

MatrixLab

1.3

Generated by Doxygen 1.8.13

Contents

1	Matrixlab	1
1.1	Introduction	1
2	![[https://raw.githubusercontent.com/mohammadul/matrixlab/master/matrixlab.ico]] Matrixlab	3
3	Data Structure Index	5
3.1	Data Structures	5
4	File Index	7
4.1	File List	7
5	Data Structure Documentation	9
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	10
5.1.2.3	class_means	10
5.1.2.4	class_priors	10
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	11
5.2.2	Field Documentation	11
5.2.2.1	next	11

5.2.2.2	v	11
5.2.2.3	weight	11
5.3	mat_graph Struct Reference	11
5.3.1	Detailed Description	12
5.3.2	Field Documentation	12
5.3.2.1	adj	12
5.3.2.2	dad	12
5.3.2.3	id	12
5.3.2.4	nedges	12
5.3.2.5	nvertices	12
5.3.2.6	pq	12
5.3.2.7	val	13
5.3.2.8	vseq	13
5.3.2.9	weighted	13
5.3.2.10	z	13
5.4	mat_int_priorityqueue Struct Reference	13
5.4.1	Detailed Description	13
5.4.2	Field Documentation	14
5.4.2.1	element	14
5.4.2.2	length	14
5.4.2.3	p	14
5.4.2.4	type	14
5.5	mat_int_queue Struct Reference	14
5.5.1	Detailed Description	15
5.5.2	Field Documentation	15
5.5.2.1	head	15
5.5.2.2	p	15
5.5.2.3	tail	15
5.6	mat_int_stack Struct Reference	15
5.6.1	Detailed Description	15

5.6.2	Field Documentation	16
5.6.2.1	length	16
5.6.2.2	p	16
5.6.2.3	stack	16
5.7	mat_intpqnode Struct Reference	16
5.7.1	Detailed Description	16
5.7.2	Field Documentation	16
5.7.2.1	data	17
5.7.2.2	priority	17
5.8	mat_kdnode Struct Reference	17
5.8.1	Field Documentation	17
5.8.1.1	idx	17
5.8.1.2	left	17
5.8.1.3	right	18
5.8.1.4	x	18
5.9	mat_kdtree Struct Reference	18
5.9.1	Field Documentation	18
5.9.1.1	_is_allocated	18
5.9.1.2	data	18
5.9.1.3	kdtree	19
5.9.1.4	length	19
5.9.1.5	ndims	19
5.10	mat_mtype_priorityqueue Struct Reference	19
5.10.1	Detailed Description	19
5.10.2	Field Documentation	19
5.10.2.1	element	20
5.10.2.2	length	20
5.10.2.3	p	20
5.10.2.4	type	20
5.11	mat_mtype_queue Struct Reference	20

5.11.1 Detailed Description	20
5.11.2 Field Documentation	21
5.11.2.1 head	21
5.11.2.2 p	21
5.11.2.3 tail	21
5.12 mat_mtype_stack Struct Reference	21
5.12.1 Detailed Description	21
5.12.2 Field Documentation	21
5.12.2.1 length	22
5.12.2.2 p	22
5.12.2.3 stack	22
5.13 mat_mtypepqnode Struct Reference	22
5.13.1 Detailed Description	22
5.13.2 Field Documentation	22
5.13.2.1 data	22
5.13.2.2 priority	23
5.14 mat_perceptron Struct Reference	23
5.14.1 Detailed Description	23
5.14.2 Field Documentation	23
5.14.2.1 class_labels	23
5.14.2.2 class_weights	23
5.14.2.3 istrained	24
5.14.2.4 num_of_classes	24
5.14.2.5 num_of_features	24
5.14.2.6 num_of_iterations	24
5.15 mat_qintnode Struct Reference	24
5.15.1 Detailed Description	24
5.15.2 Field Documentation	25
5.15.2.1 data	25
5.15.2.2 next	25
5.16 mat_qmtyphenode Struct Reference	25
5.16.1 Detailed Description	25
5.16.2 Field Documentation	25
5.16.2.1 data	25
5.16.2.2 next	26
5.17 mat_tree_node Struct Reference	26
5.17.1 Detailed Description	26
5.17.2 Field Documentation	26
5.17.2.1 element	26
5.17.2.2 left	26
5.17.2.3 right	26

6 File Documentation	27
6.1 matabs.c File Reference	27
6.1.1 Function Documentation	27
6.1.1.1 int_vec_abs()	27
6.1.1.2 mat_abs()	28
6.2 matadd.c File Reference	28
6.2.1 Function Documentation	28
6.2.1.1 int_vec_add()	28
6.2.1.2 int_vec_adds()	29
6.2.1.3 mat_add()	29
6.2.1.4 mat_adds()	30
6.3 matcompress.c File Reference	30
6.4 matconcat.c File Reference	30
6.4.1 Function Documentation	30
6.4.1.1 int_vec_concat()	30
6.4.1.2 mat_concat()	31
6.5 matconv.c File Reference	31
6.5.1 Function Documentation	32
6.5.1.1 int_vec2_mat()	32
6.5.1.2 mat_2int_vec()	32
6.5.1.3 mat_vectorize()	32
6.5.1.4 mat_vectorize_tr()	33
6.6 matcreat.c File Reference	33
6.6.1 Function Documentation	34
6.6.1.1 int_vec_append()	35
6.6.1.2 int_vec_copy()	35
6.6.1.3 int_vec_creat()	35
6.6.1.4 int_vec_fill()	36
6.6.1.5 int_vec_fill_type()	36
6.6.1.6 int_vec_free()	36

6.6.1.7	int_vecstack_creat()	37
6.6.1.8	int_vecstack_free()	37
6.6.1.9	mat_bayes_model_creat()	37
6.6.1.10	mat_bayes_model_free()	38
6.6.1.11	mat_colcopy()	38
6.6.1.12	mat_copy()	38
6.6.1.13	mat_creat()	39
6.6.1.14	mat_creat_diag()	39
6.6.1.15	mat_fgetmat()	40
6.6.1.16	mat_fill()	40
6.6.1.17	mat_fill_type()	40
6.6.1.18	mat_free()	41
6.6.1.19	mat_perceptron_creat()	41
6.6.1.20	mat_perceptron_free()	41
6.6.1.21	mat_rowcopy()	42
6.6.1.22	mat_xcopy()	42
6.6.1.23	mat_xjoin()	43
6.6.1.24	matstack_append()	43
6.6.1.25	matstack_creat()	44
6.6.1.26	matstack_free()	44
6.6.1.27	matvec_creat()	44
6.6.1.28	matvec_free()	44
6.7	matdatastruct.c File Reference	45
6.7.1	Function Documentation	46
6.7.1.1	mat_bs_delete()	46
6.7.1.2	mat_bs_find()	46
6.7.1.3	mat_bs_find_max()	46
6.7.1.4	mat_bs_find_min()	46
6.7.1.5	mat_bs_free()	46
6.7.1.6	mat_bs_inorder()	46

6.7.1.7	<code>mat_bs_insert()</code>	47
6.7.1.8	<code>mat_bs_make_null()</code>	47
6.7.1.9	<code>mat_int_priorityqueue_creat()</code>	47
6.7.1.10	<code>mat_int_priorityqueue_dequeue()</code>	47
6.7.1.11	<code>mat_int_priorityqueue_enqueue()</code>	47
6.7.1.12	<code>mat_int_priorityqueue_free()</code>	47
6.7.1.13	<code>mat_int_priorityqueue_is_empty()</code>	48
6.7.1.14	<code>mat_int_priorityqueue_update()</code>	48
6.7.1.15	<code>mat_int_queue_creat()</code>	48
6.7.1.16	<code>mat_int_queue_dequeue()</code>	48
6.7.1.17	<code>mat_int_queue_enqueue()</code>	48
6.7.1.18	<code>mat_int_queue_free()</code>	48
6.7.1.19	<code>mat_int_queue_is_empty()</code>	49
6.7.1.20	<code>mat_int_stack_creat()</code>	49
6.7.1.21	<code>mat_int_stack_free()</code>	49
6.7.1.22	<code>mat_int_stack_is_empty()</code>	49
6.7.1.23	<code>mat_int_stack_pop()</code>	49
6.7.1.24	<code>mat_int_stack_push()</code>	49
6.7.1.25	<code>mat_mtype_priorityqueue_creat()</code>	49
6.7.1.26	<code>mat_mtype_priorityqueue_dequeue()</code>	50
6.7.1.27	<code>mat_mtype_priorityqueue_enqueue()</code>	50
6.7.1.28	<code>mat_mtype_priorityqueue_free()</code>	50
6.7.1.29	<code>mat_mtype_priorityqueue_is_empty()</code>	50
6.7.1.30	<code>mat_mtype_priorityqueue_update()</code>	50
6.7.1.31	<code>mat_mtype_queue_creat()</code>	50
6.7.1.32	<code>mat_mtype_queue_dequeue()</code>	51
6.7.1.33	<code>mat_mtype_queue_enqueue()</code>	51
6.7.1.34	<code>mat_mtype_queue_free()</code>	51
6.7.1.35	<code>mat_mtype_queue_is_empty()</code>	51
6.7.1.36	<code>mat_mtype_stack_creat()</code>	51

6.7.1.37	<code>mat_mtype_stack_free()</code>	51
6.7.1.38	<code>mat_mtype_stack_is_empty()</code>	51
6.7.1.39	<code>mat_mtype_stack_pop()</code>	52
6.7.1.40	<code>mat_mtype_stack_push()</code>	52
6.8	<code>matdet.c</code> File Reference	52
6.8.1	Function Documentation	52
6.8.1.1	<code>mat_cofact()</code>	52
6.8.1.2	<code>mat_det()</code>	53
6.8.1.3	<code>mat_minor()</code>	53
6.9	<code>matdiv.c</code> File Reference	53
6.9.1	Function Documentation	54
6.9.1.1	<code>int_vec_div()</code>	54
6.9.1.2	<code>int_vec_divs()</code>	54
6.9.1.3	<code>int_vec_divs_inv()</code>	55
6.9.1.4	<code>int_vec_inv()</code>	55
6.9.1.5	<code>mat_div_dot()</code>	56
6.9.1.6	<code>mat_divs()</code>	56
6.9.1.7	<code>mat_divs_inv()</code>	57
6.9.1.8	<code>mat_inv_dot()</code>	57
6.10	<code>matdump.c</code> File Reference	57
6.10.1	Function Documentation	58
6.10.1.1	<code>int_vec_dump()</code>	58
6.10.1.2	<code>int_vec_dumpf()</code>	58
6.10.1.3	<code>int_vec_fdump()</code>	59
6.10.1.4	<code>int_vec_fdumpf()</code>	59
6.10.1.5	<code>mat_dump()</code>	59
6.10.1.6	<code>mat_dumpf()</code>	59
6.10.1.7	<code>mat_fdump()</code>	60
6.10.1.8	<code>mat_fdumpf()</code>	60
6.11	<code>matdurbn.c</code> File Reference	60

6.11.1	Function Documentation	61
6.11.1.1	mat_durbin()	61
6.11.1.2	mat_solve_durbin()	61
6.11.1.3	mat_qr()	62
6.12	materr.c File Reference	62
6.12.1	Function Documentation	63
6.12.1.1	gen_error()	63
6.12.1.2	graph_error()	63
6.12.1.3	int_vec_error()	63
6.12.1.4	int_vecstack_error()	64
6.12.1.5	mat_error()	64
6.12.1.6	matstack_error()	64
6.12.1.7	pq_error()	64
6.12.1.8	queue_error()	65
6.12.1.9	stack_error()	65
6.13	matfft.c File Reference	65
6.13.1	Function Documentation	65
6.13.1.1	mat_fft2()	66
6.14	matfilter.c File Reference	66
6.14.1	Function Documentation	66
6.14.1.1	mat_conv2()	66
6.15	matfit.c File Reference	67
6.15.1	Function Documentation	67
6.15.1.1	mat_least_squares()	67
6.15.1.2	mat_linear_ls_fit()	68
6.15.1.3	mat_rob_least_squares()	68
6.15.1.4	mat_robust_fit()	68
6.15.1.5	mat_w_least_squares()	69
6.16	matflip.c File Reference	69
6.16.1	Function Documentation	70

6.16.1.1	<code>mat_fliplr()</code>	70
6.16.1.2	<code>mat_flipud()</code>	70
6.17	<code>matfuncs.c</code> File Reference	70
6.17.1	Function Documentation	71
6.17.1.1	<code>__mat_addfunc()</code>	71
6.17.1.2	<code>__mat_arccosh()</code>	71
6.17.1.3	<code>__mat_arcsinh()</code>	72
6.17.1.4	<code>__mat_arctanh()</code>	72
6.17.1.5	<code>__mat_bisquare_wt()</code>	72
6.17.1.6	<code>__mat_divfunc()</code>	73
6.17.1.7	<code>__mat_huber_wt()</code>	73
6.17.1.8	<code>__mat_logplusone()</code>	74
6.17.1.9	<code>__mat_mulfunc()</code>	74
6.17.1.10	<code>__mat_round_away_zero()</code>	74
6.17.1.11	<code>__mat_round_towards_zero()</code>	75
6.17.1.12	<code>__mat_sqrfunc()</code>	75
6.17.1.13	<code>__mat_sqrtfunc()</code>	75
6.17.1.14	<code>__mat_subfunc()</code>	76
6.17.1.15	<code>mat_bisquare_wt()</code>	76
6.17.1.16	<code>mat_gfunc()</code>	77
6.17.1.17	<code>mat_huber_wt()</code>	77
6.18	<code>matgraph.c</code> File Reference	77
6.18.1	Function Documentation	78
6.18.1.1	<code>mat_graph_adjlist()</code>	78
6.18.1.2	<code>mat_graph_adjm_to_adjl()</code>	78
6.18.1.3	<code>mat_graph_creat()</code>	78
6.18.1.4	<code>mat_graph_dump()</code>	78
6.18.1.5	<code>mat_graph_dumpf()</code>	79
6.18.1.6	<code>mat_graph_reverse()</code>	79
6.18.1.7	<code>mat_graph_search()</code>	79

6.18.1.8	<code>mat_graph_visit()</code>	79
6.19	<code>matinnerprod.c</code> File Reference	79
6.19.1	Function Documentation	80
6.19.1.1	<code>mat_innerprod()</code>	80
6.19.1.2	<code>mat_norm_inf()</code>	80
6.19.1.3	<code>mat_norm_one()</code>	80
6.19.1.4	<code>mat_norm_p()</code>	80
6.20	<code>matintegrate.c</code> File Reference	80
6.20.1	Function Documentation	81
6.20.1.1	<code>mat_int_qadrat()</code>	81
6.20.1.2	<code>mat_int_simpson()</code>	82
6.20.1.3	<code>mat_int_trapezoid()</code>	82
6.21	<code>matinv.c</code> File Reference	83
6.21.1	Function Documentation	83
6.21.1.1	<code>mat_inv()</code>	83
6.21.1.2	<code>mat_reg_inv()</code>	83
6.22	<code>matkdtree.c</code> File Reference	84
6.22.1	Function Documentation	84
6.22.1.1	<code>mat_kdtree_free()</code>	84
6.22.1.2	<code>mat_kdtree_k_nearest()</code>	84
6.22.1.3	<code>mat_kdtree_make_tree()</code>	85
6.22.1.4	<code>mat_kdtree_nearest()</code>	85
6.23	<code>matmaxmin.c</code> File Reference	86
6.23.1	Function Documentation	86
6.23.1.1	<code>mat_max()</code>	86
6.23.1.2	<code>mat_min()</code>	86
6.24	<code>matmds.c</code> File Reference	86
6.24.1	Function Documentation	86
6.24.1.1	<code>__mat_mds_metric()</code>	87
6.24.1.2	<code>__mat_mds_nonmetric()</code>	87

6.24.1.3	<code>mat_mds()</code>	87
6.25	<code>matmean.c</code> File Reference	87
6.25.1	Function Documentation	87
6.25.1.1	<code>int_vec_mean()</code>	87
6.25.1.2	<code>mat_mean()</code>	88
6.25.1.3	<code>mat_mean_col()</code>	88
6.25.1.4	<code>mat_mean_row()</code>	89
6.26	<code>matmisc.c</code> File Reference	89
6.26.1	Function Documentation	90
6.26.1.1	<code>__mat_cart2pol()</code>	90
6.26.1.2	<code>__mat_pol2cart()</code>	90
6.26.1.3	<code>int_vec_permute_vect()</code>	90
6.26.1.4	<code>int_vec_unique()</code>	91
6.26.1.5	<code>mat_bsxfun()</code>	91
6.26.1.6	<code>mat_calc_dist_sq()</code>	91
6.26.1.7	<code>mat_cart2pol()</code>	92
6.26.1.8	<code>mat_find_within_dist()</code>	92
6.26.1.9	<code>mat_fnextline()</code>	93
6.26.1.10	<code>mat_get_sub_matrix_from_cols()</code>	93
6.26.1.11	<code>mat_get_sub_matrix_from_rows()</code>	93
6.26.1.12	<code>mat_get_sub_vector()</code>	94
6.26.1.13	<code>mat_nextline()</code>	94
6.26.1.14	<code>mat_pick_col()</code>	94
6.26.1.15	<code>mat_pick_row()</code>	95
6.26.1.16	<code>mat_pol2cart()</code>	95
6.26.1.17	<code>mats_isinf()</code>	96
6.26.1.18	<code>mats_isnan()</code>	96
6.27	<code>matmul.c</code> File Reference	96
6.27.1	Function Documentation	97
6.27.1.1	<code>int_vec_mul()</code>	97

6.27.1.2	int_vec_muls()	97
6.27.1.3	mat_diagmul()	98
6.27.1.4	mat_mul()	98
6.27.1.5	mat_mul_dot()	98
6.27.1.6	mat_mul_fast()	100
6.27.1.7	mat_muls()	100
6.28	matpca.c File Reference	101
6.28.1	Function Documentation	101
6.28.1.1	mat_corcol()	101
6.28.1.2	mat_covcol()	101
6.28.1.3	mat_eig_sym()	101
6.28.1.4	mat_pca()	102
6.28.1.5	mat_scpcol()	102
6.28.1.6	mat_tqli()	102
6.28.1.7	mat_tred2()	102
6.29	matpinv.c File Reference	102
6.29.1	Function Documentation	102
6.29.1.1	mat_pinv()	102
6.29.1.2	mat_wpinv()	103
6.30	matpoly.c File Reference	103
6.30.1	Function Documentation	104
6.30.1.1	mat_binom()	104
6.30.1.2	mat_binom_init()	105
6.30.1.3	mat_cheby()	105
6.30.1.4	mat_cheby_approx()	105
6.30.1.5	mat_cheby_coeffs_to_poly()	106
6.30.1.6	mat_cheby_init()	106
6.30.1.7	mat_legendre()	106
6.30.1.8	mat_legendre_init()	107
6.30.1.9	mat_poly_add()	107

6.30.1.10	<code>mat_poly_diff()</code>	107
6.30.1.11	<code>mat_poly_diff_eval()</code>	108
6.30.1.12	<code>mat_poly_div()</code>	108
6.30.1.13	<code>mat_poly_eval()</code>	108
6.30.1.14	<code>mat_poly_mul()</code>	110
6.30.1.15	<code>mat_poly_scale()</code>	110
6.30.1.16	<code>mat_poly_shift()</code>	111
6.30.2	Variable Documentation	111
6.30.2.1	<code>mat_binom_series_table</code>	111
6.30.2.2	<code>mat_cheby_series_table</code>	111
6.30.2.3	<code>mat_legendre_series_table</code>	111
6.31	<code>matprec.c</code> File Reference	112
6.31.1	Function Documentation	112
6.31.1.1	<code>mat_bayes_classifier_test()</code>	112
6.31.1.2	<code>mat_bayes_classifier_train()</code>	112
6.31.1.3	<code>mat_kmeans()</code>	112
6.31.1.4	<code>mat_perceptron_test()</code>	112
6.31.1.5	<code>mat_perceptron_train()</code>	113
6.31.1.6	<code>mat_perceptron_train_()</code>	113
6.32	<code>matpursuit.c</code> File Reference	113
6.32.1	Function Documentation	113
6.32.1.1	<code>mat_omp()</code>	113
6.33	<code>matrand.c</code> File Reference	114
6.33.1	Function Documentation	114
6.33.1.1	<code>__mat_rand()</code>	114
6.33.1.2	<code>__mat_randexp()</code>	114
6.33.1.3	<code>__mat_randfun()</code>	114
6.33.1.4	<code>__mat_randn()</code>	115
6.33.1.5	<code>int_vec_randperm()</code>	115
6.33.1.6	<code>mat_rand()</code>	115

6.33.1.7	mat_randexp()	115
6.33.1.8	mat_randfun()	115
6.33.1.9	mat_randn()	116
6.33.1.10	mat_randperm()	116
6.33.1.11	mat_randperm_n()	116
6.33.1.12	mat_set_seed()	116
6.33.2	Variable Documentation	116
6.33.2.1	MAT_SEED	116
6.33.2.2	MAT_SET_SEED	116
6.34	matrix.c File Reference	117
6.35	matrix.h File Reference	117
6.35.1	Typedef Documentation	128
6.35.1.1	INT_VECSTACK	128
6.35.1.2	INT_VECTOR	128
6.35.1.3	mat_bayes_model	128
6.35.1.4	MAT_BAYES_MODEL	129
6.35.1.5	mat_gnode	129
6.35.1.6	MAT_GNODE	129
6.35.1.7	mat_graph	129
6.35.1.8	MAT_GRAPH	129
6.35.1.9	mat_int_priorityqueue	129
6.35.1.10	MAT_INT_PRIORITYQUEUE	129
6.35.1.11	mat_int_queue	130
6.35.1.12	MAT_INT_QUEUE	130
6.35.1.13	mat_int_stack	130
6.35.1.14	MAT_INT_STACK	130
6.35.1.15	mat_intpqnode	130
6.35.1.16	MAT_INTPQNODE	130
6.35.1.17	mat_kdnode	130
6.35.1.18	MAT_KDNODE	131

6.35.1.19	<code>mat_kdtree</code>	131
6.35.1.20	<code>MAT_KDTREE</code>	131
6.35.1.21	<code>mat_mtype_priorityqueue</code>	131
6.35.1.22	<code>MAT_MTYPE_PRIORITYQUEUE</code>	131
6.35.1.23	<code>mat_mtype_queue</code>	131
6.35.1.24	<code>MAT_MTYPE_QUEUE</code>	131
6.35.1.25	<code>mat_mtype_stack</code>	132
6.35.1.26	<code>MAT_MTYPE_STACK</code>	132
6.35.1.27	<code>mat_mtypepqnode</code>	132
6.35.1.28	<code>MAT_MTYPEPQNODE</code>	132
6.35.1.29	<code>mat_perceptron</code>	132
6.35.1.30	<code>MAT_PERCEPTRON</code>	132
6.35.1.31	<code>mat_qintnode</code>	132
6.35.1.32	<code>MAT_QINTNODE</code>	133
6.35.1.33	<code>mat_qmtyphenode</code>	133
6.35.1.34	<code>MAT_QMTYPENODE</code>	133
6.35.1.35	<code>MAT_TREE</code>	133
6.35.1.36	<code>mat_tree_node</code>	133
6.35.1.37	<code>MAT_TREE_NODE</code>	133
6.35.1.38	<code>MATRIX</code>	133
6.35.1.39	<code>MATSTACK</code>	134
6.35.1.40	<code>MATVEC_DPOINTER</code>	134
6.35.2	Function Documentation	134
6.35.2.1	<code>__declspec()</code>	134
6.35.2.2	<code>__int_vec_creat()</code>	134
6.35.2.3	<code>__int_vecstack_creat()</code>	134
6.35.2.4	<code>__mat_addfunc()</code>	134
6.35.2.5	<code>__mat_arccosh()</code>	135
6.35.2.6	<code>__mat_arcsinh()</code>	135
6.35.2.7	<code>__mat_arctanh()</code>	135

