_mul
  - matmul.c, 58
  - matrix.h, 107
- mat\_mul\_dot
  - matmul.c, 59
  - matrix.h, 107
- mat\_mul\_fast
  - matmul.c, 59
  - matrix.h, 108
- mat\_muls
  - matmul.c, 59
  - matrix.h, 108
- mat\_nextline
  - matmisc.c, 57

- matrix.h, 108
- mat\_norm\_inf
  - matinnerprod.c, 49
  - matrix.h, 108
- mat\_norm\_one
  - matinnerprod.c, 49
  - matrix.h, 108
- mat\_norm\_p
  - matinnerprod.c, 49
  - matrix.h, 108
- mat\_omp
  - matpursuit.c, 65
  - matrix.h, 108
- mat\_order\_statistic
  - matrix.h, 108
  - matsort.c, 120
- mat\_pca
  - matpca.c, 59
  - matrix.h, 108
- mat\_perceptron, 18
  - class\_labels, 18
  - class\_weights, 18
  - istrained, 18
  - matrix.h, 79
  - num\_of\_classes, 18
  - num\_of\_features, 18
  - num\_of\_iterations, 19
- mat\_perceptron\_creat
  - matcreat.c, 30
  - matrix.h, 108
- mat\_perceptron\_free
  - matcreat.c, 30
  - matrix.h, 108
- mat\_perceptron\_test
  - matprec.c, 65
  - matrix.h, 108
- mat\_perceptron\_train
  - matprec.c, 65
  - matrix.h, 109
- mat\_perceptron\_train\_
  - matprec.c, 65
  - matrix.h, 109
- mat\_pick\_col
  - matmisc.c, 57
  - matrix.h, 109
- mat\_pick\_row
  - matmisc.c, 57
  - matrix.h, 109
- mat\_pinv
  - matpinv.c, 59
  - matrix.h, 109
- mat\_pol2cart
  - matmisc.c, 57
  - matrix.h, 109
- mat\_poly\_add
  - matpoly.c, 62
  - matrix.h, 110
- mat\_poly\_diff
  - matpoly.c, 63
  - matrix.h, 110
- mat\_poly\_diff\_eval
  - matpoly.c, 63
  - matrix.h, 110
- mat\_poly\_div
  - matpoly.c, 63
  - matrix.h, 110
- mat\_poly\_eval
  - matpoly.c, 63
  - matrix.h, 111
- mat\_poly\_mul
  - matpoly.c, 64
  - matrix.h, 111
- mat\_poly\_scale
  - matpoly.c, 64
  - matrix.h, 111
- mat\_poly\_shift
  - matpoly.c, 64
  - matrix.h, 111
- mat\_qintnode, 19
  - data, 19
  - matrix.h, 79
  - next, 19
- mat\_qmtyphenode, 19
  - data, 20
  - matrix.h, 79
  - next, 20
- mat\_qr
  - matdurbn.c, 40
  - matrix.h, 112
- mat\_qsort
  - matrix.h, 112
  - matsort.c, 120
- mat\_rand
  - matrand.c, 66
  - matrix.h, 112
- mat\_randexp
  - matrand.c, 66
  - matrix.h, 112
- mat\_randfun
  - matrand.c, 66
  - matrix.h, 112
- mat\_randn
  - matrand.c, 66
  - matrix.h, 112
- mat\_randperm
  - matrand.c, 66
  - matrix.h, 112
- mat\_randperm\_n
  - matrand.c, 66
  - matrix.h, 112
- mat\_read\_word
  - matrix.h, 112
  - mattext.c, 123
- mat\_reg\_inv
  - matinv.c, 51
  - matrix.h, 113

- mat\_rob\_least\_squares
  - matfit.c, [44](#)
  - matrix.h, [113](#)
- mat\_robust\_fit
  - matfit.c, [44](#)
  - matrix.h, [113](#)
- mat\_rowcopy
  - matcreat.c, [31](#)
  - matrix.h, [113](#)
- mat\_scpcol
  - matpca.c, [59](#)
  - matrix.h, [113](#)
- mat\_set\_seed
  - matrand.c, [66](#)
  - matrix.h, [113](#)
- mat\_sub
  - matrix.h, [113](#)
  - matsub.c, [121](#)
- mat\_submat
  - matrix.h, [113](#)
  - matsubx.c, [121](#)
- mat\_subs
  - matrix.h, [114](#)
  - matsub.c, [121](#)
- mat\_sum
  - matrix.h, [114](#)
  - matsum.c, [122](#)
- mat\_sum\_col
  - matrix.h, [114](#)
  - matsum.c, [122](#)
- mat\_sum\_row
  - matrix.h, [114](#)
  - matsum.c, [122](#)
- mat\_symtoeplz
  - matrix.h, [114](#)
  - mattoeplz.c, [125](#)
- mat\_tic
  - matrix.h, [114](#)
  - mattimers.c, [124](#)
- mat\_toc
  - matrix.h, [114](#)
  - mattimers.c, [124](#)
- mat\_toc\_print
  - matrix.h, [114](#)
  - mattimers.c, [124](#)
- mat\_tqli
  - matpca.c, [59](#)
  - matrix.h, [114](#)
- mat\_tran
  - matrix.h, [114](#)
  - mattran.c, [125](#)
- mat\_tred2
  - matpca.c, [59](#)
  - matrix.h, [114](#)
- mat\_tree\_node, [20](#)
  - element, [20](#)
  - left, [20](#)
  - matrix.h, [80](#)
  - right, [20](#)
- mat\_vectorize
  - matconv.c, [24](#)
  - matrix.h, [114](#)
- mat\_vectorize\_tr
  - matconv.c, [24](#)
  - matrix.h, [115](#)
- mat\_w\_least\_squares
  - matfit.c, [44](#)
  - matrix.h, [115](#)
- mat\_wpinv
  - matpinv.c, [60](#)
  - matrix.h, [115](#)
- mat\_xcopy
  - matcreat.c, [31](#)
  - matrix.h, [115](#)
- mat\_xjoin
  - matcreat.c, [31](#)
  - matrix.h, [116](#)
- matabs.c, [21](#)
  - mat\_abs, [21](#)
- matadd.c, [21](#)
  - int\_vec\_add, [22](#)
  - int\_vec\_adds, [22](#)
  - mat\_add, [22](#)
  - mat\_adds, [22](#)
- matcompress.c, [23](#)
- matconcat.c, [23](#)
  - int\_vec\_concat, [23](#)
  - mat\_concat, [23](#)
- matconv.c, [23](#)
  - int\_vec2\_mat, [24](#)
  - mat\_2int\_vec, [24](#)
  - mat\_vectorize, [24](#)
  - mat\_vectorize\_tr, [24](#)
- matcreat.c, [25](#)
  - int\_vec\_append, [26](#)
  - int\_vec\_copy, [26](#)
  - int\_vec\_creat, [26](#)
  - int\_vec\_fill, [27](#)
  - int\_vec\_fill\_type, [27](#)
  - int\_vec\_free, [27](#)
  - int\_vecstack\_creat, [27](#)
  - int\_vecstack\_free, [28](#)
  - mat\_bayes\_model\_creat, [28](#)
  - mat\_bayes\_model\_free, [28](#)
  - mat\_colcopy, [28](#)
  - mat\_copy, [29](#)
  - mat\_creat, [29](#)
  - mat\_creat\_diag, [29](#)
  - mat\_fgetmat, [29](#)
  - mat\_fill, [30](#)
  - mat\_fill\_type, [30](#)
  - mat\_free, [30](#)
  - mat\_perceptron\_creat, [30](#)
  - mat\_perceptron\_free, [30](#)
  - mat\_rowcopy, [31](#)
  - mat\_xcopy, [31](#)

- mat\_xjoin, [31](#)
- matstack\_append, [32](#)
- matstack\_creat, [32](#)
- matstack\_free, [32](#)
- matvec\_creat, [32](#)
- matvec\_free, [32](#)
- matdatastruct.c, [33](#)
  - mat\_bs\_delete, [34](#)
  - mat\_bs\_find, [34](#)
  - mat\_bs\_find\_max, [34](#)
  - mat\_bs\_find\_min, [34](#)
  - mat\_bs\_free, [34](#)
  - mat\_bs\_inorder, [34](#)
  - mat\_bs\_insert, [34](#)
  - mat\_bs\_make\_null, [34](#)
  - mat\_int\_priorityqueue\_creat, [34](#)
  - mat\_int\_priorityqueue\_dequeue, [34](#)
  - mat\_int\_priorityqueue\_enqueue, [34](#)
  - mat\_int\_priorityqueue\_free, [34](#)
  - mat\_int\_priorityqueue\_is\_empty, [34](#)