6.35.2.8	__mat_bisquare_wt()	136
6.35.2.9	__mat_cart2pol()	136
6.35.2.10	__mat_creat()	136
6.35.2.11	__mat_divfunc()	137
6.35.2.12	__mat_fft()	137
6.35.2.13	__mat_huber_wt()	137
6.35.2.14	__mat_kd_find_median()	138
6.35.2.15	__mat_kdtree_k_nearest()	138
6.35.2.16	__mat_kdtree_make_tree()	138
6.35.2.17	__mat_kdtree_nearest()	138
6.35.2.18	__mat_lint()	139
6.35.2.19	__mat_logplusone()	139
6.35.2.20	__mat_mds_metric()	139
6.35.2.21	__mat_mds_nonmetric()	139
6.35.2.22	__mat_mulfunc()	140
6.35.2.23	__mat_pol2cart()	140
6.35.2.24	__mat_powerof2()	140
6.35.2.25	__mat_quicksort()	140
6.35.2.26	__mat_rand()	141
6.35.2.27	__mat_randexp()	141
6.35.2.28	__mat_randfun()	141
6.35.2.29	__mat_randn()	141
6.35.2.30	__mat_round_away_zero()	141
6.35.2.31	__mat_round_towards_zero()	142
6.35.2.32	__mat_sqrfunc()	142
6.35.2.33	__mat_sqrtfunc()	142
6.35.2.34	__mat_subfunc()	143
6.35.2.35	__matstack_creat()	143
6.35.2.36	gen_eq()	143
6.35.2.37	gen_error()	143

6.35.2.38 <code>gen_gt()</code>	144
6.35.2.39 <code>gen_lt()</code>	144
6.35.2.40 <code>graph_error()</code>	144
6.35.2.41 <code>int_vec2_mat()</code>	145
6.35.2.42 <code>int_vec_abs()</code>	145
6.35.2.43 <code>int_vec_add()</code>	145
6.35.2.44 <code>int_vec_adds()</code>	146
6.35.2.45 <code>int_vec_append()</code>	146
6.35.2.46 <code>int_vec_concat()</code>	147
6.35.2.47 <code>int_vec_copy()</code>	147
6.35.2.48 <code>int_vec_creat()</code>	148
6.35.2.49 <code>int_vec_div()</code>	148
6.35.2.50 <code>int_vec_divs()</code>	148
6.35.2.51 <code>int_vec_divs_inv()</code>	149
6.35.2.52 <code>int_vec_dump()</code>	149
6.35.2.53 <code>int_vec_dumpf()</code>	150
6.35.2.54 <code>int_vec_error()</code>	150
6.35.2.55 <code>int_vec_fdump()</code>	150
6.35.2.56 <code>int_vec_fdumpf()</code>	150
6.35.2.57 <code>int_vec_fill()</code>	151
6.35.2.58 <code>int_vec_fill_type()</code>	151
6.35.2.59 <code>int_vec_find()</code>	152
6.35.2.60 <code>int_vec_free()</code>	152
6.35.2.61 <code>int_vec_inv()</code>	152
6.35.2.62 <code>int_vec_mean()</code>	152
6.35.2.63 <code>int_vec_mul()</code>	153
6.35.2.64 <code>int_vec_muls()</code>	153
6.35.2.65 <code>int_vec_permute_vect()</code>	154
6.35.2.66 <code>int_vec_randperm()</code>	154
6.35.2.67 <code>int_vec_sub()</code>	154

6.35.2.68 <code>int_vec_subs()</code>	155
6.35.2.69 <code>int_vec_subs_neg()</code>	155
6.35.2.70 <code>int_vec_sum()</code>	156
6.35.2.71 <code>int_vec_unique()</code>	156
6.35.2.72 <code>int_vecstack_creat()</code>	156
6.35.2.73 <code>int_vecstack_error()</code>	157
6.35.2.74 <code>int_vecstack_free()</code>	157
6.35.2.75 <code>mat_2int_vec()</code>	157
6.35.2.76 <code>mat_abs()</code>	158
6.35.2.77 <code>mat_add()</code>	158
6.35.2.78 <code>mat_adds()</code>	158
6.35.2.79 <code>mat_backsubs1()</code>	159
6.35.2.80 <code>mat_bayes_classifier_test()</code>	159
6.35.2.81 <code>mat_bayes_classifier_train()</code>	159
6.35.2.82 <code>mat_bayes_model_creat()</code>	159
6.35.2.83 <code>mat_bayes_model_free()</code>	160
6.35.2.84 <code>mat_binom()</code>	160
6.35.2.85 <code>mat_binom_init()</code>	160
6.35.2.86 <code>mat_bisquare_wt()</code>	160
6.35.2.87 <code>mat_bs_delete()</code>	161
6.35.2.88 <code>mat_bs_find()</code>	161
6.35.2.89 <code>mat_bs_find_max()</code>	161
6.35.2.90 <code>mat_bs_find_min()</code>	161
6.35.2.91 <code>mat_bs_free()</code>	161
6.35.2.92 <code>mat_bs_inorder()</code>	162
6.35.2.93 <code>mat_bs_insert()</code>	162
6.35.2.94 <code>mat_bs_make_null()</code>	162
6.35.2.95 <code>mat_bsxfun()</code>	162
6.35.2.96 <code>mat_calc_dist_sq()</code>	163
6.35.2.97 <code>mat_cart2pol()</code>	163

6.35.2.98 mat_cheby()	163
6.35.2.99 mat_cheby_approx()	164
6.35.2.100 mat_cheby_coefs_to_poly()	164
6.35.2.101 mat_cheby_init()	165
6.35.2.102 mat_cholesky()	165
6.35.2.103 mat_cofact()	165
6.35.2.104 mat_colcopy()	166
6.35.2.105 mat_concat()	166
6.35.2.106 mat_conjgrad()	167
6.35.2.107 mat_conv2()	167
6.35.2.108 mat_copy()	167
6.35.2.109 mat_corcol()	168
6.35.2.110 mat_count_words_in_line()	168
6.35.2.111 mat_covcol()	168
6.35.2.112 mat_creat()	169
6.35.2.113 mat_creat_diag()	169
6.35.2.114 mat_det()	169
6.35.2.115 mat_diagmul()	170
6.35.2.116 mat_div_dot()	170
6.35.2.117 mat_divs()	171
6.35.2.118 mat_divs_inv()	171
6.35.2.119 mat_dlmread()	171
6.35.2.120 mat_dlmwrite()	172
6.35.2.121 mat_dump()	172
6.35.2.122 mat_dumpf()	172
6.35.2.123 mat_durbin()	173
6.35.2.124 mat_eig_sym()	173
6.35.2.125 mat_error()	173
6.35.2.126 mat_fdump()	174
6.35.2.127 mat_fdumpf()	174

6.35.2.128	mat_fft2()	174
6.35.2.129	mat_fgetmat()	175
6.35.2.130	mat_fill()	175
6.35.2.131	mat_fill_type()	175
6.35.2.132	mat_find()	176
6.35.2.133	mat_find_within_dist()	176
6.35.2.134	mat_fliplr()	176
6.35.2.135	mat_flipud()	177
6.35.2.136	mat_fnextline()	177
6.35.2.137	mat_free()	177
6.35.2.138	mat_get_sub_matrix_from_cols()	177
6.35.2.139	mat_get_sub_matrix_from_rows()	178
6.35.2.140	mat_get_sub_vector()	178
6.35.2.141	mat_gfunc()	179
6.35.2.142	mat_go_next_word()	179
6.35.2.143	mat_graph_adjlist()	179
6.35.2.144	mat_graph_adjm_to_adjl()	180
6.35.2.145	mat_graph_creat()	180
6.35.2.146	mat_graph_dump()	180
6.35.2.147	mat_graph_dumpf()	180
6.35.2.148	mat_graph_reverse()	180
6.35.2.149	mat_graph_search()	180
6.35.2.150	mat_graph_visit()	181
6.35.2.151	mat_huber_wt()	181
6.35.2.152	mat_innerprod()	181
6.35.2.153	mat_int_priorityqueue_creat()	181
6.35.2.154	mat_int_priorityqueue_dequeue()	182
6.35.2.155	mat_int_priorityqueue_enqueue()	182
6.35.2.156	mat_int_priorityqueue_free()	182
6.35.2.157	mat_int_priorityqueue_is_empty()	182

6.35.2.158	<code>mat_int_priorityqueue_update()</code>	182
6.35.2.159	<code>mat_int_qadtrat()</code>	182
6.35.2.160	<code>mat_int_queue_creat()</code>	183
6.35.2.161	<code>mat_int_queue_dequeue()</code>	183
6.35.2.162	<code>mat_int_queue_enqueue()</code>	183
6.35.2.163	<code>mat_int_queue_free()</code>	183
6.35.2.164	<code>mat_int_queue_is_empty()</code>	183
6.35.2.165	<code>mat_int_simpson()</code>	183
6.35.2.166	<code>mat_int_stack_creat()</code>	184
6.35.2.167	<code>mat_int_stack_free()</code>	184
6.35.2.168	<code>mat_int_stack_is_empty()</code>	184
6.35.2.169	<code>mat_int_stack_pop()</code>	184
6.35.2.170	<code>mat_int_stack_push()</code>	184
6.35.2.171	<code>mat_int_trapezoid()</code>	184
6.35.2.172	<code>mat_inv()</code>	185
6.35.2.173	<code>mat_inv_dot()</code>	185
6.35.2.174	<code>mat_isnumeric()</code>	186
6.35.2.175	<code>mat_kdtree_free()</code>	186
6.35.2.176	<code>mat_kdtree_k_nearest()</code>	186
6.35.2.177	<code>mat_kdtree_make_tree()</code>	187
6.35.2.178	<code>mat_kdtree_nearest()</code>	187
6.35.2.179	<code>mat_kmeans()</code>	187
6.35.2.180	<code>mat_least_squares()</code>	188
6.35.2.181	<code>mat_legendre()</code>	188
6.35.2.182	<code>mat_legendre_init()</code>	188
6.35.2.183	<code>mat_linear_ls_fit()</code>	189
6.35.2.184	<code>mat_lsolve()</code>	189
6.35.2.185	<code>mat_lsolve_durbin()</code>	189
6.35.2.186	<code>mat_lu()</code>	190
6.35.2.187	<code>mat_max()</code>	190

6.35.2.188	mat_mds()	191
6.35.2.189	mat_mean()	191
6.35.2.190	mat_mean_col()	191
6.35.2.191	mat_mean_row()	191
6.35.2.192	mat_median()	192
6.35.2.193	mat_min()	192
6.35.2.194	mat_minor()	192
6.35.2.195	mat_mtype_priorityqueue_creat()	193
6.35.2.196	mat_mtype_priorityqueue_dequeue()	193
6.35.2.197	mat_mtype_priorityqueue_enqueue()	193
6.35.2.198	mat_mtype_priorityqueue_free()	193
6.35.2.199	mat_mtype_priorityqueue_is_empty()	193
6.35.2.200	mat_mtype_priorityqueue_update()	194
6.35.2.201	mat_mtype_queue_creat()	194
6.35.2.202	mat_mtype_queue_dequeue()	194
6.35.2.203	mat_mtype_queue_enqueue()	194
6.35.2.204	mat_mtype_queue_free()	194
6.35.2.205	mat_mtype_queue_is_empty()	194
6.35.2.206	mat_mtype_stack_creat()	195
6.35.2.207	mat_mtype_stack_free()	195
6.35.2.208	mat_mtype_stack_is_empty()	195
6.35.2.209	mat_mtype_stack_pop()	195
6.35.2.210	mat_mtype_stack_push()	195
6.35.2.211	mat_mul()	195
6.35.2.212	mat_mul_dot()	196
6.35.2.213	mat_mul_fast()	196
6.35.2.214	mat_muls()	197
6.35.2.215	mat_nextline()	197
6.35.2.216	mat_norm_inf()	197
6.35.2.217	mat_norm_one()	197

6.35.2.218	<code>mat_norm_p()</code>	197
6.35.2.219	<code>mat_omp()</code>	198
6.35.2.220	<code>mat_order_statistic()</code>	198
6.35.2.221	<code>mat_pca()</code>	198
6.35.2.222	<code>mat_perceptron_creat()</code>	198
6.35.2.223	<code>mat_perceptron_free()</code>	199
6.35.2.224	<code>mat_perceptron_test()</code>	200
6.35.2.225	<code>mat_perceptron_train()</code>	200
6.35.2.226	<code>mat_perceptron_train_()</code>	200
6.35.2.227	<code>mat_pick_col()</code>	200
6.35.2.228	<code>mat_pick_row()</code>	201
6.35.2.229	<code>mat_pinv()</code>	201
6.35.2.230	<code>mat_pol2cart()</code>	201
6.35.2.231	<code>mat_poly_add()</code>	202
6.35.2.232	<code>mat_poly_diff()</code>	202
6.35.2.233	<code>mat_poly_diff_eval()</code>	203
6.35.2.234	<code>mat_poly_div()</code>	203
6.35.2.235	<code>mat_poly_eval()</code>	204
6.35.2.236	<code>mat_poly_mul()</code>	204
6.35.2.237	<code>mat_poly_scale()</code>	204
6.35.2.238	<code>mat_poly_shift()</code>	205
6.35.2.239	<code>mat_qr()</code>	205
6.35.2.240	<code>mat_qsort()</code>	206
6.35.2.241	<code>mat_rand()</code>	206
6.35.2.242	<code>mat_randexp()</code>	206
6.35.2.243	<code>mat_randfun()</code>	207
6.35.2.244	<code>mat_randn()</code>	207
6.35.2.245	<code>mat_randperm()</code>	207
6.35.2.246	<code>mat_randperm_n()</code>	207
6.35.2.247	<code>mat_read_word()</code>	207

6.35.2.248	<code>mat_reg_inv()</code>	208
6.35.2.249	<code>mat_rob_least_squares()</code>	208
6.35.2.250	<code>mat_robust_fit()</code>	209
6.35.2.251	<code>mat_rowcopy()</code>	209
6.35.2.252	<code>mat_scpcol()</code>	210
6.35.2.253	<code>mat_set_seed()</code>	210
6.35.2.254	<code>mat_sub()</code>	210
6.35.2.255	<code>mat_submat()</code>	210
6.35.2.256	<code>mat_subs()</code>	211
6.35.2.257	<code>mat_subs_neg()</code>	211
6.35.2.258	<code>mat_sum()</code>	212
6.35.2.259	<code>mat_sum_col()</code>	212
6.35.2.260	<code>mat_sum_row()</code>	212
6.35.2.261	<code>mat_svd()</code>	213
6.35.2.262	<code>mat_symtoeplz()</code>	213
6.35.2.263	<code>mat_tic()</code>	213
6.35.2.264	<code>mat_toc()</code>	214
6.35.2.265	<code>mat_toc_print()</code>	214
6.35.2.266	<code>mat_tqli()</code>	214
6.35.2.267	<code>mat_tran()</code>	214
6.35.2.268	<code>mat_tred2()</code>	215
6.35.2.269	<code>mat_vectorize()</code>	215
6.35.2.270	<code>mat_vectorize_tr()</code>	215
6.35.2.271	<code>mat_w_least_squares()</code>	216
6.35.2.272	<code>mat_wpinv()</code>	216
6.35.2.273	<code>mat_xcopy()</code>	216
6.35.2.274	<code>mat_xjoin()</code>	217
6.35.2.275	<code>mats_isinf()</code>	217
6.35.2.276	<code>mats_isnan()</code>	218
6.35.2.277	<code>matstack_append()</code>	218

6.35.2.278	matstack_creat()	219
6.35.2.279	matstack_error()	219
6.35.2.280	matstack_free()	219
6.35.2.281	matvec_creat()	220
6.35.2.282	matvec_free()	220
6.35.2.283	pq_error()	220
6.35.2.284	queue_error()	220
6.35.2.285	stack_error()	221
6.35.3	Variable Documentation	221
6.35.3.1	mat_binom_series_table	221
6.35.3.2	mat_cheby_series_table	221
6.35.3.3	MAT_CLOCK_TIME	221
6.35.3.4	mat_legendre_series_table	221
6.35.3.5	MAT_SEED	222
6.35.3.6	MAT_SET_SEED	222
6.36	matsearch.c File Reference	222
6.36.1	Function Documentation	222
6.36.1.1	int_vec_find()	222
6.36.1.2	mat_find()	222
6.37	matsolve.c File Reference	223
6.37.1	Function Documentation	223
6.37.1.1	mat_backsubs1()	223
6.37.1.2	mat_cholesky()	223
6.37.1.3	mat_conjgrad()	224
6.37.1.4	mat_lsolve()	224
6.37.1.5	mat_lu()	224
6.38	matsort.c File Reference	225
6.38.1	Function Documentation	225
6.38.1.1	mat_median()	225
6.38.1.2	mat_order_statistic()	226

6.38.1.3	<code>mat_qsort()</code>	226
6.39	<code>matstdrels.c</code> File Reference	226
6.39.1	Function Documentation	227
6.39.1.1	<code>gen_eq()</code>	227
6.39.1.2	<code>gen_gt()</code>	227
6.39.1.3	<code>gen_lt()</code>	227
6.40	<code>matsub.c</code> File Reference	228
6.40.1	Function Documentation	228
6.40.1.1	<code>int_vec_sub()</code>	228
6.40.1.2	<code>int_vec_subs()</code>	229
6.40.1.3	<code>int_vec_subs_neg()</code>	229
6.40.1.4	<code>mat_sub()</code>	229
6.40.1.5	<code>mat_subs()</code>	230
6.40.1.6	<code>mat_subs_neg()</code>	230
6.41	<code>matsubx.c</code> File Reference	231
6.41.1	Function Documentation	231
6.41.1.1	<code>mat_submat()</code>	231
6.42	<code>matsum.c</code> File Reference	231
6.42.1	Function Documentation	232
6.42.1.1	<code>int_vec_sum()</code>	232
6.42.1.2	<code>mat_sum()</code>	232
6.42.1.3	<code>mat_sum_col()</code>	233
6.42.1.4	<code>mat_sum_row()</code>	233
6.43	<code>matsvd.c</code> File Reference	233
6.43.1	Function Documentation	233
6.43.1.1	<code>mat_svd()</code>	234
6.44	<code>mattext.c</code> File Reference	234
6.44.1	Function Documentation	234
6.44.1.1	<code>mat_count_words_in_line()</code>	234
6.44.1.2	<code>mat_dlmread()</code>	235

6.44.1.3	<code>mat_dlmwrite()</code>	235
6.44.1.4	<code>mat_go_next_word()</code>	235
6.44.1.5	<code>mat_isnumeric()</code>	237
6.44.1.6	<code>mat_read_word()</code>	237
6.45	<code>mattimers.c</code> File Reference	238
6.45.1	Function Documentation	238
6.45.1.1	<code>__declspec()</code>	238
6.45.1.2	<code>mat_toc()</code>	238
6.45.1.3	<code>mat_toc_print()</code>	238
6.45.2	Variable Documentation	239
6.45.2.1	<code>MAT_CLOCK_TIME</code>	239
6.46	<code>mattoepz.c</code> File Reference	239
6.46.1	Function Documentation	239
6.46.1.1	<code>mat_symtoeplz()</code>	239
6.47	<code>mattran.c</code> File Reference	239
6.47.1	Function Documentation	240
6.47.1.1	<code>mat_tran()</code>	240
6.48	<code>README.md</code> File Reference	240
Index		241

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

![[alt tag]](https://raw.githubusercontent.com/mohammadul/matrixlab/master/matrixlab.ico)
Matrixlab

A C Matrix Library Originally adapted from Small Matrix Toolbox for C programmers, ver. 0.4 by Patrick Ko Shu-pui

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

** For more details and updates, visit <http://mohammadulhaque.alotspace.com>.

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

mat_bayes_model	Bayes Classifier Model Structure	9
mat_gnode	Graph Node Structure	10
mat_graph	Graph Structure	11
mat_int_priorityqueue	Integer Priority Queue Structure	13
mat_int_queue	Integer Queue Structure	14
mat_int_stack	Integer Stack Structure	15
mat_intpqnode	Integer Priority Queue Node Structure	16
mat_kdnode	17
mat_kdtree	18
mat_mtype_priorityqueue	Mtype Priority Queue Structure	19
mat_mtype_queue	Mtype Queue Structure	20
mat_mtype_stack	Mtype Stack Structure	21
mat_mtypepqnode	Mtype Priority Queue Node Structure	22
mat_perceptron	Perceptron Classifier Model Structure	23
mat_qintnode	Integer Queue Node Structure	24
mat_qmtypenode	Mtype Queue Node Structure	25
mat_tree_node	Search Tree Node Structure	26

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

matabs.c	27
matadd.c	28
matcompress.c	30
matconcat.c	30
matconv.c	31
matcreat.c	33
matdatastruct.c	45
matdet.c	52
matdiv.c	53
matdump.c	57
matdurn.c	60
materr.c	62
matfft.c	65
matfilter.c	66
matfit.c	67
matflip.c	69
matfuncs.c	70
matgraph.c	77
matinnerprod.c	79
matintegrate.c	80
matinv.c	83
matkdtree.c	84
matmaxmin.c	86
matmds.c	86
matmean.c	87
matmisc.c	89
matmul.c	96
matpca.c	101
matpinv.c	102
matpoly.c	103
matprec.c	112
matpursuit.c	113
matrand.c	114
matrix.c	117
matrix.h	117

matsearch.c	222
matsolve.c	223
matsort.c	225
matstdrels.c	226
matsub.c	228
matsubx.c	231
matsum.c	231
matsvd.c	233
mattext.c	234
mattimers.c	238
mattoepz.c	239
mattran.c	239

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 class_covars

[MATSTACK](#) `mat_bayes_model::class_covars`

Training data class covariances

5.1.2.2 class_labels

`INT_VECTOR` `mat_bayes_model::class_labels`

Training data class label vector

5.1.2.3 class_means

`MATSTACK` `mat_bayes_model::class_means`

Training data class means

5.1.2.4 class_priors

`MATRIX` `mat_bayes_model::class_priors`

Training data prior information

5.1.2.5 num_of_classes

`int` `mat_bayes_model::num_of_classes`

Number of training class

5.1.2.6 num_of_features

`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>
```

Collaboration diagram for `mat_gnode`:

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 next

```
struct mat_gnode* mat_gnode::next
```

Pointer to next node

5.2.2.2 v

```
int mat_gnode::v
```

Value

5.2.2.3 weight

```
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>
```

Collaboration diagram for mat_graph:

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 adj

`MAT_GNODE*` `mat_graph::adj`

5.3.2.2 dad

`int*` `mat_graph::dad`

5.3.2.3 id

`int` `mat_graph::id`

5.3.2.4 nedges

`int` `mat_graph::nedges`

Number of edges

5.3.2.5 nvertices

`int` `mat_graph::nvertices`

Number of vertices

5.3.2.6 pq

`MAT_INT_PRIORITYQUEUE` `mat_graph::pq`

5.3.2.7 val