  - mat\_int\_priorityqueue\_update, [34](#)
  - mat\_int\_queue\_creat, [34](#)
  - mat\_int\_queue\_dequeue, [34](#)
  - mat\_int\_queue\_enqueue, [34](#)
  - mat\_int\_queue\_free, [34](#)
  - mat\_int\_queue\_is\_empty, [34](#)
  - mat\_int\_stack\_creat, [34](#)
  - mat\_int\_stack\_free, [34](#)
  - mat\_int\_stack\_is\_empty, [34](#)
  - mat\_int\_stack\_pop, [34](#)
  - mat\_int\_stack\_push, [34](#)
  - mat\_mtype\_priorityqueue\_creat, [34](#)
  - mat\_mtype\_priorityqueue\_dequeue, [34](#)
  - mat\_mtype\_priorityqueue\_enqueue, [34](#)
  - mat\_mtype\_priorityqueue\_free, [34](#)
  - mat\_mtype\_priorityqueue\_is\_empty, [35](#)
  - mat\_mtype\_priorityqueue\_update, [35](#)
  - mat\_mtype\_queue\_creat, [35](#)
  - mat\_mtype\_queue\_dequeue, [35](#)
  - mat\_mtype\_queue\_enqueue, [35](#)
  - mat\_mtype\_queue\_free, [35](#)
  - mat\_mtype\_queue\_is\_empty, [35](#)
  - mat\_mtype\_stack\_creat, [35](#)
  - mat\_mtype\_stack\_free, [35](#)
  - mat\_mtype\_stack\_is\_empty, [35](#)
  - mat\_mtype\_stack\_pop, [35](#)
  - mat\_mtype\_stack\_push, [35](#)
- matdet.c, [35](#)
  - mat\_cofact, [35](#)
  - mat\_det, [35](#)
  - mat\_minor, [36](#)
- matdiv.c, [36](#)
  - int\_vec\_div, [36](#)
  - int\_vec\_divs, [37](#)
  - mat\_div\_dot, [37](#)
  - mat\_divs, [37](#)
- matdump.c, [37](#)
  - int\_vec\_dump, [38](#)
  - int\_vec\_dumpf, [38](#)
  - int\_vec\_fdump, [38](#)
  - int\_vec\_fdumpf, [38](#)
  - mat\_dump, [39](#)
  - mat\_dumpf, [39](#)
  - mat\_fdump, [39](#)
  - mat\_fdumpf, [39](#)
- matdurbin.c, [39](#)
  - mat\_durbin, [39](#)
  - mat\_lsolve\_durbin, [40](#)
  - mat\_qr, [40](#)
- materr.c, [40](#)
  - gen\_error, [41](#)
  - graph\_error, [41](#)
  - int\_vec\_error, [41](#)
  - int\_vecstack\_error, [41](#)
  - mat\_error, [41](#)
  - matstack\_error, [42](#)
  - pq\_error, [42](#)
  - queue\_error, [42](#)
  - stack\_error, [42](#)
- matfft.c, [42](#)
  - mat\_fft2, [43](#)
- matfilter.c, [43](#)
  - mat\_conv2, [43](#)
- matfit.c, [43](#)
  - mat\_least\_squares, [43](#)
  - mat\_linear\_ls\_fit, [44](#)
  - mat\_rob\_least\_squares, [44](#)
  - mat\_robust\_fit, [44](#)
  - mat\_w\_least\_squares, [44](#)
- matflip.c, [44](#)
  - mat\_fliplr, [44](#)
  - mat\_flipud, [44](#)
- matfuncs.c, [44](#)
  - \_\_mat\_addfunc, [45](#)
  - \_\_mat\_arccosh, [45](#)
  - \_\_mat\_arcsinh, [45](#)
  - \_\_mat\_arctanh, [46](#)
  - \_\_mat\_bisquare\_wt, [46](#)
  - \_\_mat\_divfunc, [46](#)
  - \_\_mat\_huber\_wt, [46](#)
  - \_\_mat\_logplusone, [47](#)
  - \_\_mat\_mulfunc, [47](#)
  - \_\_mat\_sqrfunc, [47](#)
  - \_\_mat\_sqrtfunc, [47](#)
  - \_\_mat\_subfunc, [47](#)
  - mat\_bisquare\_wt, [48](#)
  - mat\_gfunc, [48](#)
  - mat\_huber\_wt, [48](#)
- matgraph.c, [49](#)
  - mat\_graph\_adjlist, [49](#)
  - mat\_graph\_adjm\_to\_adjl, [49](#)
  - mat\_graph\_creat, [49](#)
  - mat\_graph\_dump, [49](#)
  - mat\_graph\_dumpf, [49](#)
  - mat\_graph\_reverse, [49](#)
  - mat\_graph\_search, [49](#)

- mat\_graph\_visit, 49
- matinnerprod.c, 49
  - mat\_innerprod, 49
  - mat\_norm\_inf, 49
  - mat\_norm\_one, 49
  - mat\_norm\_p, 49
- matintegrate.c, 50
  - mat\_int\_qadrat, 50
  - mat\_int\_simpson, 50
  - mat\_int\_trapezoid, 50
- matinv.c, 51
  - mat\_inv, 51
  - mat\_reg\_inv, 51
- matkdtree.c, 51
  - mat\_kdtree\_free, 52
  - mat\_kdtree\_k\_nearest, 52
  - mat\_kdtree\_make\_tree, 52
  - mat\_kdtree\_nearest, 52
- matmaxmin.c, 53
  - mat\_max, 53
  - mat\_min, 53
- matmds.c, 53
  - \_\_mat\_mds\_metric, 53
  - \_\_mat\_mds\_nonmetric, 53
  - mat\_mds, 53
- matmean.c, 53
  - mat\_mean, 53
  - mat\_mean\_col, 53
  - mat\_mean\_row, 54
- matmisc.c, 54
  - \_\_mat\_cart2pol, 54
  - \_\_mat\_pol2cart, 54
  - int\_vec\_permute\_vect, 54
  - int\_vec\_unique, 55
  - mat\_bsxfun, 55
  - mat\_calc\_dist\_sq, 55
  - mat\_cart2pol, 55
  - mat\_find\_within\_dist, 56
  - mat\_fnextline, 56
  - mat\_get\_sub\_matrix\_from\_cols, 56
  - mat\_get\_sub\_matrix\_from\_rows, 56
  - mat\_get\_sub\_vector, 57
  - mat\_nextline, 57
  - mat\_pick\_col, 57
  - mat\_pick\_row, 57
  - mat\_pol2cart, 57
  - mats\_isinf, 58
  - mats\_isnan, 58
- matmul.c, 58
  - int\_vec\_mul, 58
  - int\_vec\_muls, 58
  - mat\_diagmul, 58
  - mat\_mul, 58
  - mat\_mul\_dot, 59
  - mat\_mul\_fast, 59
  - mat\_muls, 59
- matpca.c, 59
  - mat\_corcol, 59
  - mat\_covcol, 59
  - mat\_eig\_sym, 59
  - mat\_pca, 59
  - mat\_scpcol, 59
  - mat\_tqli, 59
  - mat\_tred2, 59
- matpinv.c, 59
  - mat\_pinv, 59
  - mat\_wpinv, 60
- matpoly.c, 60
  - mat\_binom, 61
  - mat\_binom\_init, 61
  - mat\_binom\_series\_table, 65
  - mat\_cheby, 61
  - mat\_cheby\_approx, 61
  - mat\_cheby\_coeffs\_to\_poly, 62
  - mat\_cheby\_init, 62
  - mat\_cheby\_series\_table, 65
  - mat\_legendre, 62
  - mat\_legendre\_init, 62
  - mat\_legendre\_series\_table, 65
  - mat\_poly\_add, 62
  - mat\_poly\_diff, 63
  - mat\_poly\_diff\_eval, 63
  - mat\_poly\_div, 63
  - mat\_poly\_eval, 63
  - mat\_poly\_mul, 64
  - mat\_poly\_scale, 64
  - mat\_poly\_shift, 64
- matprec.c, 65
  - mat\_bayes\_classifier\_test, 65
  - mat\_bayes\_classifier\_train, 65
  - mat\_kmeans, 65
  - mat\_perceptron\_test, 65
  - mat\_perceptron\_train, 65
  - mat\_perceptron\_train\_, 65
- matpursuit.c, 65
  - mat\_omp, 65
- matrand.c, 65
  - \_\_mat\_rand, 66
  - \_\_mat\_randexp, 66
  - \_\_mat\_randfun, 66
  - \_\_mat\_randn, 66
  - int\_vec\_randperm, 66
  - MAT\_SEED, 66
  - MAT\_SET\_SEED, 66
  - mat\_rand, 66
  - mat\_randexp, 66
  - mat\_randfun, 66
  - mat\_randn, 66
  - mat\_randperm, 66
  - mat\_randperm\_n, 66
  - mat\_set\_seed, 66
- matrix.c, 66
- matrix.h, 66
  - \_\_int\_vec\_creat, 80
  - \_\_int\_vecstack\_creat, 80