```
int* mat_graph::val
```

5.3.2.8 vseq

```
int* mat_graph::vseq
```

5.3.2.9 weighted

```
int mat_graph::weighted
```

5.3.2.10 z

```
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>
```

Collaboration diagram for mat_int_priorityqueue:

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 element

```
MAT_INTPQNODE mat_int_priorityqueue::element
```

Pointer to priority queue data

5.4.2.2 length

```
int mat_int_priorityqueue::length
```

Total allocated priority queue length

5.4.2.3 p

```
int mat_int_priorityqueue::p
```

Current priority queue position

5.4.2.4 type

```
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>
```

Collaboration diagram for mat_int_queue:

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 head

[MAT_QINTNODE](#) mat_int_queue::head

Queue head node

5.5.2.2 p

int mat_int_queue::p

Current queue position

5.5.2.3 tail

[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 length

```
int mat_int_stack::length
```

Total allocated stack length

5.6.2.2 p

```
int mat_int_stack::p
```

Current stack position

5.6.2.3 stack

```
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 data

```
int mat_intpqnode::data
```

Integer node data

5.7.2.2 priority

```
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>
```

Collaboration diagram for mat_kdnode:

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 idx

```
int mat_kdnode::idx
```

5.8.1.2 left

```
struct mat\_kdnode* mat_kdnode::left
```

5.8.1.3 right

```
struct mat\_kdnode * mat_kdnode::right
```

5.8.1.4 x

```
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>
```

Collaboration diagram for mat_kdtree:

Data Fields

- int [ndims](#)
- int [length](#)
- int [_is_allocated](#)
- [MAT_KDNODE](#) data
- [MAT_KDNODE](#) kdtree

5.9.1 Field Documentation

5.9.1.1 _is_allocated

```
int mat_kdtree::_is_allocated
```

5.9.1.2 data

```
MAT\_KDNODE mat_kdtree::data
```


5.9.1.3 kdtree

[MAT_KDNODE](#) mat_kdtree::kdtree

5.9.1.4 length

int mat_kdtree::length

5.9.1.5 ndims

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>
```

Collaboration diagram for mat_mtype_priorityqueue:

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 element

[MAT_MTYPEPQNODE](#) mat_mtype_priorityqueue::element

Pointer to priority queue data

5.10.2.2 length

int mat_mtype_priorityqueue::length

Total allocated priority queue length

5.10.2.3 p

int mat_mtype_priorityqueue::p

Current priority queue position

5.10.2.4 type

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>
```

Collaboration diagram for mat_mtype_queue:

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 head

`MAT_QMTYPENODE mat_mtype_queue::head`

Queue head node

5.11.2.2 p

`int mat_mtype_queue::p`

Current queue position

5.11.2.3 tail

`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 length

```
int mat_mtype_stack::length
```

Total allocated stack length

5.12.2.2 p

```
int mat_mtype_stack::p
```

Current stack position

5.12.2.3 stack

```
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 data

```
mtype mat_mtypepqnode::data
```

Mtype node data

5.13.2.2 priority

```
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 class_labels

```
INT\_VECTOR mat_perceptron::class_labels
```

Training data class label vector

5.14.2.2 class_weights

```
MATRIX mat_perceptron::class_weights
```

Trained Classifier Weights

5.14.2.3 `istrained`

```
int mat_perceptron::istrained
```

Is trained

5.14.2.4 `num_of_classes`

```
int mat_perceptron::num_of_classes
```

Number of training classes

5.14.2.5 `num_of_features`

```
int mat_perceptron::num_of_features
```

Number of training features

5.14.2.6 `num_of_iterations`

```
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>
```

Collaboration diagram for `mat_qintnode`:

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 data

```
int mat_qintnode::data
```

Integer node data

5.15.2.2 next

```
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_qmtypenode Struct Reference

Mtype Queue Node Structure.

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

Collaboration diagram for mat_qmtypenode:

Data Fields

- mtype [data](#)
- struct [mat_qmtypenode](#) * [next](#)

5.16.1 Detailed Description

Mtype Queue Node Structure.

5.16.2 Field Documentation

5.16.2.1 data

```
mtype mat_qmtypenode::data
```

Mtype node data

5.16.2.2 next

```
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>
```

Collaboration diagram for mat_tree_node:

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 element

```
mtype mat_tree_node::element
```

Search tree node data

5.17.2.2 left

```
struct mat_tree_node* mat_tree_node::left
```

Pointer to left child node

5.17.2.3 right

```
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.
- `INT_VECTOR int_vec_abs (INT_VECTOR A, INT_VECTOR result)`
Computes absolute value of an integer vector.

6.1.1 Function Documentation

6.1.1.1 int_vec_abs()

```
INT_VECTOR int_vec_abs (  
    INT_VECTOR A,  
    INT_VECTOR result )
```

Computes absolute value of an integer vector.

Parameters

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

Returns

`abs(A)`

Here is the call graph for this function:

6.1.1.2 mat_abs()

```
MATRIX mat_abs (
    MATRIX A,
    MATRIX result )
```

Computes absolute value of matrix.

Parameters

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

Returns

$\text{abs}(\mathbf{A})$

Here is the call graph for this function: Here is the caller graph for this function:

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_vec_add()

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

Adds two integer vectors.

Parameters

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

Returns

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

Here is the call graph for this function:

6.2.1.2 int_vec_adds()

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

Adds an integer to an integer vector.

Parameters

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

Returns

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

Here is the call graph for this function:

6.2.1.3 mat_add()

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

Adds two matrices.

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

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

Here is the call graph for this function: Here is the caller graph for this function:

6.2.1.4 mat_adds()

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

Adds a scalar to a matrix.

Parameters

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

Returns

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

Here is the call graph for this function: Here is the caller graph for this function:

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_vec_concat()**

```
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>result</i>	Vector to store the result

Returns

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

Here is the call graph for this function:

6.4.1.2 mat_concat()

```
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

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

Here is the call graph for this function: Here is the caller graph for this function:

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 int_vec2_mat()

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

Converts an integer vector to a matrix.

Parameters

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

Returns

Output matrix

Here is the call graph for this function:

6.5.1.2 mat_2int_vec()

```
INT_VECTOR mat_2int_vec (
    MATRIX A )
```

Converts a matrix to an integer vector.

Parameters

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

Returns

Output vector

Here is the call graph for this function: Here is the caller graph for this function:

6.5.1.3 mat_vectorize()

```
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

$vec(\mathbf{A})$

Here is the call graph for this function:

6.5.1.4 mat_vectorize_tr()

```
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

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

Here is the call graph for this function: Here is the caller graph for this function:

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_vec_append()

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

Appends an integer to an integer vector.

Parameters

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

Returns

Appended vector

Here is the call graph for this function: Here is the caller graph for this function:

6.6.1.2 int_vec_copy()

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

Copies an integer vector.

Parameters

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

Returns

Output vector

Here is the call graph for this function:

6.6.1.3 int_vec_creat()

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

Creates an integer vector.

Parameters

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

Returns

Output vector

Here is the call graph for this function: Here is the caller graph for this function:

6.6.1.4 int_vec_fill()

```
INT_VECTOR int_vec_fill (
    INT_VECTOR A,
    int val )
```

Fills an integer vector with a value.

Parameters

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

Returns

Filled vector

6.6.1.5 int_vec_fill_type()

```
INT_VECTOR int_vec_fill_type (
    INT_VECTOR A,
    int type )
```

Fills an integer vector to a type.

Parameters

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

Returns

Filled vector

Here is the caller graph for this function:

6.6.1.6 int_vec_free()

```
int int_vec_free (
    INT_VECTOR A )
```

Frees an integer vector.

Parameters

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

Returns

Success

Here is the call graph for this function: Here is the caller graph for this function:

6.6.1.7 int_vecstack_creat()

```
INT_VECSTACK int_vecstack_creat (
    int len )
```

Creates an integer vector stack.

Parameters

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

Returns

Output vector stack

Here is the call graph for this function: Here is the caller graph for this function:

6.6.1.8 int_vecstack_free()

```
int int_vecstack_free (
    INT_VECSTACK A )
```

Frees an integer vector stack.

Parameters

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

Returns

Success

Here is the call graph for this function: Here is the caller graph for this function:

6.6.1.9 mat_bayes_model_creat()

```
MAT_BAYES_MODEL mat_bayes_model_creat (
    void )
```

Creates a Bayes model.

Returns

Output Bayes model

Here is the caller graph for this function:

6.6.1.10 mat_bayes_model_free()

```
int mat_bayes_model_free (
    MAT_BAYES_MODEL a )
```

Frees a Bayes model.

Parameters

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

Returns

Success

Here is the call graph for this function:

6.6.1.11 mat_colcopy()

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

Copies a column from a matrix.

Parameters

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

Returns

Copied matrix

Here is the caller graph for this function:

6.6.1.12 mat_copy()

```
MATRIX mat_copy (
    MATRIX A,
    MATRIX result )
```

Copies a matrix.

Parameters

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

Returns

Output matrix

Here is the call graph for this function: Here is the caller graph for this function:

6.6.1.13 mat_creat()

```
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

Here is the call graph for this function:

6.6.1.14 mat_creat_diag()

```
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

Here is the call graph for this function: Here is the caller graph for this function:

6.6.1.15 mat_fgetmat()

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

Gets matrix data from opened file.

Parameters

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

Returns

Number of elements copied

6.6.1.16 mat_fill()

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

Fills a matrix with a value.

Parameters

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

Returns

Filled matrix

Here is the caller graph for this function:

6.6.1.17 mat_fill_type()

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

Fills a matrix to a type.

Parameters

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

Returns

Filled matrix

Here is the caller graph for this function:

6.6.1.18 mat_free()

```
int mat_free (
    MATRIX A )
```

Frees a matrix.

Parameters

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

Returns

Success

Here is the call graph for this function: Here is the caller graph for this function:

6.6.1.19 mat_perceptron_creat()

```
MAT_PERCEPTRON mat_perceptron_creat (
    void )
```

Creates a perceptron.

Returns

Output perceptron

Here is the caller graph for this function:

6.6.1.20 mat_perceptron_free()

```
int mat_perceptron_free (
    MAT_PERCEPTRON a )
```

Frees a perceptron.

Parameters

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

Returns

Success

Here is the call graph for this function:

6.6.1.21 mat_rowcopy()

```
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 mat_xcopy()

```
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 $A_{s_i:e_i,s_j:e_j}$

Here is the call graph for this function: Here is the caller graph for this function:

6.6.1.23 mat_xjoin()

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

Here is the call graph for this function: Here is the caller graph for this function:

6.6.1.24 matstack_append()

```
MATSTACK matstack_append (
    MATSTACK s,
    MATRIX A )
```

Appends a matrix to a matrix stack.

Parameters

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

Returns

Output matrix stack

Here is the call graph for this function:

6.6.1.25 matstack_creat()

```
MATSTACK matstack_creat (
    int len )
```

Creates a matrix stack.

Parameters

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

Returns

Output matrix stack

Here is the call graph for this function: Here is the caller graph for this function:

6.6.1.26 matstack_free()

```
int matstack_free (
    MATSTACK A )
```

Frees a matrix stack.

Parameters

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

Returns

Success

Here is the call graph for this function: Here is the caller graph for this function:

6.6.1.27 matvec_creat()

```
MATVEC_DPOINTER matvec_creat (
    void )
```

Creates a matrix-vector pair.

Returns

Output matrix-vector pair

Here is the caller graph for this function:

6.6.1.28 matvec_free()

```
int matvec_free (
    MATVEC_DPOINTER a )
```

Frees a matrix-vector pair.

Parameters

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

Returns

Success

Here is the call graph for this function: Here is the caller graph for this function:

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_bs_delete()`

```
MAT_TREE mat_bs_delete (
    mtype x,
    MAT_TREE T )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.7.1.2 `mat_bs_find()`

```
MAT_TREE mat_bs_find (
    mtype x,
    MAT_TREE T )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.7.1.3 `mat_bs_find_max()`

```
MAT_TREE mat_bs_find_max (
    MAT_TREE T )
```

6.7.1.4 `mat_bs_find_min()`

```
MAT_TREE mat_bs_find_min (
    MAT_TREE T )
```

Here is the caller graph for this function:

6.7.1.5 `mat_bs_free()`

```
MAT_TREE mat_bs_free (
    MAT_TREE T )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.7.1.6 `mat_bs_inorder()`

```
int mat_bs_inorder (
    MAT_TREE T,
    int index,
    mtype ** p_ordered )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.7.1.7 mat_bs_insert()

```
MAT_TREE mat_bs_insert (
    mtype x,
    MAT_TREE T )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.7.1.8 mat_bs_make_null()

```
MAT_TREE mat_bs_make_null (
    void )
```

Here is the caller graph for this function:

6.7.1.9 mat_int_priorityqueue_creat()

```
MAT_INT_PRIORITYQUEUE mat_int_priorityqueue_creat (
    int type )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.7.1.10 mat_int_priorityqueue_dequeue()

```
mat_intpqnode mat_int_priorityqueue_dequeue (
    MAT_INT_PRIORITYQUEUE H )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.7.1.11 mat_int_priorityqueue_enqueue()

```
void mat_int_priorityqueue_enqueue (
    MAT_INT_PRIORITYQUEUE H,
    int data,
    int priority )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.7.1.12 mat_int_priorityqueue_free()

```
int mat_int_priorityqueue_free (
    MAT_INT_PRIORITYQUEUE H )
```

Here is the caller graph for this function:

6.7.1.13 mat_int_priorityqueue_is_empty()

```
int mat_int_priorityqueue_is_empty (
    MAT_INT_PRIORITYQUEUE H )
```

6.7.1.14 mat_int_priorityqueue_update()

```
int mat_int_priorityqueue_update (
    MAT_INT_PRIORITYQUEUE H,
    int data,
    int priority,
    int type )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.7.1.15 mat_int_queue_creat()

```
MAT_INT_QUEUE mat_int_queue_creat (
    void )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.7.1.16 mat_int_queue_dequeue()

```
int mat_int_queue_dequeue (
    MAT_INT_QUEUE s )
```

Here is the call graph for this function:

6.7.1.17 mat_int_queue_enqueue()

```
void mat_int_queue_enqueue (
    MAT_INT_QUEUE s,
    int value )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.7.1.18 mat_int_queue_free()

```
int mat_int_queue_free (
    MAT_INT_QUEUE s )
```

6.7.1.19 mat_int_queue_is_empty()

```
int mat_int_queue_is_empty (
    MAT_INT_QUEUE s )
```

6.7.1.20 mat_int_stack_creat()

```
MAT_INT_STACK mat_int_stack_creat (
    void )
```

Here is the call graph for this function:

6.7.1.21 mat_int_stack_free()

```
int mat_int_stack_free (
    MAT_INT_STACK s )
```

6.7.1.22 mat_int_stack_is_empty()

```
int mat_int_stack_is_empty (
    MAT_INT_STACK s )
```

6.7.1.23 mat_int_stack_pop()

```
int mat_int_stack_pop (
    MAT_INT_STACK s )
```

Here is the call graph for this function:

6.7.1.24 mat_int_stack_push()

```
void mat_int_stack_push (
    MAT_INT_STACK s,
    int value )
```

Here is the call graph for this function:

6.7.1.25 mat_mtype_priorityqueue_creat()

```
MAT_MTYPE_PRIORITYQUEUE mat_mtype_priorityqueue_creat (
    int type )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.7.1.26 mat_mtype_priorityqueue_dequeue()

```
mat_mtypepqnode mat_mtype_priorityqueue_dequeue (
    MAT_MTYPE_PRIORITYQUEUE H )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.7.1.27 mat_mtype_priorityqueue_enqueue()

```
void mat_mtype_priorityqueue_enqueue (
    MAT_MTYPE_PRIORITYQUEUE H,
    mtype data,
    mtype priority )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.7.1.28 mat_mtype_priorityqueue_free()

```
int mat_mtype_priorityqueue_free (
    MAT_MTYPE_PRIORITYQUEUE H )
```

Here is the caller graph for this function:

6.7.1.29 mat_mtype_priorityqueue_is_empty()

```
int mat_mtype_priorityqueue_is_empty (
    MAT_MTYPE_PRIORITYQUEUE H )
```

6.7.1.30 mat_mtype_priorityqueue_update()

```
int mat_mtype_priorityqueue_update (
    MAT_MTYPE_PRIORITYQUEUE H,
    mtype data,
    mtype priority,
    int type )
```

Here is the call graph for this function:

6.7.1.31 mat_mtype_queue_creat()

```
MAT_MTYPE_QUEUE mat_mtype_queue_creat (
    void )
```

Here is the call graph for this function:

6.7.1.32 mat_mtype_queue_dequeue()

```
mtype mat_mtype_queue_dequeue (
    MAT_MTYPE_QUEUE s )
```

Here is the call graph for this function:

6.7.1.33 mat_mtype_queue_enqueue()

```
void mat_mtype_queue_enqueue (
    MAT_MTYPE_QUEUE s,
    mtype value )
```

Here is the call graph for this function:

6.7.1.34 mat_mtype_queue_free()

```
int mat_mtype_queue_free (
    MAT_MTYPE_QUEUE s )
```

6.7.1.35 mat_mtype_queue_is_empty()

```
int mat_mtype_queue_is_empty (
    MAT_MTYPE_QUEUE s )
```

6.7.1.36 mat_mtype_stack_creat()

```
MAT_MTYPE_STACK mat_mtype_stack_creat (
    void )
```

Here is the call graph for this function:

6.7.1.37 mat_mtype_stack_free()

```
int mat_mtype_stack_free (
    MAT_MTYPE_STACK s )
```

6.7.1.38 mat_mtype_stack_is_empty()

```
int mat_mtype_stack_is_empty (
    MAT_MTYPE_STACK s )
```

6.7.1.39 mat_mtype_stack_pop()

```
mtype mat_mtype_stack_pop (
    MAT_MTYPE_STACK s )
```

Here is the call graph for this function:

6.7.1.40 mat_mtype_stack_push()

```
void mat_mtype_stack_push (
    MAT_MTYPE_STACK s,
    mtype value )
```

Here is the call graph for this function:

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 mat_cofact()

```
mtype mat_cofact (
    MATRIX A,
    int i,
    int j )
```

Computes a cofactor of a matrix.

Parameters

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

ReturnsCofactor C_{ij} **6.8.1.2 mat_det()**

```

mtype mat_det (
    MATRIX A )

```

Computes the determinant of a matrix.

Parameters

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

Returns $\det(A)$

Here is the call graph for this function: Here is the caller graph for this function:

6.8.1.3 mat_minor()

```

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

ReturnsMinor M_{ij}

Here is the call graph for this function:

6.9 matdiv.c File Reference**Functions**

- [MATRIX mat_inv_dot](#) (MATRIX A, MATRIX result)

Computes element-wise matrix inverse.

- `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.

- `MATRIX mat_divs_inv` (`MATRIX` A, `mtype` s, `MATRIX` result)

Divides a scalar by a matrix.

- `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.

- `INT_VECTOR int_vec_inv` (`INT_VECTOR` A, `INT_VECTOR` result)

Computes element-wise integer vector inverse.

- `INT_VECTOR int_vec_divs_inv` (`INT_VECTOR` A, `int` s, `INT_VECTOR` result)

Divides a scalar by an integer vector.

6.9.1 Function Documentation

6.9.1.1 `int_vec_div()`

```
INT_VECTOR int_vec_div (
    INT_VECTOR A,
    INT_VECTOR B,
    INT_VECTOR result )
```

Computes element-wise integer vector division.

Parameters

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

Returns

$A./B$

Here is the call graph for this function:

6.9.1.2 `int_vec_divs()`

```
INT_VECTOR int_vec_divs (
    INT_VECTOR A,
    int s,
    INT_VECTOR result )
```

Divides an integer vector by a scalar.

Parameters

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

Returns

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

Here is the call graph for this function:

6.9.1.3 int_vec_divs_inv()

```
INT_VECTOR int_vec_divs_inv (
    INT_VECTOR A,
    int s,
    INT_VECTOR result )
```

Divides a scalar by an integer vector.

Parameters

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

Returns

$$s1./A$$

Here is the call graph for this function:

6.9.1.4 int_vec_inv()

```
INT_VECTOR int_vec_inv (
    INT_VECTOR A,
    INT_VECTOR result )
```

Computes element-wise integer vector inverse.

Parameters

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

Returns

$$1./A$$

Here is the call graph for this function:

6.9.1.5 mat_div_dot()

```
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

$$A./B$$

Here is the call graph for this function:

6.9.1.6 mat_divs()

```
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

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

Here is the call graph for this function: Here is the caller graph for this function:

6.9.1.7 mat_divs_inv()

```
MATRIX mat_divs_inv (
    MATRIX A,
    mtype s,
    MATRIX result )
```

Divides a scalar by a matrix.

Parameters

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

Returns

$$s\mathbf{11}^T ./ \mathbf{A}$$

Here is the call graph for this function:

6.9.1.8 mat_inv_dot()

```
MATRIX mat_inv_dot (
    MATRIX A,
    MATRIX result )
```

Computes element-wise matrix inverse.

Parameters

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

Returns

$$\mathbf{11}^T ./ \mathbf{A}$$

Here is the call graph for this function:

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_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.

6.10.1 Function Documentation

6.10.1.1 `int_vec_dump()`

```
void int_vec_dump (
    INT_VECTOR A )
```

Dumps an integer vector in the stdout.

Parameters

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

Here is the call graph for this function:

6.10.1.2 `int_vec_dumpf()`

```
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

Here is the call graph for this function:

6.10.1.3 int_vec_fdump()

```
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

Here is the call graph for this function:

6.10.1.4 int_vec_fdumpf()

```
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

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

Here is the call graph for this function: Here is the caller graph for this function:

6.10.1.5 mat_dump()

```
void mat_dump (
    MATRIX A )
```

Dumps a matrix in the stdout.

Parameters

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

Here is the call graph for this function:

6.10.1.6 mat_dumpf()

```
void mat_dumpf (
```

```
MATRIX A,
const char * s )
```

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

Parameters

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

Here is the call graph for this function:

6.10.1.7 mat_fdump()

```
void mat_fdump (
    MATRIX A,
    MAT_FILEPOINTER fp )
```

Dumps a matrix in an opened file.

Parameters

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

Here is the call graph for this function: Here is the caller graph for this function:

6.10.1.8 mat_fdumpf()

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

Here is the call graph for this function: Here is the caller graph for this function:

6.11 matdurbn.c File Reference

Functions

- [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.

- [MATSTACK](#) `mat_qr` ([MATRIX](#) `A`, [MATSTACK](#) `qr`)

Computes QR decomposition.

6.11.1 Function Documentation

6.11.1.1 `mat_durbin()`

```
MATRIX mat_durbin (
    MATRIX R,
    MATRIX result )
```

Runs Levinson-Durbin algorithm.

Parameters

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

Returns

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

Here is the call graph for this function: Here is the caller graph for this function:

6.11.1.2 `mat_solve_durbin()`

```
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 $B = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \end{bmatrix}$
Generated by Doxygen		
in	<i>result</i>	Matrix to store the result

Returns

X where $RX = B$

Here is the call graph for this function:

6.11.1.3 mat_qr()

```
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

Here is the call graph for this function:

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 gen_error()

```
int gen_error (
    int err_ )
```

Generates error message for general errors and exits.

Parameters

in	err	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)
----	-----	---

Here is the caller graph for this function:

6.12.1.2 graph_error()

```
int graph_error (
    int err_ )
```

Generates error message for graph errors and exits.

Parameters

in	err	Error type (GRAPH_MALLOC/GRAPH_READ/GRAPH_ELSE)
----	-----	---

Here is the caller graph for this function:

6.12.1.3 int_vec_error()

```
INT_VECTOR int_vec_error (
    int err_ )
```

Generates error message for integer vector errors and exits.

Parameters

in	err	Error type (INT_VEC_MALLOC/INT_VEC_FNOTOPEN/INT_VEC_FNOTGETINT_VEC/INT_VE↔ C_SIZEMISMATCH)
----	-----	---

Here is the caller graph for this function:

6.12.1.4 int_vecstack_error()

```
INT_VECSTACK int_vecstack_error (
    int err_ )
```

Generates error message for integer vector stack errors and exits.

Parameters

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

Here is the caller graph for this function:

6.12.1.5 mat_error()

```
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)
----	-----	--

6.12.1.6 matstack_error()

```
MATSTACK matstack_error (
    int err_ )
```

Generates error message for matrix stack errors and exits.

Parameters

in	err	Error type (MATSTACK_MALLOC/MATSTACK_FNOTOPEN/MATSTACK_FNOTGETMAT/MATSTACK_SIZEMISMATCH/MATSTACK_INVERSE_ERROR)
----	-----	---

Here is the caller graph for this function:

6.12.1.7 pq_error()

```
int pq_error (
    int err_ )
```

Generates error message for priority queue errors and exits.

Parameters

in	err	Error type (PQ_MALLOC/PQ_EMPTY)
----	-----	---------------------------------

Here is the caller graph for this function:

6.12.1.8 queue_error()

```
int queue_error (
    int err_ )
```

Generates error message for queue errors and exits.

Parameters

in	err	Error type (QUEUE_MALLOC/QUEUE_EMPTY)
----	-----	---------------------------------------

Here is the caller graph for this function:

6.12.1.9 stack_error()

```
int stack_error (
    int err_ )
```

Generates error message for stack errors and exits.

Parameters

in	err	Error type (STACK_MALLOC/STACK_EMPTY)
----	-----	---------------------------------------

Here is the caller graph for this function:

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 mat_fft2()

```
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 (MAT_FFT2_FORWARD/MAT_FFT2_BACKWARD)
in	<i>result</i>	Matrix stack to store the result

Returns

Transformed matrix stack

Here is the call graph for this function:

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 mat_conv2()

```
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

Here is the call graph for this function:

6.15 matfit.c File Reference

Functions

- [MATRIX mat_linear_ls_fit](#) ([MATRIX](#) A, [MATRIX](#) Y, int deg, [MATRIX](#) result)
Performs 2-d polynomial model fitting using least squares.
- [MATRIX mat_least_squares](#) ([MATRIX](#) A, [MATRIX](#) Y, [MATRIX](#) result)
Solves linear equations using least squares.
- [MATRIX mat_w_least_squares](#) ([MATRIX](#) A, [MATRIX](#) Y, [MATRIX](#) w, [MATRIX](#) result)
Solves linear equations using weighted least squares.
- [MATRIX mat_rob_least_squares](#) ([MATRIX](#) A, [MATRIX](#) Y, int lossfunc, [MATRIX](#) result)
Solves linear equations using robust reweighted least squares.
- [MATRIX mat_robust_fit](#) ([MATRIX](#) A, [MATRIX](#) Y, int deg, int lossfunc, [MATRIX](#) result)
Performs 2-d polynomial model fitting using robust least squares.

6.15.1 Function Documentation

6.15.1.1 mat_least_squares()

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

Solves linear equations using least squares.

Parameters

in	A	Input data matrix
in	Y	Input observation matrix
in	result	Matrix to store the result

Returns

$$(\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{Y}$$

Here is the call graph for this function: Here is the caller graph for this function:

6.15.1.2 mat_linear_ls_fit()

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

Performs 2-d polynomial model fitting using least squares.

Parameters

in	<i>A</i>	Input data column matrix
in	<i>Y</i>	Input observation column matrix
in	<i>deg</i>	Polynomial degree N
in	<i>result</i>	Matrix to store the result

Returns

Polynomial co-efficient matrix $[\alpha_N \quad \cdots \quad \alpha_0]^T$

Here is the call graph for this function:

6.15.1.3 mat_rob_least_squares()

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

Solves linear equations using robust reweighted least squares.

Parameters

in	<i>A</i>	Input data matrix
in	<i>Y</i>	Input observation matrix
in	<i>lossfunc</i>	Loss function type (MAT_LOSS_BISQUARE/MAT_LOSS_HUBER)
in	<i>result</i>	Matrix to store the result

Returns

Robust **X**

Here is the call graph for this function: Here is the caller graph for this function:

6.15.1.4 mat_robust_fit()

```
MATRIX mat_robust_fit (
    MATRIX A,
```

```

    MATRIX Y,
    int deg,
    int lossfunc,
    MATRIX result )

```

Performs 2-d polynomial model fitting using robust least squares.

Parameters

in	<i>A</i>	Input data column matrix
in	<i>Y</i>	Input observation column matrix
in	<i>deg</i>	Polynomial degree N
in	<i>lossfunc</i>	Loss function type (MAT_LOSS_BISQUARE/MAT_LOSS_HUBER)
in	<i>result</i>	Matrix to store the result

Returns

Polynomial co-efficient matrix $[\alpha_N \ \cdots \ \alpha_0]^T$

Here is the call graph for this function:

6.15.1.5 mat_w_least_squares()

```

MATRIX mat_w_least_squares (
    MATRIX A,
    MATRIX Y,
    MATRIX w,
    MATRIX result )

```

Solves linear equations using weighted least squares.

Parameters

in	<i>A</i>	Input data matrix
in	<i>Y</i>	Input observation matrix
in	<i>w</i>	Input weight column matrix
in	<i>result</i>	Matrix to store the result

Returns

$(\mathbf{A}^T \text{diag}(w) \mathbf{A})^{-1} \mathbf{A}^T \text{diag}(w) \mathbf{Y}$

Here is the call graph for this function: Here is the caller graph for this function:

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 `mat_fliplr()`

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

Here is the call graph for this function:

6.16.1.2 `mat_flipud()`

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

Here is the call graph for this function:

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.
- `mtype __mat_bisquare_wt` (`mtype x`, `mtype k`)
Computes bisquare weight 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.
- `mtype __mat_logplusone` (`mtype x`)
Computes logarithm plus one function.

- mtype [__mat_round_away_zero](#) (mtype x)
Rounds a number away from zero.
- mtype [__mat_round_towards_zero](#) (mtype x)
Rounds a number towards zero.
- [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 [__mat_addfunc\(\)](#)

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

Computes addition function.

Parameters

in	x	
in	y	

Returns

$x + y$

6.17.1.2 [__mat_arccosh\(\)](#)

```
mtype __mat_arccosh (
    mtype x )
```

Computes inverse hyperbolic cosine function.

Parameters

in	x	
----	-------------------	--

Returns

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

6.17.1.3 __mat_arcsinh()

```
mtype __mat_arcsinh (
    mtype x )
```

Computes inverse hyperbolic sine function.

Parameters

in	x	
----	---	--

Returns

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

Here is the call graph for this function:

6.17.1.4 __mat_arctanh()

```
mtype __mat_arctanh (
    mtype x )
```

Computes inverse hyperbolic tangent function.

Parameters

in	x	
----	---	--

Returns

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

Here is the call graph for this function:

6.17.1.5 __mat_bisquare_wt()

```
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 __mat_divfunc()

```

mtype __mat_divfunc (
    mtype x,
    mtype y )

```

Computes division function.

Parameters

in	x	
in	y	

Returns

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

6.17.1.7 __mat_huber_wt()

```

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 __mat_logplusone()

```

mtype __mat_logplusone (
    mtype x )

```

Computes logarithm plus one function.

Parameters

in	x	
----	-----	--

Returns

$\log(1 + x)$

Here is the caller graph for this function:

6.17.1.9 __mat_mulfunc()

```

mtype __mat_mulfunc (
    mtype x,
    mtype y )

```

Computes multiplication function.

Parameters

in	x	
in	y	

Returns

xy

Here is the caller graph for this function:

6.17.1.10 __mat_round_away_zero()

```

mtype __mat_round_away_zero (
    mtype x )

```

Rounds a number away from zero.

Parameters

in	x	Input value
----	-----	-------------

Returns

$$\text{sgn}(x) \lfloor |x| + 0.5 \rfloor$$

6.17.1.11 __mat_round_towards_zero()

```
mtype __mat_round_towards_zero (
    mtype x )
```

Rounds a number towards zero.

Parameters

in	x	Input value
----	-----	-------------

Returns

$$\text{sgn}(x) \lceil |x| - 0.5 \rceil$$

6.17.1.12 __mat_sqrfunc()

```
mtype __mat_sqrfunc (
    mtype x )
```

Computes square function.

Parameters

in	x	
----	-----	--

Returns

$$x^2$$

Here is the caller graph for this function:

6.17.1.13 __mat_sqrtfunc()

```
mtype __mat_sqrtfunc (
    mtype x )
```

Computes square root function.

Parameters

in	x	
----	-----	--

Returns

$$\sqrt{x}$$

Here is the caller graph for this function:

6.17.1.14 __mat_subfunc()

```

mtype __mat_subfunc (
    mtype x,
    mtype y )

```

Computes subtraction function.

Parameters

in	x	
in	y	

Returns

$$x - y$$

Here is the caller graph for this function:

6.17.1.15 mat_bisquare_wt()

```

MATRIX mat_bisquare_wt (
    MATRIX A,
    mtype k,
    mtype sigma,
    MATRIX result )

```

Computes bisquare weight function element-wise on a matrix.

Parameters

in	A	Input matrix
in	k	Bisquare parameter

Returns

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

Here is the call graph for this function: Here is the caller graph for this function:

6.17.1.16 mat_gfunc()

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

Computes a given function element-wise on a matrix.

Parameters

in	A	Input matrix
in	f	Given function

Returns

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

Here is the call graph for this function: Here is the caller graph for this function:

6.17.1.17 mat_huber_wt()

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

Computes Huber weight function element-wise on a matrix.

Parameters

in	A	Input matrix
in	k	Huber parameter

Returns

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

Here is the call graph for this function: Here is the caller graph for this 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 `mat_graph_adjlist()`

```
void mat_graph_adjlist (
    MAT_GRAPH g,
    int directed,
    int weighted,
    MAT_FILEPOINTER fp )
```

Here is the call graph for this function:

6.18.1.2 `mat_graph_adjm_to_adjl()`

```
void mat_graph_adjm_to_adjl (
    MAT_GRAPH g,
    MATRIX a )
```

Here is the call graph for this function:

6.18.1.3 `mat_graph_creat()`

```
MAT_GRAPH mat_graph_creat (
    void )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.18.1.4 `mat_graph_dump()`

```
void mat_graph_dump (
    MAT_GRAPH g,
    int mst )
```

Here is the call graph for this function:

6.18.1.5 mat_graph_dumpf()

```
void mat_graph_dumpf (
    MAT_GRAPH g,
    int mst,
    MAT_FILEPOINTER fp )
```

Here is the caller graph for this function:

6.18.1.6 mat_graph_reverse()

```
MAT_GRAPH mat_graph_reverse (
    MAT_GRAPH g,
    MAT_GRAPH r )
```

Here is the call graph for this function:

6.18.1.7 mat_graph_search()

```
MAT_INT_QUEUE mat_graph_search (
    MAT_GRAPH g,
    int connected,
    int mst )
```

Here is the call graph for this function:

6.18.1.8 mat_graph_visit()

```
void mat_graph_visit (
    MAT_GRAPH g,
    int k,
    int connected,
    int mst,
    MAT_INT_PRIORITYQUEUE pq,
    MAT_INT_QUEUE q )
```

Here is the call graph for this function: Here is the caller graph for this function:

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 `mat_innerprod()`

```

mtype mat_innerprod (
    MATRIX A,
    MATRIX B )

```

Here is the call graph for this function: Here is the caller graph for this function:

6.19.1.2 `mat_norm_inf()`

```

mtype mat_norm_inf (
    MATRIX A )

```

6.19.1.3 `mat_norm_one()`

```

mtype mat_norm_one (
    MATRIX A )

```

Here is the caller graph for this function:

6.19.1.4 `mat_norm_p()`

```

mtype mat_norm_p (
    MATRIX A,
    mtype p )

```

Here is the caller graph for this function:

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_qadrat` (mtype(*func)(mtype), mtype lower, mtype upper)
Computes Gauss quadrature integration.

6.20.1 Function Documentation

6.20.1.1 mat_int_qadrat()

```
mtype mat_int_qadrat (
    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

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

Here is the call graph for this function:

6.20.1.2 mat_int_simpson()

```

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

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

6.20.1.3 mat_int_trapezoid()

```

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

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

6.21 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 mat_inv()

```
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

$$A^{-1}$$

Here is the call graph for this function: Here is the caller graph for this function:

6.21.1.2 mat_reg_inv()

```
MATRIX mat_reg_inv (
    MATRIX A,
    mtype r,
    MATRIX result )
```

Computes the regularized inverse of a matrix.

Parameters

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

Returns

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

Here is the call graph for this function: Here is the caller graph for this function:

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 mat_kdtree_free()

```
int mat_kdtree_free (
    MAT\_KDTREE t )
```

Frees a k-d tree.

Parameters

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

Returns

Success

Here is the call graph for this function:

6.22.1.2 mat_kdtree_k_nearest()

```
MATRIX mat_kdtree_k_nearest (
    MAT\_KDTREE t,
    MATRIX A,
    int k,
    MATRIX result )
```

Computes k nearest neighbors.

Parameters

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

Returns

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

Here is the call graph for this function:

6.22.1.3 mat_kdtree_make_tree()

```
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

Here is the call graph for this function:

6.22.1.4 mat_kdtree_nearest()

```
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 $j = 1, 2, \dots, N$

Here is the call graph for this function:

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 mat_max()

```
MATVEC_DPOINTER mat_max (
    MATRIX A,
    int dim )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.23.1.2 mat_min()

```
MATVEC_DPOINTER mat_min (
    MATRIX A,
    int dim )
```

Here is the call graph for this function: Here is the caller graph for this function:

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 __mat_mds_metric()

```
MATRIX __mat_mds_metric (
    MATRIX d,
    int dims,
    MATRIX result )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.24.1.2 __mat_mds_nonmetric()

```
MATRIX __mat_mds_nonmetric (
    MATRIX d,
    int dims,
    MATRIX result )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.24.1.3 mat_mds()

```
MATRIX mat_mds (
    MATRIX d,
    int dims,
    int type,
    MATRIX result )
```

Here is the call graph for this function:

6.25 matmean.c File Reference

Functions

- mtype [mat_mean](#) (MATRIX A)
Computes the mean of a matrix.
- [MATRIX mat_mean_row](#) (MATRIX A, MATRIX result)
Computes row-mean of a matrix.
- [MATRIX mat_mean_col](#) (MATRIX A, MATRIX result)
Computes column-mean of a matrix.
- mtype [int_vec_mean](#) (INT_VECTOR A)
Computes element-mean of an integer vector.

6.25.1 Function Documentation

6.25.1.1 int_vec_mean()

```
mtype int_vec_mean (
    INT_VECTOR A )
```

Computes element-mean of an integer vector.

Parameters

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

Returns

$\text{mean}(A)$

6.25.1.2 mat_mean()

```
mtype mat_mean (
    MATRIX A )
```

Computes the mean of a matrix.

Parameters

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

Returns

$\text{mean}(A)$

Here is the call graph for this function:

6.25.1.3 mat_mean_col()

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

Computes column-mean of a matrix.

Parameters

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

Returns

$\mathbf{1}^T A / \#rows$

Here is the call graph for this function:

6.25.1.4 mat_mean_row()

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

Computes row-mean of a matrix.

Parameters

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

Returns

$A1/\#cols$

Here is the call graph for this function: Here is the caller graph for this function:

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 `__mat_cart2pol()`

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

6.26.1.2 `__mat_pol2cart()`

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

6.26.1.3 `int_vec_permute_vect()`

```
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

Here is the call graph for this function:

6.26.1.4 int_vec_unique()

```
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

Here is the call graph for this function: Here is the caller graph for this function:

6.26.1.5 mat_bsxfun()

```
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

Here is the call graph for this function: Here is the caller graph for this function:

6.26.1.6 mat_calc_dist_sq()

```
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

Here is the call graph for this function: Here is the caller graph for this function:

6.26.1.7 mat_cart2pol()

```

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

Here is the call graph for this function:

6.26.1.8 mat_find_within_dist()

```

INT_VECTOR mat_find_within_dist (
    MATRIX A,
    MATRIX d,
    mtype range )

```

Finds points within a neighborhood.

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>range</i>	Radius to search within

Returns

Indices Vector

Here is the call graph for this function:

6.26.1.9 mat_fnextline()

```
void mat_fnextline (
    MAT_FILEPOINTER fp )
```

Prints nextline to file.

Parameters

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

Here is the caller graph for this function:

6.26.1.10 mat_get_sub_matrix_from_cols()

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

Extracts sub-matrix from columns of a matrix.

Parameters

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

Returns

Extracted matrix

Here is the call graph for this function: Here is the caller graph for this function:

6.26.1.11 mat_get_sub_matrix_from_rows()

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

Extracts sub-matrix from rows of a matrix.

Parameters

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

Returns

Extracted matrix

Here is the call graph for this function: Here is the caller graph for this function:

6.26.1.12 mat_get_sub_vector()

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

Extracts sub-vector from an integer vector.

Parameters

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

Returns

Extracted vector

Here is the call graph for this function:

6.26.1.13 mat_nextline()

```
void mat_nextline (
    void )
```

Prints nextline to stdout.

Here is the call graph for this function:

6.26.1.14 mat_pick_col()

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

Picks a column from a matrix.

Parameters

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

Returns

Column matrix

Here is the call graph for this function: Here is the caller graph for this function:

6.26.1.15 mat_pick_row()

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

Picks a row from a matrix.

Parameters

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

Returns

Row matrix

Here is the call graph for this function:

6.26.1.16 mat_pol2cart()

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

Converts polar co-ordinates to Cartesian co-ordinates.

Parameters

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

Returns

Cartesian co-ordinate matrix

Here is the call graph for this function:

6.26.1.17 mats_isinf()

```
int mats_isinf (
    mtype x )
```

Checks if scalar is infinite.

Parameters

in	x	Input scalar
----	---	--------------

Returns

Zero/non-zero

6.26.1.18 mats_isnan()

```
int mats_isnan (
    mtype x )
```

Checks if scalar is NaN.

Parameters

in	x	Input scalar
----	---	--------------

Returns

Zero/non-zero

6.27 matmul.c File Reference**Functions**

- [MATRIX mat_mul](#) ([MATRIX A](#), [MATRIX B](#), [MATRIX result](#))
Computes matrix multiplication.
- [MATRIX mat_mul_fast](#) ([MATRIX A](#), [MATRIX B](#), [MATRIX result](#))
Computes fast matrix multiplication (not implemented)
- [MATRIX mat_muls](#) ([MATRIX A](#), [mtype s](#), [MATRIX result](#))

- Multiplies a matrix by a scalar.*

 - `MATRIX mat_mul_dot` (`MATRIX A`, `MATRIX B`, `MATRIX result`)

Computes element-wise matrix multiplication.
- `mtype mat_diagmul` (`MATRIX A`)

Computes matrix diagonal product.
- `INT_VECTOR int_vec_mul` (`INT_VECTOR A`, `INT_VECTOR B`, `INT_VECTOR result`)

Computes element-wise integer vector multiplication.
- `INT_VECTOR int_vec_muls` (`INT_VECTOR A`, `int x`, `INT_VECTOR result`)

Multiplies an integer vector by a scalar.

6.27.1 Function Documentation

6.27.1.1 int_vec_mul()

```
INT_VECTOR int_vec_mul (
    INT_VECTOR A,
    INT_VECTOR B,
    INT_VECTOR result )
```

Computes element-wise integer vector multiplication.

Parameters

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

Returns

$A * B$

Here is the call graph for this function:

6.27.1.2 int_vec_muls()

```
INT_VECTOR int_vec_muls (
    INT_VECTOR A,
    int x,
    INT_VECTOR result )
```

Multiplies an integer vector by a scalar.

Parameters

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

Returns
 sA

Here is the call graph for this function: Here is the caller graph for this function:

6.27.1.3 mat_diagmul()

```
mtype mat_diagmul (
    MATRIX A )
```

Computes matrix diagonal product.

Parameters

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

Returns
 $\text{prod}(\text{diag}(\mathbf{A}))$
6.27.1.4 mat_mul()

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

Computes matrix multiplication.

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
 \mathbf{AB}

Here is the call graph for this function: Here is the caller graph for this function:

6.27.1.5 mat_mul_dot()

```
MATRIX mat_mul_dot (
    MATRIX A,
```



```
MATRIX B,  
MATRIX result )
```

Computes element-wise matrix multiplication.

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

$A * B$

Here is the call graph for this function: Here is the caller graph for this function:

6.27.1.6 mat_mul_fast()

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

Computes fast matrix multiplication (not implemented)

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

AB

Here is the call graph for this function: Here is the caller graph for this function:

6.27.1.7 mat_muls()

```
MATRIX mat_muls (
    MATRIX A,
    mtype s,
    MATRIX result )
```

Multiplies 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

$s\mathbf{A}$

Here is the call graph for this function: Here is the caller graph for this function:

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_spcol](#) ([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 mat_corcol()

```
MATSTACK mat_corcol (
    MATRIX data )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.28.1.2 mat_covcol()

```
MATSTACK mat_covcol (
    MATRIX data )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.28.1.3 mat_eig_sym()

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

Here is the call graph for this function: Here is the caller graph for this function:

6.28.1.4 mat_pca()

```
MATSTACK mat_pca (
    MATRIX data,
    int pca_type )
```

Here is the call graph for this function:

6.28.1.5 mat_scpcol()

```
MATRIX mat_scpcol (
    MATRIX data )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.28.1.6 mat_tqli()

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

Here is the call graph for this function: Here is the caller graph for this function:

6.28.1.7 mat_tred2()

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

Here is the caller graph for this function:

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 mat_pinv()

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

Computes pseudo-inverse of a matrix.

Parameters

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

Returns

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

Here is the call graph for this function: Here is the caller graph for this function:

6.29.1.2 mat_wpinv()

```
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

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

Here is the call graph for this function: Here is the caller graph for this function:

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](#)(*f)([mtype](#)), [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 mat_binom()

```
mtype mat_binom (
    int n,
    int k )
```

Computes a binomial co-efficient.

Parameters

in	n	1 st argument
in	k	2 nd argument

Returns

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

Here is the call graph for this function: Here is the caller graph for this function:

6.30.1.2 mat_binom_init()

```
void mat_binom_init ( )
```

Initializes the binomial series.

Here is the call graph for this function: Here is the caller graph for this function:

6.30.1.3 mat_cheby()

```
MATRIX mat_cheby (
    int n )
```

Computes the n^{th} Chebyshev polynomial.

Parameters

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

Returns

Output polynomial matrix

Here is the call graph for this function: Here is the caller graph for this function:

6.30.1.4 mat_cheby_approx()

```
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

Here is the call graph for this function:

6.30.1.5 mat_cheby_coeffs_to_poly()

```
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

Here is the call graph for this function: Here is the caller graph for this function:

6.30.1.6 mat_cheby_init()

```
void mat_cheby_init ( )
```

Initializes the Chebyshev polynomial series.

Here is the call graph for this function:

6.30.1.7 mat_legendre()

```
MATRIX mat_legendre (
    int n )
```

Computes the n^{th} Legendre polynomial.