  - \_\_mat\_addfunc, 80



- [\\_\\_mat\\_arccosh, 81](#)
- [\\_\\_mat\\_arcsinh, 81](#)
- [\\_\\_mat\\_arctanh, 81](#)
- [\\_\\_mat\\_bisquare\\_wt, 81](#)
- [\\_\\_mat\\_cart2pol, 81](#)
- [\\_\\_mat\\_creat, 82](#)
- [\\_\\_mat\\_divfunc, 82](#)
- [\\_\\_mat\\_fft, 82](#)
- [\\_\\_mat\\_huber\\_wt, 82](#)
- [\\_\\_mat\\_kd\\_find\\_median, 82](#)
- [\\_\\_mat\\_kdtree\\_k\\_nearest, 82](#)
- [\\_\\_mat\\_kdtree\\_make\\_tree, 82](#)
- [\\_\\_mat\\_kdtree\\_nearest, 82](#)
- [\\_\\_mat\\_lint, 82](#)
- [\\_\\_mat\\_logplusone, 82](#)
- [\\_\\_mat\\_mds\\_metric, 83](#)
- [\\_\\_mat\\_mds\\_nonmetric, 83](#)
- [\\_\\_mat\\_mulfunc, 83](#)
- [\\_\\_mat\\_pol2cart, 83](#)
- [\\_\\_mat\\_powerof2, 83](#)
- [\\_\\_mat\\_quicksort, 83](#)
- [\\_\\_mat\\_rand, 83](#)
- [\\_\\_mat\\_randexp, 83](#)
- [\\_\\_mat\\_randfun, 83](#)
- [\\_\\_mat\\_randn, 83](#)
- [\\_\\_mat\\_sqrfunc, 83](#)
- [\\_\\_mat\\_sqrtfunc, 83](#)
- [\\_\\_mat\\_subfunc, 84](#)
- [\\_\\_matstack\\_creat, 84](#)
- [gen\\_abs\\_ceil, 84](#)
- [gen\\_eq, 84](#)
- [gen\\_error, 84](#)
- [gen\\_gt, 84](#)
- [gen\\_lt, 84](#)
- [graph\\_error, 84](#)
- [INT\\_VECSTACK, 77](#)
- [INT\\_VECTOR, 77](#)
- [int\\_vec2\\_mat, 84](#)
- [int\\_vec\\_add, 85](#)
- [int\\_vec\\_adds, 85](#)
- [int\\_vec\\_append, 85](#)
- [int\\_vec\\_concat, 85](#)
- [int\\_vec\\_copy, 86](#)
- [int\\_vec\\_creat, 86](#)
- [int\\_vec\\_div, 86](#)
- [int\\_vec\\_divs, 86](#)
- [int\\_vec\\_dump, 87](#)
- [int\\_vec\\_dumpf, 87](#)
- [int\\_vec\\_error, 87](#)
- [int\\_vec\\_fdump, 87](#)
- [int\\_vec\\_fdumpf, 87](#)
- [int\\_vec\\_fill, 87](#)
- [int\\_vec\\_fill\\_type, 88](#)
- [int\\_vec\\_find, 88](#)
- [int\\_vec\\_free, 88](#)
- [int\\_vec\\_mul, 88](#)
- [int\\_vec\\_muls, 88](#)
- [int\\_vec\\_permute\\_vect, 88](#)
- [int\\_vec\\_randperm, 89](#)
- [int\\_vec\\_sub, 89](#)
- [int\\_vec\\_subs, 89](#)
- [int\\_vec\\_unique, 89](#)
- [int\\_vecstack\\_creat, 89](#)
- [int\\_vecstack\\_error, 89](#)
- [int\\_vecstack\\_free, 89](#)
- [MAT\\_BAYES\\_MODEL, 77](#)
- [MAT\\_CLOCK\\_TIME, 118](#)
- [MAT\\_GNODE, 77](#)
- [MAT\\_GRAPH, 77](#)
- [MAT\\_INT\\_QUEUE, 78](#)
- [MAT\\_INT\\_STACK, 78](#)
- [MAT\\_INTPQNODE, 78](#)
- [MAT\\_KDNODE, 78](#)
- [MAT\\_KDTREE, 78](#)
- [MAT\\_MTYPE\\_QUEUE, 79](#)
- [MAT\\_MTYPE\\_STACK, 79](#)
- [MAT\\_MTYPEPQNODE, 79](#)
- [MAT\\_PERCEPTRON, 79](#)
- [MAT\\_QINTNODE, 79](#)
- [MAT\\_QMTYPENODE, 80](#)
- [MAT\\_SEED, 118](#)
- [MAT\\_SET\\_SEED, 118](#)
- [MAT\\_TREE, 80](#)
- [MAT\\_TREE\\_NODE, 80](#)
- [MATRIX, 80](#)
- [MATSTACK, 80](#)
- [MATVEC\\_DPOINTER, 80](#)
- [mat\\_2int\\_vec, 90](#)
- [mat\\_abs, 90](#)
- [mat\\_add, 90](#)
- [mat\\_adds, 90](#)
- [mat\\_backsubs1, 91](#)
- [mat\\_bayes\\_classifier\\_test, 91](#)
- [mat\\_bayes\\_classifier\\_train, 91](#)
- [mat\\_bayes\\_model, 77](#)