Parameters

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

Returns

Output polynomial matrix

Here is the call graph for this function: Here is the caller graph for this function:

6.30.1.8 mat_legendre_init()

```
void mat_legendre_init ( )
```

Initializes the Legendre polynomial series.

Here is the call graph for this function:

6.30.1.9 mat_poly_add()

```
MATRIX mat_poly_add (
    MATRIX A,
    MATRIX B,
    MATRIX result )
```

Adds 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

Here is the call graph for this function: Here is the caller graph for this function:

6.30.1.10 mat_poly_diff()

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

Computes derivative polynomial of a polynomial.

Parameters

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

Returns

Output matrix

Here is the call graph for this function:

6.30.1.11 `mat_poly_diff_eval()`

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

Evaluates derivative polynomial at a point.

Parameters

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

Returns

Output matrix

Here is the call graph for this function:

6.30.1.12 `mat_poly_div()`

```
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

Here is the call graph for this function:

6.30.1.13 `mat_poly_eval()`

```
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

Here is the call graph for this function:

6.30.1.14 mat_poly_mul()

```
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

Here is the call graph for this function:

6.30.1.15 mat_poly_scale()

```
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

Here is the call graph for this function: Here is the caller graph for this function:

6.30.1.16 mat_poly_shift()

```
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

Here is the call graph for this function: Here is the caller graph for this function:

6.30.2 Variable Documentation**6.30.2.1 mat_binom_series_table**

```
MATSTACK mat_binom_series_table
```

6.30.2.2 mat_cheby_series_table

```
MATSTACK mat_cheby_series_table
```

6.30.2.3 mat_legendre_series_table

```
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 mat_bayes_classifier_test()

```
INT_VECTOR mat_bayes_classifier_test (
    MATRIX data,
    MAT_BAYES_MODEL b_model )
```

Here is the call graph for this function:

6.31.1.2 mat_bayes_classifier_train()

```
MAT_BAYES_MODEL mat_bayes_classifier_train (
    MATRIX data,
    INT_VECTOR labels )
```

Here is the call graph for this function:

6.31.1.3 mat_kmeans()

```
MATVEC_DPOINTER mat_kmeans (
    MATRIX data,
    int k,
    int iters,
    MATVEC_DPOINTER result )
```

Here is the call graph for this function:

6.31.1.4 mat_perceptron_test()

```
INT_VECTOR mat_perceptron_test (
    MATRIX data,
    MAT_PERCEPTRON p_model )
```

Here is the call graph for this function:

6.31.1.5 mat_perceptron_train()

```
MAT_PERCEPTRON mat_perceptron_train (
    MATRIX data,
    INT_VECTOR labels,
    int num_of_iterations )
```

Here is the call graph for this function:

6.31.1.6 mat_perceptron_train_()

```
MAT_PERCEPTRON mat_perceptron_train_ (
    MATRIX data1,
    MATRIX data2,
    MAT_PERCEPTRON p_model,
    int class_num )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.32 matpursuit.c File Reference

Functions

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

6.32.1 Function Documentation

6.32.1.1 mat_omp()

```
MATSTACK mat_omp (
    MATRIX A,
    MATRIX b,
    int k,
    mtype tol,
    MATSTACK result )
```

Here is the call graph for this function:

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, mtype 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 __mat_rand()

```
mtype __mat_rand (
    void )
```

Here is the call graph for this function:

6.33.1.2 __mat_randexp()

```
mtype __mat_randexp (
    mtype mu )
```

Here is the call graph for this function:

6.33.1.3 __mat_randfun()

```
mtype __mat_randfun (
    mtype(*) (mtype) fun,
    mtype xmin,
    mtype xmax )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.33.1.4 __mat_randn()

```
mtype __mat_randn (  
    void )
```

Here is the call graph for this function:

6.33.1.5 int_vec_randperm()

```
INT_VECTOR int_vec_randperm (  
    int n,  
    INT_VECTOR result )
```

Here is the call graph for this function:

6.33.1.6 mat_rand()

```
MATRIX mat_rand (  
    int n,  
    int m,  
    MATRIX result )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.33.1.7 mat_randexp()

```
MATRIX mat_randexp (  
    int n,  
    int m,  
    mtype mu,  
    MATRIX result )
```

Here is the call graph for this function:

6.33.1.8 mat_randfun()

```
MATRIX mat_randfun (  
    int n,  
    int m,  
    mtype(*) (mtype) fun,  
    mtype xmin,  
    mtype xmax,  
    MATRIX result )
```

Here is the call graph for this function:

6.33.1.9 mat_randn()

```
MATRIX mat_randn (
    int n,
    int m,
    MATRIX result )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.33.1.10 mat_randperm()

```
MATRIX mat_randperm (
    int m,
    int n,
    MATRIX result )
```

Here is the call graph for this function:

6.33.1.11 mat_randperm_n()

```
MATRIX mat_randperm_n (
    int n,
    MATRIX result )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.33.1.12 mat_set_seed()

```
void mat_set_seed (
    int seed )
```

Here is the caller graph for this function:

6.33.2 Variable Documentation

6.33.2.1 MAT_SEED

```
unsigned int MAT_SEED = 0
```

6.33.2.2 MAT_SET_SEED

```
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

- [__declspec](#) (thread) clock_t [MAT_CLOCK_TIME](#)
Starts stopwatch timer.
- 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)
- Concatenates two integer vectors.*

 - [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.*

 - int [int_vec_sum](#) ([INT_VECTOR](#) A)

- Computes element-sum of an integer vector.*

 - mtype [int_vec_mean](#) ([INT_VECTOR](#) A)
- Computes element-mean of an integer vector.*

 - mtype [mat_sum](#) ([MATRIX](#) A)
- Computes element-sum of a matrix.*

 - [MATRIX mat_sum_row](#) ([MATRIX](#) A, [MATRIX](#) result)
- Computes row-sum of a matrix.*

 - [MATRIX mat_sum_col](#) ([MATRIX](#) A, [MATRIX](#) result)
- Computes column-sum of a matrix.*

 - mtype [mat_mean](#) ([MATRIX](#) A)
- Computes the mean of a matrix.*

 - [MATRIX mat_mean_row](#) ([MATRIX](#) A, [MATRIX](#) result)
- Computes row-mean of a matrix.*

 - [MATRIX mat_mean_col](#) ([MATRIX](#) A, [MATRIX](#) result)
- Computes column-mean of a matrix.*

 - [INT_VECTOR int_vec_abs](#) ([INT_VECTOR](#) A, [INT_VECTOR](#) result)
- Computes absolute value of an integer vector.*

 - [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)
- Subtracts an integer vector from integer vector.*

 - [INT_VECTOR int_vec_subs](#) ([INT_VECTOR](#) A, int s, [INT_VECTOR](#) result)
- Subtracts an integer from integer vector.*

 - [INT_VECTOR int_vec_subs_neg](#) ([INT_VECTOR](#) A, int s, [INT_VECTOR](#) result)
- Subtracts an integer vector from an integer.*

 - [INT_VECTOR int_vec_mul](#) ([INT_VECTOR](#) A, [INT_VECTOR](#) B, [INT_VECTOR](#) result)
- Computes element-wise integer vector multiplication.*

 - [INT_VECTOR int_vec_muls](#) ([INT_VECTOR](#) A, int s, [INT_VECTOR](#) result)
- Multiplies an integer vector by a scalar.*

 - [INT_VECTOR int_vec_inv](#) ([INT_VECTOR](#) A, [INT_VECTOR](#) result)
- Computes element-wise integer vector inverse.*

 - [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.*

 - [INT_VECTOR int_vec_divs_inv](#) ([INT_VECTOR](#) A, int s, [INT_VECTOR](#) result)
- Divides a scalar by an integer vector.*

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

 - [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)
- Subtracts a matrix from another matrix.*

 - [MATRIX mat_subs](#) ([MATRIX](#) A, mtype s, [MATRIX](#) result)
- Subtracts a scalar from a matrix.*

 - [MATRIX mat_subs_neg](#) ([MATRIX](#) A, mtype s, [MATRIX](#) result)
- Subtracts a matrix from a scalar.*

- [MATRIX mat_mul](#) ([MATRIX A](#), [MATRIX B](#), [MATRIX result](#))
Computes matrix multiplication.
- [MATRIX mat_mul_fast](#) ([MATRIX A](#), [MATRIX B](#), [MATRIX result](#))
Computes fast matrix multiplication (not implemented)
- [MATRIX mat_mul_dot](#) ([MATRIX A](#), [MATRIX B](#), [MATRIX result](#))
Computes element-wise matrix multiplication.
- [MATRIX mat_muls](#) ([MATRIX A](#), [mtype s](#), [MATRIX result](#))
Multiplies a matrix by a scalar.
- [MATRIX mat_inv_dot](#) ([MATRIX A](#), [MATRIX result](#))
Computes element-wise matrix inverse.
- [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.
- [MATRIX mat_divs_inv](#) ([MATRIX A](#), [mtype s](#), [MATRIX result](#))
Divides a scalar by a matrix.
- [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](#))
Computes matrix diagonal product.
- [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](#))
Computes the symmetric Toeplitz matrix from a co-efficient matrix.
- [int mat_lu](#) ([MATRIX A](#), [MATRIX P](#))
Computes LU decomposition of a matrix.
- [void mat_backsubs1](#) ([MATRIX A](#), [MATRIX B](#), [MATRIX C](#), [MATRIX P](#), [int xcol](#))
- [MATRIX mat_lsolve](#) ([MATRIX A](#), [MATRIX b](#), [MATRIX result](#))
Solves linear equations $Ax = b$.
- [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.
- [MATSTACK mat_svd](#) ([MATRIX a](#), [int niters](#), [MATSTACK result](#))
Computes the SVD of a matrix.
- [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_lsolve_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)
Solves linear equations using least squares.
- [MATRIX](#) [mat_w_least_squares](#) ([MATRIX](#) A, [MATRIX](#) Y, [MATRIX](#) w, [MATRIX](#) result)
Solves linear equations using weighted least squares.
- [MATRIX](#) [mat_rob_least_squares](#) ([MATRIX](#) A, [MATRIX](#) Y, int lossfunc, [MATRIX](#) result)
Solves linear equations using robust reweighted least squares.
- [MATRIX](#) [mat_linear_ls_fit](#) ([MATRIX](#) A, [MATRIX](#) Y, int deg, [MATRIX](#) result)
Performs 2-d polynomial model fitting using least squares.
- [MATRIX](#) [mat_robust_fit](#) ([MATRIX](#) A, [MATRIX](#) Y, int deg, int lossfunc, [MATRIX](#) result)
Performs 2-d polynomial model fitting using robust least squares.
- [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.
- [mtype __mat_round_away_zero](#) ([mtype x](#))
Rounds a number away from zero.
- [mtype __mat_round_towards_zero](#) ([mtype x](#))
Rounds a number towards zero.
- [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)
Checks if greater than zero.
- int [gen_lt](#) (mtype a)
Checks if less than zero.
- int [gen_eq](#) (mtype a)
Checks if equals to zero.
- 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)
- 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_PRIO](#)←[RITYQUEUE](#) pq, [MATRIX](#) bmax, [MATRIX](#) bmin)
- [MATSTACK](#) [mat_omp](#) ([MATRIX](#) A, [MATRIX](#) b, int k, mtype tol, [MATSTACK](#) result)

Variables

- [_Thread_local](#) 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 INT_VECSTACK

```
typedef INT\_VECTOR* INT\_VECSTACK
```

Integer Vector Stack

6.35.1.2 INT_VECTOR

```
typedef int* INT\_VECTOR
```

Integer Vector

6.35.1.3 mat_bayes_model

```
typedef struct mat\_bayes\_model mat\_bayes\_model
```

Bayes Classifier Model Structure.

Bayes Classifier Model

6.35.1.4 MAT_BAYES_MODEL

```
typedef mat\_bayes\_model\* MAT\_BAYES\_MODEL
```

Bayes Classifier Model Pointer

6.35.1.5 mat_gnode

```
typedef struct mat\_gnode mat\_gnode
```

Graph Node Structure.

Graph Node

6.35.1.6 MAT_GNODE

```
typedef mat\_gnode\* MAT\_GNODE
```

Graph Node Pointer

6.35.1.7 mat_graph

```
typedef struct mat\_graph mat\_graph
```

Graph Structure.

6.35.1.8 MAT_GRAPH

```
typedef mat\_graph\* MAT\_GRAPH
```

6.35.1.9 mat_int_priorityqueue

```
typedef struct mat\_int\_priorityqueue mat\_int\_priorityqueue
```

Integer Priority Queue Structure.

Integer Priority Queue

6.35.1.10 MAT_INT_PRIORITYQUEUE

```
typedef mat\_int\_priorityqueue\* MAT\_INT\_PRIORITYQUEUE
```

Integer Priority Queue Pointer

6.35.1.11 mat_int_queue

```
typedef struct mat_int_queue mat_int_queue
```

Integer Queue Structure.

Integer Queue

6.35.1.12 MAT_INT_QUEUE

```
typedef mat_int_queue* MAT_INT_QUEUE
```

Integer Queue Pointer

6.35.1.13 mat_int_stack

```
typedef struct mat_int_stack mat_int_stack
```

Integer Stack Structure.

Integer Stack

6.35.1.14 MAT_INT_STACK

```
typedef mat_int_stack* MAT_INT_STACK
```

Integer Stack Pointer

6.35.1.15 mat_intpqnode

```
typedef struct mat_intpqnode mat_intpqnode
```

Integer Priority Queue Node Structure.

Integer Priority Queue Node

6.35.1.16 MAT_INTPQNODE

```
typedef mat_intpqnode* MAT_INTPQNODE
```

Integer Priority Queue Node Pointer

6.35.1.17 mat_kdnode

```
typedef struct mat_kdnode mat_kdnode
```


6.35.1.18 MAT_KDNODE

```
typedef mat_kdnode* MAT_KDNODE
```

6.35.1.19 mat_kdtree

```
typedef struct mat_kdtree mat_kdtree
```

6.35.1.20 MAT_KDTREE

```
typedef mat_kdtree* MAT_KDTREE
```

6.35.1.21 mat_mtype_priorityqueue

```
typedef struct mat_mtype_priorityqueue mat_mtype_priorityqueue
```

Mtype Priority Queue Structure.

Mtype Priority Queue

6.35.1.22 MAT_MTYPE_PRIORITYQUEUE

```
typedef mat_mtype_priorityqueue* MAT_MTYPE_PRIORITYQUEUE
```

Mtype Priority Queue Pointer

6.35.1.23 mat_mtype_queue

```
typedef struct mat_mtype_queue mat_mtype_queue
```

Mtype Queue Structure.

Mtype Queue

6.35.1.24 MAT_MTYPE_QUEUE

```
typedef mat_mtype_queue* MAT_MTYPE_QUEUE
```

Mtype Queue Pointer

6.35.1.25 mat_mtype_stack

```
typedef struct mat_mtype_stack mat_mtype_stack
```

Mtype Stack Structure.

Mtype Stack

6.35.1.26 MAT_MTYPE_STACK

```
typedef mat_mtype_stack* MAT_MTYPE_STACK
```

Mtype Stack Pointer

6.35.1.27 mat_mtypepqnode

```
typedef struct mat_mtypepqnode mat_mtypepqnode
```

Mtype Priority Queue Node Structure.

Mtype Priority Queue Node

6.35.1.28 MAT_MTYPEPQNODE

```
typedef mat_mtypepqnode* MAT_MTYPEPQNODE
```

Mtype Priority Queue Node Pointer

6.35.1.29 mat_perceptron

```
typedef struct mat_perceptron mat_perceptron
```

Perceptron Classifier Model Structure.

Perceptron Classifier Model

6.35.1.30 MAT_PERCEPTRON

```
typedef mat_perceptron* MAT_PERCEPTRON
```

Perceptron Classifier Model Pointer

6.35.1.31 mat_qintnode

```
typedef struct mat_qintnode mat_qintnode
```

Integer Queue Node Structure.

Integer Queue Node

6.35.1.32 MAT_QINTNODE

```
typedef mat_qintnode* MAT_QINTNODE
```

Integer Queue Node Pointer

6.35.1.33 mat_qmtypenode

```
typedef struct mat_qmtypenode mat_qmtypenode
```

Mtype Queue Node Structure.

Mtype Queue Node

6.35.1.34 MAT_QMTYPENODE

```
typedef mat_qmtypenode* MAT_QMTYPENODE
```

Mtype Queue Node Pointer

6.35.1.35 MAT_TREE

```
typedef mat_tree_node* MAT_TREE
```

Search Tree Pointer

6.35.1.36 mat_tree_node

```
typedef struct mat_tree_node mat_tree_node
```

Search Tree Node Structure.

Search Tree Node

6.35.1.37 MAT_TREE_NODE

```
typedef mat_tree_node* MAT_TREE_NODE
```

Search Tree Node Pointer

6.35.1.38 MATRIX

```
typedef mtype** MATRIX
```

Mtype Matrix

6.35.1.39 MATSTACK

```
typedef MATRIX\* MATSTACK
```

Mtype Matrix Stack

6.35.1.40 MATVEC_DPOINTER

```
typedef void** MATVEC\_DPOINTER
```

Mtype Matrix - Integer Vector Pair

6.35.2 Function Documentation

6.35.2.1 __declspec()

```
__declspec (
    thread )
```

Starts stopwatch timer.

6.35.2.2 __int_vec_creat()

```
INT\_VECTOR __int_vec_creat (
    int len )
```

Here is the caller graph for this function:

6.35.2.3 __int_vecstack_creat()

```
INT\_VECSTACK __int_vecstack_creat (
    int len )
```

Here is the caller graph for this function:

6.35.2.4 __mat_addfunc()

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

Computes addition function.

Parameters

in	x	
in	y	

Returns

$$x + y$$

6.35.2.5 __mat_arccosh()

```

mtype __mat_arccosh (
    mtype x )

```

Computes inverse hyperbolic cosine function.

Parameters

in	x	
----	-----	--

Returns

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

6.35.2.6 __mat_arcsinh()

```

mtype __mat_arcsinh (
    mtype x )

```

Computes inverse hyperbolic sine function.

Parameters

in	x	
----	-----	--

Returns

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

Here is the call graph for this function:

6.35.2.7 __mat_arctanh()

```

mtype __mat_arctanh (
    mtype x )

```

Computes inverse hyperbolic tangent function.

Parameters

in	x	
----	-----	--

Returns

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

Here is the call graph for this function:

6.35.2.8 __mat_bisquare_wt()

```

mtype __mat_bisquare_wt (
    mtype x,
    mtype k )

```

Computes bisquare weight function.

Parameters

in	x	
in	k	

Returns

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

6.35.2.9 __mat_cart2pol()

```

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

```

6.35.2.10 __mat_creat()

```

MATRIX __mat_creat (
    int r,
    int c )

```

Here is the caller graph for this function:

6.35.2.11 __mat_divfunc()

```

mtype __mat_divfunc (
    mtype x,
    mtype y )

```

Computes division function.

Parameters

in	x	
in	y	

Returns

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

6.35.2.12 __mat_fft()

```

int __mat_fft (
    int dir,
    int m,
    mtype * x,
    mtype * y )

```

Here is the caller graph for this function:

6.35.2.13 __mat_huber_wt()

```

mtype __mat_huber_wt (
    mtype x,
    mtype k )

```

Computes Huber weight function.

Parameters

in	x	
in	k	

Returns

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

6.35.2.14 __mat_kd_find_median()

```
MAT_KDNODE __mat_kd_find_median (
    MAT_KDNODE kd_start,
    MAT_KDNODE kd_end,
    int idx )
```

6.35.2.15 __mat_kdtree_k_nearest()

```
void __mat_kdtree_k_nearest (
    MAT_KDNODE root,
    MAT_KDNODE nd,
    int i,
    int dim,
    MAT_MTYPE_PRIORITYQUEUE pq,
    MATRIX bmax,
    MATRIX bmin )
```

Here is the caller graph for this function:

6.35.2.16 __mat_kdtree_make_tree()

```
MAT_KDNODE __mat_kdtree_make_tree (
    MAT_KDNODE t,
    int len,
    int i,
    int dim )
```

Here is the caller graph for this function:

6.35.2.17 __mat_kdtree_nearest()

```
void __mat_kdtree_nearest (
    MAT_KDNODE root,
    MAT_KDNODE nd,
    int i,
    int dim,
    MAT_KDNODE * best,
    mtype * best_dist )
```

Here is the caller graph for this function:

6.35.2.18 __mat_lint()

```

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 fl4,
    mtype hmin,
    mtype hmax,
    mtype re,
    mtype ae )

```

Here is the caller graph for this function:

6.35.2.19 __mat_logplusone()

```

mtype __mat_logplusone (
    mtype x )

```

Computes logarithm plus one function.

Parameters

in	x	
----	---	--

Returns

$\log(1 + x)$

Here is the caller graph for this function:

6.35.2.20 __mat_mds_metric()

```

MATRIX __mat_mds_metric (
    MATRIX d,
    int dims,
    MATRIX result )

```

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.21 __mat_mds_nonmetric()

```

MATRIX __mat_mds_nonmetric (
    MATRIX d,
    int dims,
    MATRIX result )

```

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.22 __mat_mulfunc()

```

mtype __mat_mulfunc (
    mtype x,
    mtype y )

```

Computes multiplication function.

Parameters

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

Returns

xy

Here is the caller graph for this function:

6.35.2.23 __mat_pol2cart()

```

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

```

6.35.2.24 __mat_powerof2()

```

int __mat_powerof2 (
    int width,
    int * m,
    int * twopm )

```

Here is the caller graph for this function:

6.35.2.25 __mat_quicksort()

```

void __mat_quicksort (
    MATRIX A,
    int l,
    int r,
    int offset,
    MATRIX ind )

```

Here is the caller graph for this function:

6.35.2.26 __mat_rand()

```
mtype __mat_rand (
    void )
```

Here is the call graph for this function:

6.35.2.27 __mat_randexp()

```
mtype __mat_randexp (
    mtype mu )
```

Here is the call graph for this function:

6.35.2.28 __mat_randfun()

```
mtype __mat_randfun (
    mtype(*) (mtype) fun,
    mtype xmin,
    mtype xmax )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.29 __mat_randn()

```
mtype __mat_randn (
    void )
```

Here is the call graph for this function:

6.35.2.30 __mat_round_away_zero()

```
mtype __mat_round_away_zero (
    mtype x )
```

Rounds a number away from zero.

Parameters

in	x	Input value
----	-----	-------------

Returns

$\text{sgn}(x) \lfloor |x| + 0.5 \rfloor$

6.35.2.31 __mat_round_towards_zero()

```

mtype __mat_round_towards_zero (
    mtype x )

```

Rounds a number towards zero.

Parameters

in	x	Input value
----	-----	-------------

Returns

$$\text{sgn}(x) \lceil |x| - 0.5 \rceil$$

6.35.2.32 __mat_sqrfunc()

```

mtype __mat_sqrfunc (
    mtype x )

```

Computes square function.

Parameters

in	x	
----	-----	--

Returns

$$x^2$$

Here is the caller graph for this function:

6.35.2.33 __mat_sqrtfunc()

```

mtype __mat_sqrtfunc (
    mtype x )

```

Computes square root function.

Parameters

in	x	
----	-----	--

Returns

$$\sqrt{x}$$

Here is the caller graph for this function:

6.35.2.34 __mat_subfunc()

```
mttype __mat_subfunc (
    mttype x,
    mttype y )
```

Computes subtraction function.

Parameters

in	x	
in	y	

Returns

$x - y$

Here is the caller graph for this function:

6.35.2.35 __matstack_creat()

```
MATSTACK __matstack_creat (
    int len )
```

Here is the caller graph for this function:

6.35.2.36 gen_eq()

```
int gen_eq (
    mttype a )
```

Checks if equals to zero.

Parameters

in	a	Input value
----	-----	-------------

Returns

`int $a == 0$`

6.35.2.37 gen_error()

```
int gen_error (
    int err_ )
```

Generates error message for general errors and exits.

Parameters

in	err	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)
----	-----	---

Here is the caller graph for this function:

6.35.2.38 gen_gt()

```
int gen_gt (
    mtype a )
```

Checks if greater than zero.

Parameters

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

Returns

$\text{int } a > 0$

6.35.2.39 gen_lt()

```
int gen_lt (
    mtype a )
```

Checks if less than zero.

Parameters

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

Returns

$\text{int } a < 0$

6.35.2.40 graph_error()

```
int graph_error (
    int err_ )
```

Generates error message for graph errors and exits.

Parameters

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

Here is the caller graph for this function:

6.35.2.41 int_vec2_mat()

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

Converts an integer vector to a matrix.

Parameters

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

Returns

Output matrix

Here is the call graph for this function:

6.35.2.42 int_vec_abs()

```
INT_VECTOR int_vec_abs (  
    INT_VECTOR A,  
    INT_VECTOR result )
```

Computes absolute value of an integer vector.

Parameters

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

Returns

$\text{abs}(A)$

Here is the call graph for this function:

6.35.2.43 int_vec_add()

```
INT_VECTOR int_vec_add (  
    INT_VECTOR A,
```

```

    INT_VECTOR B,
    INT_VECTOR result )

```

Adds two integer vectors.

Parameters

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

Returns

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

Here is the call graph for this function:

6.35.2.44 int_vec_adds()

```

INT_VECTOR int_vec_adds (
    INT_VECTOR A,
    int s,
    INT_VECTOR result )

```

Adds an integer to an integer vector.

Parameters

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

Returns

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

Here is the call graph for this function:

6.35.2.45 int_vec_append()

```

INT_VECTOR int_vec_append (
    INT_VECTOR a,
    int i )

```

Appends an integer to an integer vector.

Parameters

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

Returns

Appended vector

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.46 int_vec_concat()

```
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>result</i>	Vector to store the result

Returns

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

Here is the call graph for this function:

6.35.2.47 int_vec_copy()

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

Copies an integer vector.

Parameters

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

Returns

Output vector

Here is the call graph for this function:

6.35.2.48 `int_vec_creat()`

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

Creates an integer vector.

Parameters

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

Returns

Output vector

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.49 `int_vec_div()`

```
INT_VECTOR int_vec_div (
    INT_VECTOR A,
    INT_VECTOR B,
    INT_VECTOR result )
```

Computes element-wise integer vector division.

Parameters

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

Returns

$A./B$

Here is the call graph for this function:

6.35.2.50 `int_vec_divs()`

```
INT_VECTOR int_vec_divs (
    INT_VECTOR A,
    int s,
    INT_VECTOR result )
```

Divides an integer vector by a scalar.

Parameters

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

Returns

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

Here is the call graph for this function:

6.35.2.51 int_vec_divs_inv()

```
INT_VECTOR int_vec_divs_inv (  
    INT_VECTOR A,  
    int s,  
    INT_VECTOR result )
```

Divides a scalar by an integer vector.

Parameters

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

Returns

$$s1./A$$

Here is the call graph for this function:

6.35.2.52 int_vec_dump()

```
void int_vec_dump (  
    INT_VECTOR A )
```

Dumps an integer vector in the stdout.

Parameters

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

Here is the call graph for this function:

6.35.2.53 `int_vec_dumpf()`

```
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

Here is the call graph for this function:

6.35.2.54 `int_vec_error()`

```
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_FNOTGETINT_VEC/INT_VEC_C_SIZEMISMATCH)
----	------------	--

Here is the caller graph for this function:

6.35.2.55 `int_vec_fdump()`

```
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

Here is the call graph for this function:

6.35.2.56 `int_vec_fdumpf()`

```
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

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

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.57 int_vec_fill()

```
INT_VECTOR int_vec_fill (
    INT_VECTOR A,
    int val )
```

Fills an integer vector with a value.

Parameters

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

Returns

Filled vector

6.35.2.58 int_vec_fill_type()

```
INT_VECTOR int_vec_fill_type (
    INT_VECTOR A,
    int type )
```

Fills an integer vector to a type.

Parameters

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

Returns

Filled vector

Here is the caller graph for this function:

6.35.2.59 `int_vec_find()`

```
INT_VECTOR int_vec_find (
    INT_VECTOR a,
    int rel_type,
    int n )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.60 `int_vec_free()`

```
int int_vec_free (
    INT_VECTOR A )
```

Frees an integer vector.

Parameters

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

Returns

Success

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.61 `int_vec_inv()`

```
INT_VECTOR int_vec_inv (
    INT_VECTOR A,
    INT_VECTOR result )
```

Computes element-wise integer vector inverse.

Parameters

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

Returns

$1./A$

Here is the call graph for this function:

6.35.2.62 `int_vec_mean()`

```
mtype int_vec_mean (
    INT_VECTOR A )
```

Computes element-mean of an integer vector.

Parameters

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

Returns

$\text{mean}(A)$

6.35.2.63 int_vec_mul()

```
INT_VECTOR int_vec_mul (
    INT_VECTOR A,
    INT_VECTOR B,
    INT_VECTOR result )
```

Computes element-wise integer vector multiplication.

Parameters

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

Returns

$A * B$

Here is the call graph for this function:

6.35.2.64 int_vec_muls()

```
INT_VECTOR int_vec_muls (
    INT_VECTOR A,
    int x,
    INT_VECTOR result )
```

Multiplies an integer vector by a scalar.

Parameters

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

Returns sA

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.65 int_vec_permute_vect()

```
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

Here is the call graph for this function:

6.35.2.66 int_vec_randperm()

```
INT_VECTOR int_vec_randperm (
    int n,
    INT_VECTOR result )
```

Here is the call graph for this function:

6.35.2.67 int_vec_sub()

```
INT_VECTOR int_vec_sub (
    INT_VECTOR A,
    INT_VECTOR B,
    INT_VECTOR result )
```

Subtracts an integer vector from integer vector.

Parameters

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

Returns

$$\mathbf{A} - \mathbf{B}$$

Here is the call graph for this function:

6.35.2.68 int_vec_subs()

```
INT_VECTOR int_vec_subs (
    INT_VECTOR A,
    int s,
    INT_VECTOR result )
```

Subtracts an integer from integer vector.

Parameters

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

Returns

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

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.69 int_vec_subs_neg()

```
INT_VECTOR int_vec_subs_neg (
    INT_VECTOR A,
    int s,
    INT_VECTOR result )
```

Subtracts an integer vector from an integer.

Parameters

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

Returns

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

Here is the call graph for this function:

6.35.2.70 int_vec_sum()

```
int int_vec_sum (
    INT_VECTOR A )
```

Computes element-sum of an integer vector.

Parameters

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

Returns

sum(*A*)

6.35.2.71 int_vec_unique()

```
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

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.72 int_vecstack_creat()

```
INT_VECSTACK int_vecstack_creat (
    int len )
```

Creates an integer vector stack.

Parameters

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

Returns

Output vector stack

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.73 int_vecstack_error()

```
INT_VECSTACK int_vecstack_error (
    int err_ )
```

Generates error message for integer vector stack errors and exits.

Parameters

in	err	Error type (INT_VECSTACK_MALLOC/INT_VECSTACK_FNOTOPEN/INT_VECSTACK_FNOT↔GETINT_VEC/INT_VECSTACK_SIZEMISMATCH)
----	-----	---

Here is the caller graph for this function:

6.35.2.74 int_vecstack_free()

```
int int_vecstack_free (
    INT_VECSTACK A )
```

Frees an integer vector stack.

Parameters

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

Returns

Success

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.75 mat_2int_vec()

```
INT_VECTOR mat_2int_vec (
    MATRIX A )
```

Converts a matrix to an integer vector.

Parameters

in	A	Input matrix
in	v	Vector to store the result

Returns

Output vector

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.76 mat_abs()

```
MATRIX mat_abs (
    MATRIX A,
    MATRIX result )
```

Computes absolute value of matrix.

Parameters

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

Returns

$\text{abs}(\mathbf{A})$

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.77 mat_add()

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

Adds two matrices.

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

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

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.78 mat_adds()

```
MATRIX mat_adds (
    MATRIX A,
```

```

    mtype s,
    MATRIX result )

```

Adds a scalar to a matrix.

Parameters

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

Returns

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

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.79 mat_backsubs1()

```

void mat_backsubs1 (
    MATRIX A,
    MATRIX B,
    MATRIX C,
    MATRIX P,
    int xcol )

```

Here is the caller graph for this function:

6.35.2.80 mat_bayes_classifier_test()

```

INT_VECTOR mat_bayes_classifier_test (
    MATRIX data,
    MAT_BAYES_MODEL b_model )

```

Here is the call graph for this function:

6.35.2.81 mat_bayes_classifier_train()

```

MAT_BAYES_MODEL mat_bayes_classifier_train (
    MATRIX data,
    INT_VECTOR labels )

```

Here is the call graph for this function:

6.35.2.82 mat_bayes_model_creat()

```

MAT_BAYES_MODEL mat_bayes_model_creat (
    void )

```

Creates a Bayes model.

Returns

Output Bayes model

Here is the caller graph for this function:

6.35.2.83 mat_bayes_model_free()

```
int mat_bayes_model_free (
    MAT_BAYES_MODEL a )
```

Frees a Bayes model.

Parameters

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

Returns

Success

Here is the call graph for this function:

6.35.2.84 mat_binom()

```
mtype mat_binom (
    int n,
    int k )
```

Computes a binomial co-efficient.

Parameters

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

Returns

$\binom{n}{k}$

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.85 mat_binom_init()

```
void mat_binom_init ( )
```

Initializes the binomial series.

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.86 mat_bisquare_wt()

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

Computes bisquare weight function element-wise on a matrix.

Parameters

in	A	Input matrix
in	k	Bisquare parameter

Returns

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

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.87 mat_bs_delete()

```
MAT_TREE mat_bs_delete (
    mtype x,
    MAT_TREE T )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.88 mat_bs_find()

```
MAT_TREE mat_bs_find (
    mtype x,
    MAT_TREE T )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.89 mat_bs_find_max()

```
MAT_TREE mat_bs_find_max (
    MAT_TREE T )
```

6.35.2.90 mat_bs_find_min()

```
MAT_TREE mat_bs_find_min (
    MAT_TREE T )
```

Here is the caller graph for this function:

6.35.2.91 mat_bs_free()

```
MAT_TREE mat_bs_free (
    MAT_TREE T )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.92 `mat_bs_inorder()`

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

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.93 `mat_bs_insert()`

```
MAT_TREE mat_bs_insert (
    mtype x,
    MAT_TREE T )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.94 `mat_bs_make_null()`

```
MAT_TREE mat_bs_make_null (
    void )
```

Here is the caller graph for this function:

6.35.2.95 `mat_bsxfun()`

```
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

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.96 mat_calc_dist_sq()

```
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

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.97 mat_cart2pol()

```
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

Here is the call graph for this function:

6.35.2.98 mat_cheby()

```
MATRIX mat_cheby (
    int n )
```

Computes the n^{th} Chebyshev polynomial.

Parameters

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

Returns

Output polynomial matrix

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.99 mat_cheby_approx()

```
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	<i>result</i>	Matrix to store the result

Returns

Approximate polynomial matrix

Here is the call graph for this function:

6.35.2.100 mat_cheby_coeffs_to_poly()

```
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

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.101 mat_cheby_init()

```
void mat_cheby_init ( )
```

Initializes the Chebyshev polynomial series.

Here is the call graph for this function:

6.35.2.102 mat_cholesky()

```
MATRIX mat_cholesky (
    MATRIX A,
    MATRIX result )
```

Computes Cholesky factor of a matrix.

Parameters

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

Returns

Cholesky factor

Here is the call graph for this function:

6.35.2.103 mat_cofact()

```
mtype mat_cofact (
    MATRIX A,
    int i,
    int j )
```

Computes a cofactor of a matrix.

Parameters

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

ReturnsCofactor C_{ij} **6.35.2.104 mat_colcopy()**

```

MATRIX mat_colcopy (
    MATRIX A,
    int cola,
    int colb,
    MATRIX result )

```

Copies a column from a matrix.

Parameters

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

Returns

Copied matrix

Here is the caller graph for this function:

6.35.2.105 mat_concat()

```

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

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

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.106 mat_conjgrad()

```

MATRIX mat_conjgrad (
    MATRIX A,
    MATRIX b,
    MATRIX x0,
    mtype tol,
    int miter,
    MATRIX result )

```

Solves a linear system with conjugate gradients method.

Parameters

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

Returns

x

Here is the call graph for this function:

6.35.2.107 mat_conv2()

```

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

Here is the call graph for this function:

6.35.2.108 mat_copy()

```

MATRIX mat_copy (
    MATRIX A,
    MATRIX result )

```

Copies a matrix.

Parameters

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

Returns

Output matrix

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.109 mat_corcol()

```
MATSTACK mat_corcol (
    MATRIX data )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.110 mat_count_words_in_line()

```
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
in	<i>count</i>	Pointer to output count

Returns

EOF reached

Here is the caller graph for this function:

6.35.2.111 mat_covcol()

```
MATSTACK mat_covcol (
    MATRIX data )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.112 `mat_creat()`

```
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

Here is the call graph for this function:

6.35.2.113 `mat_creat_diag()`

```
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

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.114 `mat_det()`

```
mtype mat_det (
    MATRIX A )
```

Computes the determinant of a matrix.

Parameters

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

Returns
 $\det(A)$

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.115 mat_diagmul()

```
mtype mat_diagmul (
    MATRIX A )
```

Computes matrix diagonal product.

Parameters

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

Returns
 $\text{prod}(\text{diag}(\mathbf{A}))$
6.35.2.116 mat_div_dot()

```
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	<i>result</i>	Matrix to store the result

Returns
 $\mathbf{A} ./ \mathbf{B}$

Here is the call graph for this function:

6.35.2.117 mat_divs()

```
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

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

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.118 mat_divs_inv()

```
MATRIX mat_divs_inv (
    MATRIX A,
    mtype s,
    MATRIX result )
```

Divides a scalar by a matrix.

Parameters

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

Returns

$$s\mathbf{11}^T ./ \mathbf{A}$$

Here is the call graph for this function:

6.35.2.119 mat_dlmread()

```
MATRIX mat_dlmread (
    const char * fname )
```

Reads a matrix from a file.

Parameters

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

Returns

Output matrix

Here is the call graph for this function:

6.35.2.120 mat_dlmwrite()

```
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

Here is the call graph for this function:

6.35.2.121 mat_dump()

```
void mat_dump (
    MATRIX A )
```

Dumps a matrix in the stdout.

Parameters

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

Here is the call graph for this function:

6.35.2.122 mat_dumpf()

```
void mat_dumpf (
    MATRIX A,
    const char * s )
```

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

Parameters

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

Here is the call graph for this function:

6.35.2.123 mat_durbin()

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

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.124 mat_eig_sym()

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

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.125 mat_error()

```
MATRIX mat_error (
    int err_ )
```

Generates error message for matrix errors and exits.

Parameters

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

6.35.2.126 mat_fdump()

```
void mat_fdump (
    MATRIX A,
    MAT_FILEPOINTER fp )
```

Dumps a matrix in an opened file.

Parameters

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

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.127 mat_fdumpf()

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

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.128 mat_fft2()

```
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 (MAT_FFT2_FORWARD/MAT_FFT2_BACKWARD)
in	<i>result</i>	Matrix stack to store the result

Returns

Transformed matrix stack

Here is the call graph for this function:

6.35.2.129 mat_fgetmat()

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

Gets matrix data from opened file.

Parameters

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

Returns

Number of elements copied

6.35.2.130 mat_fill()

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

Fills a matrix with a value.

Parameters

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

Returns

Filled matrix

Here is the caller graph for this function:

6.35.2.131 mat_fill_type()

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

Fills a matrix to a type.

Parameters

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

Returns

Filled matrix

Here is the caller graph for this function:

6.35.2.132 mat_find()

```
INT_VECSTACK mat_find (
    MATRIX A,
    int rel_type,
    mtype x )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.133 mat_find_within_dist()

```
INT_VECTOR mat_find_within_dist (
    MATRIX A,
    MATRIX d,
    mtype range )
```

Finds points within a neighborhood.

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>range</i>	Radius to search within

Returns

Indices Vector

Here is the call graph for this function:

6.35.2.134 mat_fliplr()

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

Here is the call graph for this function:

6.35.2.135 mat_flipud()

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

Here is the call graph for this function:

6.35.2.136 mat_fnextline()

```
void mat_fnextline (  
    MAT_FILEPOINTER fp )
```

Prints nextline to file.

Parameters

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

Here is the caller graph for this function:

6.35.2.137 mat_free()

```
int mat_free (  
    MATRIX A )
```

Frees a matrix.

Parameters

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

Returns

Success

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.138 mat_get_sub_matrix_from_cols()

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

Extracts sub-matrix from columns of a matrix.

Parameters

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

Returns

Extracted matrix

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.139 mat_get_sub_matrix_from_rows()

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

Extracts sub-matrix from rows of a matrix.

Parameters

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

Returns

Extracted matrix

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.140 mat_get_sub_vector()

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

Extracts sub-vector from an integer vector.

Parameters

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

Returns

Extracted vector

Here is the call graph for this function:

6.35.2.141 mat_gfunc()

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

Computes a given function element-wise on a matrix.

Parameters

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

Returns

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

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.142 mat_go_next_word()

```
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

6.35.2.143 mat_graph_adjlist()

```
void mat_graph_adjlist (
    MAT_GRAPH g,
    int directed,
    int weighted,
    MAT_FILEPOINTER fp )
```

Here is the call graph for this function:

6.35.2.144 mat_graph_adjm_to_adjl()

```
void mat_graph_adjm_to_adjl (  
    MAT_GRAPH g,  
    MATRIX a )
```

Here is the call graph for this function:

6.35.2.145 mat_graph_creat()

```
MAT_GRAPH mat_graph_creat (  
    void )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.146 mat_graph_dump()

```
void mat_graph_dump (  
    MAT_GRAPH g,  
    int mst )
```

Here is the call graph for this function:

6.35.2.147 mat_graph_dumpf()

```
void mat_graph_dumpf (  
    MAT_GRAPH g,  
    int mst,  
    MAT_FILEPOINTER fp )
```

Here is the caller graph for this function:

6.35.2.148 mat_graph_reverse()

```
MAT_GRAPH mat_graph_reverse (  
    MAT_GRAPH g,  
    MAT_GRAPH r )
```

Here is the call graph for this function:

6.35.2.149 mat_graph_search()

```
MAT_INT_QUEUE mat_graph_search (  
    MAT_GRAPH g,  
    int connected,  
    int mst )
```

Here is the call graph for this function:

6.35.2.150 mat_graph_visit()

```
void mat_graph_visit (
    MAT_GRAPH g,
    int k,
    int connected,
    int mst,
    MAT_INT_PRIORITYQUEUE pq,
    MAT_INT_QUEUE q )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.151 mat_huber_wt()

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

Computes Huber weight function element-wise on a matrix.

Parameters

in	A	Input matrix
in	k	Huber parameter

Returns

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

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.152 mat_innerprod()

```
mtype mat_innerprod (
    MATRIX A,
    MATRIX B )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.153 mat_int_priorityqueue_creat()

```
MAT_INT_PRIORITYQUEUE mat_int_priorityqueue_creat (
    int type )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.154 mat_int_priorityqueue_dequeue()

```
mat_intpqnode mat_int_priorityqueue_dequeue (
    MAT_INT_PRIORITYQUEUE H )
```

Here is the call graph for this function: [Here is the caller graph for this function:](#)

6.35.2.155 mat_int_priorityqueue_enqueue()

```
void mat_int_priorityqueue_enqueue (
    MAT_INT_PRIORITYQUEUE H,
    int data,
    int priority )
```

Here is the call graph for this function: [Here is the caller graph for this function:](#)

6.35.2.156 mat_int_priorityqueue_free()

```
int mat_int_priorityqueue_free (
    MAT_INT_PRIORITYQUEUE H )
```

Here is the caller graph for this function: [Here is the caller graph for this function:](#)

6.35.2.157 mat_int_priorityqueue_is_empty()

```
int mat_int_priorityqueue_is_empty (
    MAT_INT_PRIORITYQUEUE H )
```

6.35.2.158 mat_int_priorityqueue_update()

```
int mat_int_priorityqueue_update (
    MAT_INT_PRIORITYQUEUE H,
    int data,
    int priority,
    int type )
```

Here is the call graph for this function: [Here is the caller graph for this function:](#)

6.35.2.159 mat_int_qadrat()

```
mtype mat_int_qadrat (
    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

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

Here is the call graph for this function:

6.35.2.160 mat_int_queue_creat()

```
MAT_INT_QUEUE mat_int_queue_creat (
    void )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.161 mat_int_queue_dequeue()

```
int mat_int_queue_dequeue (
    MAT_INT_QUEUE s )
```

Here is the call graph for this function:

6.35.2.162 mat_int_queue_enqueue()

```
void mat_int_queue_enqueue (
    MAT_INT_QUEUE s,
    int value )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.163 mat_int_queue_free()

```
int mat_int_queue_free (
    MAT_INT_QUEUE s )
```

6.35.2.164 mat_int_queue_is_empty()

```
int mat_int_queue_is_empty (
    MAT_INT_QUEUE s )
```

6.35.2.165 mat_int_simpson()

```
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

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

6.35.2.166 mat_int_stack_creat()

```
MAT_INT_STACK mat_int_stack_creat (
    void )
```

Here is the call graph for this function:

6.35.2.167 mat_int_stack_free()

```
int mat_int_stack_free (
    MAT_INT_STACK s )
```

6.35.2.168 mat_int_stack_is_empty()

```
int mat_int_stack_is_empty (
    MAT_INT_STACK s )
```

6.35.2.169 mat_int_stack_pop()

```
int mat_int_stack_pop (
    MAT_INT_STACK s )
```

Here is the call graph for this function:

6.35.2.170 mat_int_stack_push()

```
void mat_int_stack_push (
    MAT_INT_STACK s,
    int value )
```

Here is the call graph for this function:

6.35.2.171 mat_int_trapezoid()

```
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

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

6.35.2.172 mat_inv()

```
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

$$A^{-1}$$

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.173 mat_inv_dot()

```
MATRIX mat_inv_dot (
    MATRIX A,
    MATRIX result )
```

Computes element-wise matrix inverse.

Parameters

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

Returns

$$\mathbf{1}\mathbf{1}^T ./ \mathbf{A}$$

Here is the call graph for this function:

6.35.2.174 `mat_isnumeric()`

```
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

Here is the caller graph for this function:

6.35.2.175 `mat_kdtree_free()`

```
int mat_kdtree_free (
    MAT_KDTREE t )
```

Frees a k-d tree.