- [mat\\_bayes\\_model\\_creat, 91](#)
- [mat\\_bayes\\_model\\_free, 91](#)
- [mat\\_binom, 91](#)
- [mat\\_binom\\_init, 91](#)
- [mat\\_binom\\_series\\_table, 118](#)
- [mat\\_bisquare\\_wt, 91](#)
- [mat\\_bs\\_delete, 92](#)
- [mat\\_bs\\_find, 92](#)
- [mat\\_bs\\_find\\_max, 92](#)
- [mat\\_bs\\_find\\_min, 92](#)
- [mat\\_bs\\_free, 92](#)
- [mat\\_bs\\_inorder, 92](#)
- [mat\\_bs\\_insert, 92](#)
- [mat\\_bs\\_make\\_null, 92](#)
- [mat\\_bsxfun, 92](#)
- [mat\\_calc\\_dist\\_sq, 92](#)
- [mat\\_cart2pol, 92](#)
- [mat\\_cheby, 93](#)
- [mat\\_cheby\\_approx, 93](#)
- [mat\\_cheby\\_coeffs\\_to\\_poly, 93](#)
- [mat\\_cheby\\_init, 93](#)

- mat\_cheby\_series\_table, 118
- mat\_cholesky, 94
- mat\_cofact, 94
- mat\_colcopy, 94
- mat\_concat, 94
- mat\_conjgrad, 95
- mat\_conv2, 95
- mat\_copy, 95
- mat\_corcol, 95
- mat\_count\_words\_in\_line, 95
- mat\_covcol, 96
- mat\_creat, 96
- mat\_creat\_diag, 96
- mat\_det, 96
- mat\_diagmul, 96
- mat\_div\_dot, 96
- mat\_divs, 97
- mat\_dlmread, 97
- mat\_dlmwrite, 97
- mat\_dump, 97
- mat\_dumpf, 98
- mat\_durbin, 98
- mat\_eig\_sym, 98
- mat\_error, 98
- mat\_fdump, 98
- mat\_fdumpf, 98
- mat\_fft2, 99
- mat\_fgetmat, 99
- mat\_fill, 99
- mat\_fill\_type, 99
- mat\_find, 100
- mat\_find\_within\_dist, 100
- mat\_fliplr, 100
- mat\_flipud, 100
- mat\_fnextline, 100
- mat\_free, 100
- mat\_get\_sub\_matrix\_from\_cols, 100
- mat\_get\_sub\_matrix\_from\_rows, 101
- mat\_get\_sub\_vector, 101
- mat\_gfunc, 101
- mat\_gnode, 77
- mat\_go\_next\_word, 101
- mat\_graph, 77
- mat\_graph\_adjlist, 102
- mat\_graph\_adjm\_to\_adjl, 102
- mat\_graph\_creat, 102
- mat\_graph\_dump, 102
- mat\_graph\_dumpf, 102
- mat\_graph\_reverse, 102
- mat\_graph\_search, 102
- mat\_graph\_visit, 102
- mat\_huber\_wt, 102
- mat\_innerprod, 102
- mat\_int\_priorityqueue, 77
- mat\_int\_priorityqueue\_creat, 102
- mat\_int\_priorityqueue\_dequeue, 102
- mat\_int\_priorityqueue\_enqueue, 102
- mat\_int\_priorityqueue\_free, 102
- mat\_int\_priorityqueue\_is\_empty, 102
- mat\_int\_priorityqueue\_update, 102
- mat\_int\_qadrat, 102
- mat\_int\_queue, 78
- mat\_int\_queue\_creat, 103
- mat\_int\_queue\_dequeue, 103
- mat\_int\_queue\_enqueue, 103
- mat\_int\_queue\_free, 103
- mat\_int\_queue\_is\_empty, 103
- mat\_int\_simpson, 103
- mat\_int\_stack, 78
- mat\_int\_stack\_creat, 103
- mat\_int\_stack\_free, 103
- mat\_int\_stack\_is\_empty, 103
- mat\_int\_stack\_pop, 103
- mat\_int\_stack\_push, 103
- mat\_int\_trapezoid, 103
- mat\_intpqnode, 78
- mat\_inv, 104
- mat\_isnumeric, 104
- mat\_kdnode, 78
- mat\_kdtree, 78
- mat\_kdtree\_free, 104
- mat\_kdtree\_k\_nearest, 104
- mat\_kdtree\_make\_tree, 105
- mat\_kdtree\_nearest, 105
- mat\_kmeans, 105
- mat\_least\_squares, 105
- mat\_legendre, 105
- mat\_legendre\_init, 105
- mat\_legendre\_series\_table, 118