Parameters

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

Returns

Success

Here is the call graph for this function:

6.35.2.176 `mat_kdtree_k_nearest()`

```
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 $j = 1, 2, \dots, N$

Here is the call graph for this function:

6.35.2.177 mat_kdtree_make_tree()

```
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	<i>result</i>	K-d tree to store the result

Returns

Output k-d tree

Here is the call graph for this function:

6.35.2.178 mat_kdtree_nearest()

```
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	<i>result</i>	Matrix to store the result

Returns

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

Here is the call graph for this function:

6.35.2.179 mat_kmeans()

```
MATVEC_DPOINTER mat_kmeans (
    MATRIX data,
```

```

    int k,
    int iters,
    MATVEC_DPOINTER result )

```

Here is the call graph for this function:

6.35.2.180 mat_least_squares()

```

MATRIX mat_least_squares (
    MATRIX A,
    MATRIX Y,
    MATRIX result )

```

Solves linear equations using least squares.

Parameters

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

Returns

$$(\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{Y}$$

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.181 mat_legendre()

```

MATRIX mat_legendre (
    int n )

```

Computes the n^{th} Legendre polynomial.

Parameters

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

Returns

Output polynomial matrix

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.182 mat_legendre_init()

```

void mat_legendre_init ( )

```

Initializes the Legendre polynomial series.

Here is the call graph for this function:

6.35.2.183 mat_linear_ls_fit()

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

Performs 2-d polynomial model fitting using least squares.

Parameters

in	<i>A</i>	Input data column matrix
in	<i>Y</i>	Input observation column matrix
in	<i>deg</i>	Polynomial degree N
in	<i>result</i>	Matrix to store the result

Returns

Polynomial co-efficient matrix $[\alpha_N \ \cdots \ \alpha_0]^T$

Here is the call graph for this function:

6.35.2.184 mat_solve()

```
MATRIX mat_solve (
    MATRIX A,
    MATRIX b,
    MATRIX result )
```

Solves linear equations $\mathbf{Ax} = \mathbf{b}$.

Parameters

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

Returns

Output matrix **x**

Here is the call graph for this function:

6.35.2.185 mat_solve_durbin()

```
MATRIX mat_solve_durbin (
    MATRIX A,
    MATRIX B,
    MATRIX result )
```

Runs Levinson-Durbin algorithm.

Parameters

in	A	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	B	Input correlation matrix $B = \begin{bmatrix} r_1 \\ r_2 \\ \cdots \\ r_n \end{bmatrix}$
in	<i>result</i>	Matrix to store the result

Returns

X where $RX = B$

Here is the call graph for this function:

6.35.2.186 mat_lu()

```
int mat_lu (
    MATRIX A,
    MATRIX P )
```

Computes LU decomposition of a matrix.

Parameters

in	A	Input matrix overwritten by matrices L and U
in	P	Matrix to store permutation matrix P

Returns

p Status

Here is the caller graph for this function:

6.35.2.187 mat_max()

```
MATVEC_DPOINTER mat_max (
    MATRIX A,
    int dim )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.188 mat_mds()

```
MATRIX mat_mds (
    MATRIX d,
    int dims,
    int type,
    MATRIX result )
```

Here is the call graph for this function:

6.35.2.189 mat_mean()

```
mtype mat_mean (
    MATRIX A )
```

Computes the mean of a matrix.

Parameters

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

Returns

$\text{mean}(\mathbf{A})$

Here is the call graph for this function:

6.35.2.190 mat_mean_col()

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

Computes column-mean of a matrix.

Parameters

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

Returns

$\mathbf{1}^T \mathbf{A} / \# \text{rows}$

Here is the call graph for this function:

6.35.2.191 mat_mean_row()

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

Computes row-mean of a matrix.

Parameters

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

Returns

$\mathbf{A} \mathbf{1} / \# \text{cols}$

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.192 mat_median()

```
mtype mat_median (
    MATRIX A )
```

Computes the median of elements of a given matrix.

Parameters

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

Returns

$\text{med}(\{a_{ij}\})$

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.193 mat_min()

```
MATVEC_DPOINTER mat_min (
    MATRIX A,
    int dim )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.194 mat_minor()

```
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}

Here is the call graph for this function:

6.35.2.195 mat_mtype_priorityqueue_creat()

```
MAT_MTYPE_PRIORITYQUEUE mat_mtype_priorityqueue_creat (
    int type )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.196 mat_mtype_priorityqueue_dequeue()

```
mat_mtypepqnode mat_mtype_priorityqueue_dequeue (
    MAT_MTYPE_PRIORITYQUEUE H )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.197 mat_mtype_priorityqueue_enqueue()

```
void mat_mtype_priorityqueue_enqueue (
    MAT_MTYPE_PRIORITYQUEUE H,
    mtype data,
    mtype priority )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.198 mat_mtype_priorityqueue_free()

```
int mat_mtype_priorityqueue_free (
    MAT_MTYPE_PRIORITYQUEUE H )
```

Here is the caller graph for this function:

6.35.2.199 mat_mtype_priorityqueue_is_empty()

```
int mat_mtype_priorityqueue_is_empty (
    MAT_MTYPE_PRIORITYQUEUE H )
```

6.35.2.200 mat_mtype_priorityqueue_update()

```
int mat_mtype_priorityqueue_update (
    MAT_MTYPE_PRIORITYQUEUE H,
    mtype data,
    mtype priority,
    int type )
```

Here is the call graph for this function:

6.35.2.201 mat_mtype_queue_creat()

```
MAT_MTYPE_QUEUE mat_mtype_queue_creat (
    void )
```

Here is the call graph for this function:

6.35.2.202 mat_mtype_queue_dequeue()

```
mtype mat_mtype_queue_dequeue (
    MAT_MTYPE_QUEUE s )
```

Here is the call graph for this function:

6.35.2.203 mat_mtype_queue_enqueue()

```
void mat_mtype_queue_enqueue (
    MAT_MTYPE_QUEUE s,
    mtype value )
```

Here is the call graph for this function:

6.35.2.204 mat_mtype_queue_free()

```
int mat_mtype_queue_free (
    MAT_MTYPE_QUEUE s )
```

6.35.2.205 mat_mtype_queue_is_empty()

```
int mat_mtype_queue_is_empty (
    MAT_MTYPE_QUEUE s )
```


6.35.2.206 mat_mtype_stack_creat()

```
MAT_MTYPE_STACK mat_mtype_stack_creat (
    void )
```

Here is the call graph for this function:

6.35.2.207 mat_mtype_stack_free()

```
int mat_mtype_stack_free (
    MAT_MTYPE_STACK s )
```

6.35.2.208 mat_mtype_stack_is_empty()

```
int mat_mtype_stack_is_empty (
    MAT_MTYPE_STACK s )
```

6.35.2.209 mat_mtype_stack_pop()

```
mtype mat_mtype_stack_pop (
    MAT_MTYPE_STACK s )
```

Here is the call graph for this function:

6.35.2.210 mat_mtype_stack_push()

```
void mat_mtype_stack_push (
    MAT_MTYPE_STACK s,
    mtype value )
```

Here is the call graph for this function:

6.35.2.211 mat_mul()

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

Computes matrix multiplication.

Parameters

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

Returns **AB**

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.212 mat_mul_dot()

```
MATRIX mat_mul_dot (
    MATRIX A,
    MATRIX B,
    MATRIX result )
```

Computes element-wise matrix multiplication.

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 **$A * B$**

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.213 mat_mul_fast()

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

Computes fast matrix multiplication (not implemented)

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 **AB**

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.214 mat_muls()

```
MATRIX mat_muls (
    MATRIX A,
    mtype s,
    MATRIX result )
```

Multiplies 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

$s\mathbf{A}$

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.215 mat_nextline()

```
void mat_nextline (
    void )
```

Prints nextline to stdout.

Here is the call graph for this function:

6.35.2.216 mat_norm_inf()

```
mtype mat_norm_inf (
    MATRIX A )
```

6.35.2.217 mat_norm_one()

```
mtype mat_norm_one (
    MATRIX A )
```

Here is the caller graph for this function:

6.35.2.218 mat_norm_p()

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

Here is the caller graph for this function:

6.35.2.219 mat_omp()

```

MATSTACK mat_omp (
    MATRIX A,
    MATRIX b,
    int k,
    mtype tol,
    MATSTACK result )

```

Here is the call graph for this function:

6.35.2.220 mat_order_statistic()

```

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

$O_k(\{a_{ij}\})$

Here is the call graph for this function:

6.35.2.221 mat_pca()

```

MATSTACK mat_pca (
    MATRIX data,
    int pca_type )

```

Here is the call graph for this function:

6.35.2.222 mat_perceptron_creat()

```

MAT_PERCEPTRON mat_perceptron_creat (
    void )

```

Creates a perceptron.

Returns

Output perceptron

Here is the caller graph for this function:

6.35.2.223 mat_perceptron_free()

```
int mat_perceptron_free (  
    MAT_PERCEPTRON a )
```

Frees a perceptron.

Parameters

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

Returns

Success

Here is the call graph for this function:

6.35.2.224 mat_perceptron_test()

```
INT_VECTOR mat_perceptron_test (
    MATRIX data,
    MAT_PERCEPTRON p_model )
```

Here is the call graph for this function:

6.35.2.225 mat_perceptron_train()

```
MAT_PERCEPTRON mat_perceptron_train (
    MATRIX data,
    INT_VECTOR labels,
    int num_of_iterations )
```

Here is the call graph for this function:

6.35.2.226 mat_perceptron_train_()

```
MAT_PERCEPTRON mat_perceptron_train_ (
    MATRIX data1,
    MATRIX data2,
    MAT_PERCEPTRON p_model,
    int class_num )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.227 mat_pick_col()

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

Picks a column from a matrix.

Parameters

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

Returns

Column matrix

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.228 mat_pick_row()

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

Picks a row from a matrix.

Parameters

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

Returns

Row matrix

Here is the call graph for this function:

6.35.2.229 mat_pinv()

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

Computes pseudo-inverse of a matrix.

Parameters

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

Returns

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

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.230 mat_pol2cart()

```
MATRIX mat_pol2cart (
    MATRIX A,
```

```
int dim,
MATRIX result )
```

Converts polar co-ordinates to Cartesian co-ordinates.

Parameters

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

Returns

Cartesian co-ordinate matrix

Here is the call graph for this function:

6.35.2.231 mat_poly_add()

```
MATRIX mat_poly_add (
    MATRIX A,
    MATRIX B,
    MATRIX result )
```

Adds 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

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.232 mat_poly_diff()

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

Computes derivative polynomial of a polynomial.

Parameters

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

Returns

Output matrix

Here is the call graph for this function:

6.35.2.233 mat_poly_diff_eval()

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

Evaluates derivative polynomial at a point.

Parameters

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

Returns

Output matrix

Here is the call graph for this function:

6.35.2.234 mat_poly_div()

```
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

Here is the call graph for this function:

6.35.2.235 `mat_poly_eval()`

```

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

Here is the call graph for this function:

6.35.2.236 `mat_poly_mul()`

```

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

Here is the call graph for this function:

6.35.2.237 `mat_poly_scale()`

```

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

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.238 mat_poly_shift()

```
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

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.239 mat_qr()

```
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

Here is the call graph for this function:

6.35.2.240 mat_qsort()

```
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)
in	<i>result</i>	Matrix stack to store the result

Returns

Output matrix stack of sorted A and their positions

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.241 mat_rand()

```
MATRIX mat_rand (
    int r,
    int c,
    MATRIX result )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.242 mat_randexp()

```
MATRIX mat_randexp (
    int r,
    int c,
    mtype mu,
    MATRIX result )
```

Here is the call graph for this function:

6.35.2.243 mat_randfun()

```

MATRIX mat_randfun (
    int r,
    int c,
    mtype(*) (mtype) fun,
    mtype xmin,
    mtype xmax,
    MATRIX result )

```

Here is the call graph for this function:

6.35.2.244 mat_randn()

```

MATRIX mat_randn (
    int r,
    int c,
    MATRIX result )

```

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.245 mat_randperm()

```

MATRIX mat_randperm (
    int m,
    int n,
    MATRIX result )

```

Here is the call graph for this function:

6.35.2.246 mat_randperm_n()

```

MATRIX mat_randperm_n (
    int n,
    MATRIX result )

```

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.247 mat_read_word()

```

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
in	<i>c_word</i>	Pointer to word read

Returns

EOF reached

Here is the caller graph for this function:

6.35.2.248 mat_reg_inv()

```
MATRIX mat_reg_inv (
    MATRIX A,
    mtype r,
    MATRIX result )
```

Computes the regularized inverse of a matrix.

Parameters

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

Returns

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

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.249 mat_rob_least_squares()

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

Solves linear equations using robust reweighted least squares.

Parameters

in	<i>A</i>	Input data matrix
in	<i>Y</i>	Input observation matrix
in	<i>lossfunc</i>	Loss function type (MAT_LOSS_BISQUARE/MAT_LOSS_HUBER)
in	<i>result</i>	Matrix to store the result

Returns

Robust **X**

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.250 mat_robust_fit()

```

MATRIX mat_robust_fit (
    MATRIX A,
    MATRIX Y,
    int deg,
    int lossfunc,
    MATRIX result )

```

Performs 2-d polynomial model fitting using robust least squares.

Parameters

in	<i>A</i>	Input data column matrix
in	<i>Y</i>	Input observation column matrix
in	<i>deg</i>	Polynomial degree N
in	<i>lossfunc</i>	Loss function type (MAT_LOSS_BISQUARE/MAT_LOSS_HUBER)
in	<i>result</i>	Matrix to store the result

Returns

Polynomial co-efficient matrix $[\alpha_N \ \cdots \ \alpha_0]^T$

Here is the call graph for this function:

6.35.2.251 mat_rowcopy()

```

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

6.35.2.252 mat_scpcol()

```
MATRIX mat_scpcol (
    MATRIX data )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.253 mat_set_seed()

```
void mat_set_seed (
    int seed )
```

Here is the caller graph for this function:

6.35.2.254 mat_sub()

```
MATRIX mat_sub (
    MATRIX A,
    MATRIX B,
    MATRIX result )
```

Subtracts a matrix from another matrix.

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

$A - B$

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.255 mat_submat()

```
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

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.256 mat_subs()

```
MATRIX mat_subs (
    MATRIX A,
    mtype s,
    MATRIX result )
```

Subtracts a scalar from a matrix.

Parameters

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

Returns

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

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.257 mat_subs_neg()

```
MATRIX mat_subs_neg (
    MATRIX A,
    mtype s,
    MATRIX result )
```

Subtracts a matrix from a scalar.

Parameters

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

Returns

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

Here is the call graph for this function:

6.35.2.258 `mat_sum()`

```
mtype mat_sum (
    MATRIX A )
```

Computes element-sum of a matrix.

Parameters

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

Returns

$\text{sum}(\mathbf{A})$

Here is the caller graph for this function:

6.35.2.259 `mat_sum_col()`

```
MATRIX mat_sum_col (
    MATRIX A,
    MATRIX result )
```

Computes column-sum of a matrix.

Parameters

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

Returns

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

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.260 `mat_sum_row()`

```
MATRIX mat_sum_row (
    MATRIX A,
    MATRIX result )
```

Computes row-sum of a matrix.

Parameters

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

Returns

A1

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.261 mat_svd()

```
MATSTACK mat_svd (  
    MATRIX a,  
    int niters,  
    MATSTACK result )
```

Computes the SVD of a matrix.

Parameters

in	<i>a</i>	Input matrix
in	<i>niters</i>	Iterations to use
	<i>result</i>	Matrix stack to store the result

Returns

MATSTACK (U, S, V)

Here is the call graph for this function:

6.35.2.262 mat_symtoeplz()

```
MATRIX mat_symtoeplz (  
    MATRIX R,  
    MATRIX result )
```

Computes the symmetric Toeplitz matrix from a co-efficient matrix.

Parameters

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

Returns

symtoep(R**)**

Here is the call graph for this function:

6.35.2.263 mat_tic()

```
void mat_tic (  
    void )
```

6.35.2.264 mat_toc()

```
double mat_toc (
    void )
```

Computes elapsed time from last start of timer.

Returns

Elapsed time

6.35.2.265 mat_toc_print()

```
void mat_toc_print (
    void )
```

Computes and prints elapsed time from last start of timer on the stdout.

6.35.2.266 mat_tqli()

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

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.267 mat_tran()

```
MATRIX mat_tran (
    MATRIX A,
    MATRIX result )
```

Computes the transpose of a matrix.

Parameters

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

Returns

\mathbf{A}^T

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.268 `mat_tred2()`

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

Here is the caller graph for this function:

6.35.2.269 `mat_vectorize()`

```
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

$vec(\mathbf{A})$

Here is the call graph for this function:

6.35.2.270 `mat_vectorize_tr()`

```
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

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

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.271 `mat_w_least_squares()`

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

Solves linear equations using weighted least squares.

Parameters

in	<i>A</i>	Input data matrix
in	<i>Y</i>	Input observation matrix
in	<i>w</i>	Input weight column matrix
in	<i>result</i>	Matrix to store the result

Returns

$$(\mathbf{A}^T \text{diag}(w) \mathbf{A})^{-1} \mathbf{A}^T \text{diag}(w) \mathbf{Y}$$

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.272 `mat_wpinv()`

```
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

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

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.273 `mat_xcopy()`

```
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 $A_{s_i:e_i,s_j:e_j}$

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.274 mat_xjoin()

```

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

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.275 mats_isinf()

```

int mats_isinf (
    mtype x )

```

Checks if scalar is infinite.

Parameters

in	x	Input scalar
----	-----	--------------

Returns

Zero/non-zero

6.35.2.276 mats_isnan()

```
int mats_isnan (
    mtype x )
```

Checks if scalar is NaN.

Parameters

in	x	Input scalar
----	-----	--------------

Returns

Zero/non-zero

6.35.2.277 matstack_append()

```
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

Here is the call graph for this function:

6.35.2.278 matstack_creat()

```
MATSTACK matstack_creat (
    int len )
```

Creates a matrix stack.

Parameters

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

Returns

Output matrix stack

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.279 matstack_error()

```
MATSTACK matstack_error (
    int err_ )
```

Generates error message for matrix stack errors and exits.

Parameters

in	<i>err</i>	Error type (MATSTACK_MALLOC/MATSTACK_FNOTOPEN/MATSTACK_FNOTGETMAT/MATSTACK_SIZEMISMATCH/MATSTACK_INVERSE_ERROR)
----	------------	---

Here is the caller graph for this function:

6.35.2.280 matstack_free()

```
int matstack_free (
    MATSTACK A )
```

Frees a matrix stack.

Parameters

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

Returns

Success

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.281 matvec_creat()

```
MATVEC_DPOINTER matvec_creat (
    void )
```

Creates a matrix-vector pair.

Returns

Output matrix-vector pair

Here is the caller graph for this function:

6.35.2.282 matvec_free()

```
int matvec_free (
    MATVEC_DPOINTER a )
```

Frees a matrix-vector pair.

Parameters

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

Returns

Success

Here is the call graph for this function: Here is the caller graph for this function:

6.35.2.283 pq_error()

```
int pq_error (
    int err_ )
```

Generates error message for priority queue errors and exits.

Parameters

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

Here is the caller graph for this function:

6.35.2.284 queue_error()

```
int queue_error (
    int err_ )
```

Generates error message for queue errors and exits.

Parameters

in	err	Error type (QUEUE_MALLOC/QUEUE_EMPTY)
----	-----	---------------------------------------

Here is the caller graph for this function:

6.35.2.285 stack_error()

```
int stack_error (
    int err_ )
```

Generates error message for stack errors and exits.

Parameters

in	err	Error type (STACK_MALLOC/STACK_EMPTY)
----	-----	---------------------------------------

Here is the caller graph for this function:

6.35.3 Variable Documentation

6.35.3.1 mat_binom_series_table

`MATSTACK` mat_binom_series_table

6.35.3.2 mat_cheby_series_table

`MATSTACK` mat_cheby_series_table

6.35.3.3 MAT_CLOCK_TIME

`__thread clock_t` MAT_CLOCK_TIME

6.35.3.4 mat_legendre_series_table

`MATSTACK` mat_legendre_series_table

6.35.3.5 MAT_SEED

```
unsigned int MAT_SEED
```

6.35.3.6 MAT_SET_SEED

```
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_vec_find()

```
INT\_VECTOR int_vec_find (
    INT\_VECTOR a,
    int rel_type,
    int n )
```

Here is the call graph for this function: Here is the caller graph for this function:

6.36.1.2 mat_find()

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

Here is the call graph for this function: Here is the caller graph for this function:

6.37 matsolve.c File Reference

Functions

- int [mat_lu](#) ([MATRIX](#) A, [MATRIX](#) P)
Computes LU decomposition of a matrix.
- void [mat_backsubs1](#) ([MATRIX](#) A, [MATRIX](#) B, [MATRIX](#) X, [MATRIX](#) P, int xcol)
- [MATRIX](#) [mat_lsolve](#) ([MATRIX](#) A, [MATRIX](#) b, [MATRIX](#) result)
Solves linear equations $Ax = b$.
- [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 mat_backsubs1()

```
void mat_backsubs1 (
    MATRIX A,
    MATRIX B,
    MATRIX X,
    MATRIX P,
    int xcol )
```

Here is the caller graph for this function:

6.37.1.2 mat_cholesky()

```
MATRIX mat_cholesky (
    MATRIX A,
    MATRIX result )
```

Computes Cholesky factor of a matrix.

Parameters

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

Returns

Cholesky factor

Here is the call graph for this function:

6.37.1.3 mat_conjgrad()

```

MATRIX mat_conjgrad (
    MATRIX A,
    MATRIX b,
    MATRIX x0,
    mtype tol,
    int miter,
    MATRIX result )

```

Solves a linear system with conjugate gradients method.

Parameters

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

Returns

x

Here is the call graph for this function:

6.37.1.4 mat_lsolve()

```

MATRIX mat_lsolve (
    MATRIX A,
    MATRIX b,
    MATRIX result )

```

Solves linear equations $Ax = b$.

Parameters

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

Returns

Output matrix x

Here is the call graph for this function:

6.37.1.5 mat_lu()

```

int mat_lu (
    MATRIX A,
    MATRIX P )

```

Computes LU decomposition of a matrix.

Parameters

in	A	Input matrix overwritten by matrices L and U
in	P	Matrix to store permutation matrix P

Returns

p Status

Here is the caller graph for this function:

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 mat_median()

```
mtype mat_median (
    MATRIX A )
```

Computes the median of elements of a given matrix.

Parameters

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

Returns

$\text{med}(\{a_{ij}\})$

Here is the call graph for this function: Here is the caller graph for this function:

6.38.1.2 mat_order_statistic()

```

mtype mat_order_statistic (
    MATRIX A,
    int k )

```

Computes the k^{th} order statistic of elements of a given matrix.