- mat\_linear\_ls\_fit, 105
- mat\_lsolve, 106
- mat\_lsolve\_durbin, 106
- mat\_lu, 106
- mat\_max, 106
- mat\_mds, 106
- mat\_mean, 106
- mat\_mean\_col, 106
- mat\_mean\_row, 106
- mat\_median, 106
- mat\_min, 107
- mat\_minor, 107
- mat\_mtype\_priorityqueue, 78
- mat\_mtype\_priorityqueue\_creat, 107
- mat\_mtype\_priorityqueue\_dequeue, 107
- mat\_mtype\_priorityqueue\_enqueue, 107
- mat\_mtype\_priorityqueue\_free, 107
- mat\_mtype\_priorityqueue\_is\_empty, 107
- mat\_mtype\_priorityqueue\_update, 107
- mat\_mtype\_queue, 78
- mat\_mtype\_queue\_creat, 107
- mat\_mtype\_queue\_dequeue, 107
- mat\_mtype\_queue\_enqueue, 107
- mat\_mtype\_queue\_free, 107
- mat\_mtype\_queue\_is\_empty, 107
- mat\_mtype\_stack, 79
- mat\_mtype\_stack\_creat, 107

mat\_mtype\_stack\_free, 107  
 mat\_mtype\_stack\_is\_empty, 107  
 mat\_mtype\_stack\_pop, 107  
 mat\_mtype\_stack\_push, 107  
 mat\_mtypepqnode, 79  
 mat\_mul, 107  
 mat\_mul\_dot, 107  
 mat\_mul\_fast, 108  
 mat\_muls, 108  
 mat\_nextline, 108  
 mat\_norm\_inf, 108  
 mat\_norm\_one, 108  
 mat\_norm\_p, 108  
 mat\_omp, 108  
 mat\_order\_statistic, 108  
 mat\_pca, 108  
 mat\_perceptron, 79  
 mat\_perceptron\_creat, 108  
 mat\_perceptron\_free, 108  
 mat\_perceptron\_test, 108  
 mat\_perceptron\_train, 109  
 mat\_perceptron\_train\_, 109  
 mat\_pick\_col, 109  
 mat\_pick\_row, 109  
 mat\_pinv, 109  
 mat\_pol2cart, 109  
 mat\_poly\_add, 110  
 mat\_poly\_diff, 110  
 mat\_poly\_diff\_eval, 110  
 mat\_poly\_div, 110  
 mat\_poly\_eval, 111  
 mat\_poly\_mul, 111  
 mat\_poly\_scale, 111  
 mat\_poly\_shift, 111  
 mat\_qintnode, 79  
 mat\_qmtyphenode, 79  
 mat\_qr, 112  
 mat\_qsort, 112  
 mat\_rand, 112  
 mat\_randexp, 112  
 mat\_randfun, 112  
 mat\_randn, 112  
 mat\_randperm, 112  
 mat\_randperm\_n, 112  
 mat\_read\_word, 112  
 mat\_reg\_inv, 113  
 mat\_rob\_least\_squares, 113  
 mat\_robust\_fit, 113  
 mat\_rowcopy, 113  
 mat\_scpcol, 113  
 mat\_set\_seed, 113  
 mat\_sub, 113  
 mat\_submat, 113  
 mat\_subs, 114  
 mat\_sum, 114  
 mat\_sum\_col, 114  
 mat\_sum\_row, 114  
 mat\_symtoeplz, 114  
 mat\_tic, 114  
 mat\_toc, 114  
 mat\_toc\_print, 114  
 mat\_tqli, 114  
 mat\_tran, 114  
 mat\_tred2, 114  
 mat\_tree\_node, 80  
 mat\_vectorize, 114  
 mat\_vectorize\_tr, 115  
 mat\_w\_least\_squares, 115  
 mat\_wpinv, 115  
 mat\_xcopy, 115  
 mat\_xjoin, 116  
 mats\_isinf, 116  
 mats\_isnan, 116  
 matstack\_append, 116  
 matstack\_creat, 117  
 matstack\_error, 117  
 matstack\_free, 117  
 matvec\_creat, 117  
 matvec\_free, 117  
 pq\_error, 118  
 queue\_error, 118  
 stack\_error, 118  
 mats\_isinf  
     matmisc.c, 58  
     matrix.h, 116  
 mats\_isnan  
     matmisc.c, 58  
     matrix.h, 116  
 matsearch.c, 118  
     int\_vec\_find, 118  
     mat\_find, 118  
 matsolve.c, 119  
     mat\_backsubs1, 119  
     mat\_cholesky, 119  
     mat\_conjgrad, 119  
     mat\_lsolve, 119  
     mat\_lu, 119  
 matsort.c, 119  
     mat\_median, 120  
     mat\_order\_statistic, 120  
     mat\_qsort, 120  
 matstack\_append  
     matcreat.c, 32  
     matrix.h, 116  
 matstack\_creat  
     matcreat.c, 32  
     matrix.h, 117  
 matstack\_error  
     materr.c, 42  
     matrix.h, 117  
 matstack\_free  
     matcreat.c, 32  
     matrix.h, 117  
 matstdrels.c, 120  
     gen\_abs\_ceil, 121  
     gen\_eq, 121

- gen\_gt, [121](#)
  - gen\_lt, [121](#)
- matsub.c, [121](#)
  - int\_vec\_sub, [121](#)
  - int\_vec\_subs, [121](#)
  - mat\_sub, [121](#)
  - mat\_subs, [121](#)
- matsubx.c, [121](#)
  - mat\_submat, [121](#)
- matsum.c, [122](#)
  - mat\_sum, [122](#)
  - mat\_sum\_col, [122](#)
  - mat\_sum\_row, [122](#)
- mattext.c, [122](#)
  - mat\_count\_words\_in\_line, [122](#)
  - mat\_dlmread, [123](#)
  - mat\_dlmwrite, [123](#)
  - mat\_go\_next\_word, [123](#)
  - mat\_isnumeric, [123](#)
  - mat\_read\_word, [123](#)
- mattimers.c, [124](#)
  - MAT\_CLOCK\_TIME, [124](#)
  - mat\_tic, [124](#)
  - mat\_toc, [124](#)
  - mat\_toc\_print, [124](#)
- mattoepz.c, [124](#)
  - mat\_symtoeplz, [125](#)
- mattran.c, [125](#)
  - mat\_tran, [125](#)
- matvec\_creat
  - matcreat.c, [32](#)
  - matrix.h, [117](#)
- matvec\_free
  - matcreat.c, [32](#)
  - matrix.h, [117](#)
- ndims
  - mat\_kdtree, [15](#)
- nedges
  - mat\_graph, [11](#)
- next
  - mat\_gnode, [10](#)
  - mat\_qintnode, [19](#)
  - mat\_qmtyphenode, [20](#)
- num\_of\_classes
  - mat\_bayes\_model, [9](#)
  - mat\_perceptron, [18](#)
- num\_of\_features
  - mat\_bayes\_model, [10](#)
  - mat\_perceptron, [18](#)
- num\_of\_iterations
  - mat\_perceptron, [19](#)
- nvertices
  - mat\_graph, [11](#)
- p
  - mat\_int\_priorityqueue, [12](#)
  - mat\_int\_queue, [13](#)
  - mat\_int\_stack, [13](#)
  - mat\_mtype\_priorityqueue, [16](#)
  - mat\_mtype\_queue, [16](#)
  - mat\_mtype\_stack, [17](#)
- pq
  - mat\_graph, [11](#)
- pq\_error
  - materr.c, [42](#)
  - matrix.h, [118](#)
- priority
  - mat\_intpqnode, [14](#)
  - mat\_mtypepqnode, [18](#)
- queue\_error
  - materr.c, [42](#)
  - matrix.h, [118](#)
- README.md, [125](#)
- right
  - mat\_kdnode, [14](#)
  - mat\_tree\_node, [20](#)
- stack
  - mat\_int\_stack, [13](#)
  - mat\_mtype\_stack, [17](#)
- stack\_error
  - materr.c, [42](#)
  - matrix.h, [118](#)
- tail
  - mat\_int\_queue, [13](#)
  - mat\_mtype\_queue, [16](#)
- type
  - mat\_int\_priorityqueue, [12](#)
  - mat\_mtype\_priorityqueue, [16](#)
- v
  - mat\_gnode, [10](#)
- val
  - mat\_graph, [11](#)
- vseq
  - mat\_graph, [11](#)
- weight
  - mat\_gnode, [10](#)
- weighted
  - mat\_graph, [11](#)
- x
  - mat\_kdnode, [14](#)
- z
  - mat\_graph, [11](#)