Parameters

in	<i>A</i>	Input matrix
in	<i>k</i>	Order

Returns

$O_k(\{a_{ij}\})$

Here is the call graph for this function:

6.38.1.3 mat_qsort()

```

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)
in	<i>result</i>	Matrix stack to store the result

Returns

Output matrix stack of sorted A and their positions

Here is the call graph for this function: Here is the caller graph for this function:

6.39 matstdrels.c File Reference

Functions

- int [gen_gt](#) (mtype a)
Checks if greater than zero.
- int [gen_lt](#) (mtype a)
Checks if less than zero.
- int [gen_eq](#) (mtype a)
Checks if equals to zero.

6.39.1 Function Documentation

6.39.1.1 gen_eq()

```
int gen_eq (
    mtype a )
```

Checks if equals to zero.

Parameters

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

Returns

`int a == 0`

6.39.1.2 gen_gt()

```
int gen_gt (
    mtype a )
```

Checks if greater than zero.

Parameters

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

Returns

`int a > 0`

6.39.1.3 gen_lt()

```
int gen_lt (
    mtype a )
```

Checks if less than zero.

Parameters

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

Returns

`int a < 0`

6.40 matsub.c File Reference

Functions

- [MATRIX mat_sub](#) ([MATRIX A](#), [MATRIX B](#), [MATRIX result](#))
Subtracts a matrix from another matrix.
- [MATRIX mat_subs](#) ([MATRIX A](#), [mtype s](#), [MATRIX result](#))
Subtracts a scalar from a matrix.
- [MATRIX mat_subs_neg](#) ([MATRIX A](#), [mtype s](#), [MATRIX result](#))
Subtracts a matrix from a scalar.
- [INT_VECTOR int_vec_sub](#) ([INT_VECTOR A](#), [INT_VECTOR B](#), [INT_VECTOR result](#))
Subtracts an integer vector from integer vector.
- [INT_VECTOR int_vec_subs](#) ([INT_VECTOR A](#), [int s](#), [INT_VECTOR result](#))
Subtracts an integer from integer vector.
- [INT_VECTOR int_vec_subs_neg](#) ([INT_VECTOR A](#), [int s](#), [INT_VECTOR result](#))
Subtracts an integer vector from an integer.

6.40.1 Function Documentation

6.40.1.1 int_vec_sub()

```
INT_VECTOR int_vec_sub (
    INT_VECTOR A,
    INT_VECTOR B,
    INT_VECTOR result )
```

Subtracts an integer vector from integer vector.

Parameters

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

Returns

A – B

Here is the call graph for this function:

6.40.1.2 int_vec_subs()

```

INT_VECTOR int_vec_subs (
    INT_VECTOR A,
    int s,
    INT_VECTOR result )

```

Subtracts an integer from integer vector.

Parameters

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

Returns

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

Here is the call graph for this function: Here is the caller graph for this function:

6.40.1.3 int_vec_subs_neg()

```

INT_VECTOR int_vec_subs_neg (
    INT_VECTOR A,
    int s,
    INT_VECTOR result )

```

Subtracts an integer vector from an integer.

Parameters

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

Returns

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

Here is the call graph for this function:

6.40.1.4 mat_sub()

```

MATRIX mat_sub (
    MATRIX A,
    MATRIX B,
    MATRIX result )

```

Subtracts a matrix from another matrix.

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

$$\mathbf{A} - \mathbf{B}$$

Here is the call graph for this function: Here is the caller graph for this function:

6.40.1.5 mat_subs()

```
MATRIX mat_subs (
    MATRIX A,
    mtype s,
    MATRIX result )
```

Subtracts a scalar from a matrix.

Parameters

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

Returns

$$\mathbf{A} - s\mathbf{11}^T$$

Here is the call graph for this function: Here is the caller graph for this function:

6.40.1.6 mat_subs_neg()

```
MATRIX mat_subs_neg (
    MATRIX A,
    mtype s,
    MATRIX result )
```

Subtracts a matrix from a scalar.

Parameters

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

Returns

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

Here is the call graph for this function:

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 mat_submat()

```
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

Here is the call graph for this function: Here is the caller graph for this function:

6.42 matsum.c File Reference

Functions

- mtype [mat_sum](#) ([MATRIX](#) A)

Computes element-sum of a matrix.

- `MATRIX mat_sum_row (MATRIX A, MATRIX result)`

Computes row-sum of a matrix.

- `MATRIX mat_sum_col (MATRIX A, MATRIX result)`

Computes column-sum of a matrix.

- `int int_vec_sum (INT_VECTOR A)`

Computes element-sum of an integer vector.

6.42.1 Function Documentation

6.42.1.1 int_vec_sum()

```
int int_vec_sum (
    INT_VECTOR A )
```

Computes element-sum of an integer vector.

Parameters

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

Returns

`sum(A)`

6.42.1.2 mat_sum()

```
mtype mat_sum (
    MATRIX A )
```

Computes element-sum of a matrix.

Parameters

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

Returns

`sum(A)`

Here is the caller graph for this function:

6.42.1.3 mat_sum_col()

```
MATRIX mat_sum_col (
    MATRIX A,
    MATRIX result )
```

Computes column-sum of a matrix.

Parameters

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

Returns

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

Here is the call graph for this function: Here is the caller graph for this function:

6.42.1.4 mat_sum_row()

```
MATRIX mat_sum_row (
    MATRIX A,
    MATRIX result )
```

Computes row-sum of a matrix.

Parameters

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

Returns

$$\mathbf{A} \mathbf{1}$$

Here is the call graph for this function: Here is the caller graph for this function:

6.43 matsvd.c File Reference

Functions

- [MATSTACK mat_svd](#) ([MATRIX](#) a, int niters, [MATSTACK](#) result)
Computes the SVD of a matrix.

6.43.1 Function Documentation

6.43.1.1 mat_svd()

```
MATSTACK mat_svd (
    MATRIX a,
    int niters,
    MATSTACK result )
```

Computes the SVD of a matrix.

Parameters

in	<i>a</i>	Input matrix
in	<i>niters</i>	Iterations to use
	<i>result</i>	Matrix stack to store the result

Returns

MATSTACK (U, S, V)

Here is the call graph for this function:

6.44 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.

6.44.1 Function Documentation

6.44.1.1 mat_count_words_in_line()

```
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
in	<i>count</i>	Pointer to output count

Returns

EOF reached

Here is the caller graph for this function:

6.44.1.2 mat_dlmread()

```
MATRIX mat_dlmread (
    const char * fname )
```

Reads a matrix from a file.

Parameters

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

Returns

Output matrix

Here is the call graph for this function:

6.44.1.3 mat_dlmwrite()

```
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

Here is the call graph for this function:

6.44.1.4 mat_go_next_word()

```
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

6.44.1.5 mat_isnumeric()

```
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

Here is the caller graph for this function:

6.44.1.6 mat_read_word()

```
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
in	<i>c_word</i>	Pointer to word read

Returns

EOF reached

Here is the caller graph for this function:

6.45 mattimers.c File Reference

Functions

- [__declspec](#) (thread)
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

- `_Thread_local clock_t` [MAT_CLOCK_TIME](#)

6.45.1 Function Documentation

6.45.1.1 [__declspec\(\)](#)

```
__declspec (  
    thread )
```

Starts stopwatch timer.

6.45.1.2 [mat_toc\(\)](#)

```
double mat_toc (  
    void )
```

Computes elapsed time from last start of timer.

Returns

Elapsed time

6.45.1.3 [mat_toc_print\(\)](#)

```
void mat_toc_print (  
    void )
```

Computes and prints elapsed time from last start of timer on the stdout.

6.45.2 Variable Documentation

6.45.2.1 MAT_CLOCK_TIME

```
__thread clock_t MAT_CLOCK_TIME
```

6.46 mattoepz.c File Reference

Functions

- [MATRIX mat_symtoeplz](#) ([MATRIX R](#), [MATRIX result](#))
Computes the symmetric Toeplitz matrix from a co-efficient matrix.

6.46.1 Function Documentation

6.46.1.1 mat_symtoeplz()

```
MATRIX mat_symtoeplz (
    MATRIX R,
    MATRIX result )
```

Computes the symmetric Toeplitz matrix from a co-efficient matrix.

Parameters

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

Returns

symtoep(**R**)

Here is the call graph for this function:

6.47 mattran.c File Reference

Functions

- [MATRIX mat_tran](#) ([MATRIX A](#), [MATRIX result](#))
Computes the transpose of a matrix.

6.47.1 Function Documentation

6.47.1.1 `mat_tran()`

```
MATRIX mat_tran (
    MATRIX A,
    MATRIX result )
```

Computes the transpose of a matrix.

Parameters

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

Returns

\mathbf{A}^T

Here is the call graph for this function: Here is the caller graph for this function:

6.48 README.md File Reference

Index

- __declspec
 - matrix.h, [134](#)
 - mattimers.c, [238](#)
- __int_vec_creat
 - matrix.h, [134](#)
- __int_vecstack_creat
 - matrix.h, [134](#)
- __mat_addfunc
 - matfuncs.c, [71](#)
 - matrix.h, [134](#)
- __mat_arccosh
 - matfuncs.c, [71](#)
 - matrix.h, [135](#)
- __mat_arcsinh
 - matfuncs.c, [72](#)
 - matrix.h, [135](#)
- __mat_arctanh
 - matfuncs.c, [72](#)
 - matrix.h, [135](#)
- __mat_bisquare_wt
 - matfuncs.c, [72](#)
 - matrix.h, [136](#)
- __mat_cart2pol
 - matmisc.c, [90](#)
 - matrix.h, [136](#)
- __mat_creat
 - matrix.h, [136](#)
- __mat_divfunc
 - matfuncs.c, [73](#)
 - matrix.h, [136](#)
- __mat_fft
 - matrix.h, [137](#)
- __mat_huber_wt
 - matfuncs.c, [73](#)
 - matrix.h, [137](#)
- __mat_kd_find_median
 - matrix.h, [137](#)
- __mat_kdtree_k_nearest
 - matrix.h, [138](#)
- __mat_kdtree_make_tree
 - matrix.h, [138](#)
- __mat_kdtree_nearest
 - matrix.h, [138](#)
- __mat_lint
 - matrix.h, [138](#)
- __mat_logplusone
 - matfuncs.c, [73](#)
 - matrix.h, [139](#)
- __mat_mds_metric
 - matmds.c, [86](#)
 - matrix.h, [139](#)
- __mat_mds_nonmetric
 - matmds.c, [87](#)
 - matrix.h, [139](#)
- __mat_mulfunc
 - matfuncs.c, [74](#)
 - matrix.h, [139](#)
- __mat_pol2cart
 - matmisc.c, [90](#)
 - matrix.h, [140](#)
- __mat_powerof2
 - matrix.h, [140](#)
- __mat_quicksort
 - matrix.h, [140](#)
- __mat_rand
 - matrand.c, [114](#)
 - matrix.h, [140](#)
- __mat_randexp
 - matrand.c, [114](#)
 - matrix.h, [141](#)
- __mat_randfun
 - matrand.c, [114](#)
 - matrix.h, [141](#)
- __mat_randn
 - matrand.c, [114](#)
 - matrix.h, [141](#)
- __mat_round_away_zero
 - matfuncs.c, [74](#)
 - matrix.h, [141](#)
- __mat_round_towards_zero
 - matfuncs.c, [75](#)
 - matrix.h, [141](#)
- __mat_sqrfunc
 - matfuncs.c, [75](#)
 - matrix.h, [142](#)
- __mat_sqrtfunc
 - matfuncs.c, [75](#)
 - matrix.h, [142](#)
- __mat_subfunc
 - matfuncs.c, [76](#)
 - matrix.h, [143](#)
- __matstack_creat
 - matrix.h, [143](#)
- _is_allocated
 - mat_kdtree, [18](#)
- adj
 - mat_graph, [12](#)

- class_covars
 - mat_bayes_model, 9
- class_labels
 - mat_bayes_model, 9
 - mat_perceptron, 23
- class_means
 - mat_bayes_model, 10
- class_priors
 - mat_bayes_model, 10
- class_weights
 - mat_perceptron, 23
- dad
 - mat_graph, 12
- data
 - mat_intpqnode, 16
 - mat_kdtree, 18
 - mat_mtypepqnode, 22
 - mat_qintnode, 25
 - mat_qmtyphenode, 25
- element
 - mat_int_priorityqueue, 14
 - mat_mtype_priorityqueue, 19
 - mat_tree_node, 26
- gen_eq
 - matrix.h, 143
 - matstdrels.c, 227
- gen_error
 - materr.c, 63
 - matrix.h, 143
- gen_gt
 - matrix.h, 144
 - matstdrels.c, 227
- gen_lt
 - matrix.h, 144
 - matstdrels.c, 227
- graph_error
 - materr.c, 63
 - matrix.h, 144
- head
 - mat_int_queue, 15
 - mat_mtype_queue, 21
- INT_VECSTACK
 - matrix.h, 128
- INT_VECTOR
 - matrix.h, 128
- id
 - mat_graph, 12
- idx
 - mat_kdnode, 17
- int_vec2_mat
 - matconv.c, 32
 - matrix.h, 145
- int_vec_abs
 - matabs.c, 27
 - matrix.h, 145
- int_vec_add
 - matadd.c, 28
 - matrix.h, 145
- int_vec_adds
 - matadd.c, 29
 - matrix.h, 146
- int_vec_append
 - matcreat.c, 34
 - matrix.h, 146
- int_vec_concat
 - matconcat.c, 30
 - matrix.h, 147
- int_vec_copy
 - matcreat.c, 35
 - matrix.h, 147
- int_vec_creat
 - matcreat.c, 35
 - matrix.h, 147
- int_vec_div
 - matdiv.c, 54
 - matrix.h, 148
- int_vec_divs
 - matdiv.c, 54
 - matrix.h, 148
- int_vec_divs_inv
 - matdiv.c, 55
 - matrix.h, 149
- int_vec_dump
 - matdump.c, 58
 - matrix.h, 149
- int_vec_dumpf
 - matdump.c, 58
 - matrix.h, 149
- int_vec_error
 - materr.c, 63
 - matrix.h, 150
- int_vec_fdump
 - matdump.c, 58
 - matrix.h, 150
- int_vec_fdumpf
 - matdump.c, 59
 - matrix.h, 150
- int_vec_fill
 - matcreat.c, 36
 - matrix.h, 151
- int_vec_fill_type
 - matcreat.c, 36
 - matrix.h, 151
- int_vec_find
 - matrix.h, 151
 - matsearch.c, 222
- int_vec_free
 - matcreat.c, 36
 - matrix.h, 152
- int_vec_inv
 - matdiv.c, 55
 - matrix.h, 152

- int_vec_mean
 - matmean.c, 87
 - matrix.h, 152
- int_vec_mul
 - matmul.c, 97
 - matrix.h, 153
- int_vec_muls
 - matmul.c, 97
 - matrix.h, 153
- int_vec_permute_vect
 - matmisc.c, 90
 - matrix.h, 154
- int_vec_randperm
 - matrand.c, 115
 - matrix.h, 154
- int_vec_sub
 - matrix.h, 154
 - matsub.c, 228
- int_vec_subs
 - matrix.h, 155
 - matsub.c, 228
- int_vec_subs_neg
 - matrix.h, 155
 - matsub.c, 229
- int_vec_sum
 - matrix.h, 155
 - matsum.c, 232
- int_vec_unique
 - matmisc.c, 91
 - matrix.h, 156
- int_vecstack_creat
 - matcreat.c, 37
 - matrix.h, 156
- int_vecstack_error
 - materr.c, 63
 - matrix.h, 157
- int_vecstack_free
 - matcreat.c, 37
 - matrix.h, 157
- istrained
 - mat_perceptron, 23
- kdtree
 - mat_kdtree, 18
- left
 - mat_kdnode, 17
 - mat_tree_node, 26
- length
 - mat_int_priorityqueue, 14
 - mat_int_stack, 16
 - mat_kdtree, 19
 - mat_mtype_priorityqueue, 20
 - mat_mtype_stack, 21
- MAT_BAYES_MODEL
 - matrix.h, 128
- MAT_CLOCK_TIME
 - matrix.h, 221
- mattimers.c, 239
- MAT_GNODE
 - matrix.h, 129
- MAT_GRAPH
 - matrix.h, 129
- MAT_INT_PRIORITYQUEUE
 - matrix.h, 129
- MAT_INT_QUEUE
 - matrix.h, 130
- MAT_INT_STACK
 - matrix.h, 130
- MAT_INTPQNODE
 - matrix.h, 130
- MAT_KDNODE
 - matrix.h, 130
- MAT_KDTREE
 - matrix.h, 131
- MAT_MTYPE_PRIORITYQUEUE
 - matrix.h, 131
- MAT_MTYPE_QUEUE
 - matrix.h, 131
- MAT_MTYPE_STACK
 - matrix.h, 132
- MAT_MTYPEPQNODE
 - matrix.h, 132
- MAT_PERCEPTRON
 - matrix.h, 132
- MAT_QINTNODE
 - matrix.h, 132
- MAT_QMTYPENODE
 - matrix.h, 133
- MAT_SEED
 - matrand.c, 116
 - matrix.h, 221
- MAT_SET_SEED
 - matrand.c, 116
 - matrix.h, 222
- MAT_TREE_NODE
 - matrix.h, 133
- MAT_TREE
 - matrix.h, 133
- MATRIX
 - matrix.h, 133
- MATSTACK
 - matrix.h, 133
- MATVEC_DPOINTER
 - matrix.h, 134
- mat_2int_vec
 - matconv.c, 32
 - matrix.h, 157
- mat_abs
 - matabs.c, 27
 - matrix.h, 158
- mat_add
 - matadd.c, 29
 - matrix.h, 158
- mat_adds
 - matadd.c, 30

- matrix.h, 158
- mat_backsubs1
 - matrix.h, 159
 - matsolve.c, 223
- mat_bayes_classifier_test
 - matprec.c, 112
 - matrix.h, 159
- mat_bayes_classifier_train
 - matprec.c, 112
 - matrix.h, 159
- mat_bayes_model, 9
 - class_covars, 9
 - class_labels, 9
 - class_means, 10
 - class_priors, 10
 - matrix.h, 128
 - num_of_classes, 10
 - num_of_features, 10
- mat_bayes_model_creat
 - matcreat.c, 37
 - matrix.h, 159
- mat_bayes_model_free
 - matcreat.c, 38
 - matrix.h, 159
- mat_binom
 - matpoly.c, 104
 - matrix.h, 160
- mat_binom_init
 - matpoly.c, 105
 - matrix.h, 160
- mat_binom_series_table
 - matpoly.c, 111
 - matrix.h, 221
- mat_bisquare_wt
 - matfuncs.c, 76
 - matrix.h, 160
- mat_bs_delete
 - matdatastruct.c, 46
 - matrix.h, 161
- mat_bs_find
 - matdatastruct.c, 46
 - matrix.h, 161
- mat_bs_find_max
 - matdatastruct.c, 46
 - matrix.h, 161
- mat_bs_find_min
 - matdatastruct.c, 46
 - matrix.h, 161
- mat_bs_free
 - matdatastruct.c, 46
 - matrix.h, 161
- mat_bs_inorder
 - matdatastruct.c, 46
 - matrix.h, 161
- mat_bs_insert
 - matdatastruct.c, 46
 - matrix.h, 162
- mat_bs_make_null
 - matdatastruct.c, 47
 - matrix.h, 162
- mat_bsxfun
 - matmisc.c, 91
 - matrix.h, 162
- mat_calc_dist_sq
 - matmisc.c, 91
 - matrix.h, 162
- mat_cart2pol
 - matmisc.c, 92
 - matrix.h, 163
- mat_cheby
 - matpoly.c, 105
 - matrix.h, 163
- mat_cheby_approx
 - matpoly.c, 105
 - matrix.h, 164
- mat_cheby_coeffs_to_poly
 - matpoly.c, 106
 - matrix.h, 164
- mat_cheby_init
 - matpoly.c, 106
 - matrix.h, 165
- mat_cheby_series_table
 - matpoly.c, 111
 - matrix.h, 221
- mat_cholesky
 - matrix.h, 165
 - matsolve.c, 223
- mat_cofact
 - matdet.c, 52
 - matrix.h, 165
- mat_colcopy
 - matcreat.c, 38
 - matrix.h, 166
- mat_concat
 - matconcat.c, 31
 - matrix.h, 166
- mat_conjgrad
 - matrix.h, 166
 - matsolve.c, 223
- mat_conv2
 - matfilter.c, 66
 - matrix.h, 167
- mat_copy
 - matcreat.c, 38
 - matrix.h, 167
- mat_corcol
 - matpca.c, 101
 - matrix.h, 168
- mat_count_words_in_line
 - matrix.h, 168
 - mattext.c, 234
- mat_covcol
 - matpca.c, 101
 - matrix.h, 168
- mat_creat
 - matcreat.c, 39

- matrix.h, 168
- mat_creat_diag
 - matcreat.c, 39
 - matrix.h, 169
- mat_det
 - matdet.c, 53
 - matrix.h, 169
- mat_diagmul
 - matmul.c, 98
 - matrix.h, 170
- mat_div_dot
 - matdiv.c, 56
 - matrix.h, 170
- mat_divs
 - matdiv.c, 56
 - matrix.h, 170
- mat_divs_inv
 - matdiv.c, 56
 - matrix.h, 171
- mat_dlmread
 - matrix.h, 171
 - mattext.c, 235
- mat_dlmwrite
 - matrix.h, 172
 - mattext.c, 235
- mat_dump
 - matdump.c, 59
 - matrix.h, 172
- mat_dumpf
 - matdump.c, 59
 - matrix.h, 172
- mat_durbin
 - matdurbn.c, 61
 - matrix.h, 173
- mat_eig_sym
 - matpca.c, 101
 - matrix.h, 173
- mat_error
 - materr.c, 64
 - matrix.h, 173
- mat_fdump
 - matdump.c, 60
 - matrix.h, 174
- mat_fdumpf
 - matdump.c, 60
 - matrix.h, 174
- mat_fft2
 - matfft.c, 65
 - matrix.h, 174
- mat_fgetmat
 - matcreat.c, 40
 - matrix.h, 175
- mat_fill
 - matcreat.c, 40
 - matrix.h, 175
- mat_fill_type
 - matcreat.c, 40
 - matrix.h, 175
- mat_find
 - matrix.h, 176
 - matsearch.c, 222
- mat_find_within_dist
 - matmisc.c, 92
 - matrix.h, 176
- mat_fliplr
 - matflip.c, 70
 - matrix.h, 176
- mat_flipud
 - matflip.c, 70
 - matrix.h, 176
- mat_fnextline
 - matmisc.c, 93
 - matrix.h, 177
- mat_free
 - matcreat.c, 41
 - matrix.h, 177
- mat_get_sub_matrix_from_cols
 - matmisc.c, 93
 - matrix.h, 177
- mat_get_sub_matrix_from_rows
 - matmisc.c, 93
 - matrix.h, 178
- mat_get_sub_vector
 - matmisc.c, 94
 - matrix.h, 178
- mat_gfunc
 - matfuncs.c, 76
 - matrix.h, 179
- mat_gnode, 10
 - matrix.h, 129
 - next, 11
 - v, 11
 - weight, 11
- mat_go_next_word
 - matrix.h, 179
 - mattext.c, 235
- mat_graph, 11
 - adj, 12
 - dad, 12
 - id, 12
 - matrix.h, 129
 - nedges, 12
 - nvertices, 12
 - pq, 12
 - val, 12
 - vseq, 13
 - weighted, 13
 - z, 13
- mat_graph_adjlist
 - matgraph.c, 78
 - matrix.h, 179
- mat_graph_adjm_to_adjl
 - matgraph.c, 78
 - matrix.h, 179
- mat_graph_creat
 - matgraph.c, 78

- matrix.h, 180
- mat_graph_dump
 - matgraph.c, 78
 - matrix.h, 180
- mat_graph_dumpf
 - matgraph.c, 78
 - matrix.h, 180
- mat_graph_reverse
 - matgraph.c, 79
 - matrix.h, 180
- mat_graph_search
 - matgraph.c, 79
 - matrix.h, 180
- mat_graph_visit
 - matgraph.c, 79
 - matrix.h, 180
- mat_huber_wt
 - matfuncs.c, 77
 - matrix.h, 181
- mat_innerprod
 - matinnerprod.c, 80
 - matrix.h, 181
- mat_int_priorityqueue, 13
 - element, 14
 - length, 14
 - matrix.h, 129
 - p, 14
 - type, 14
- mat_int_priorityqueue_creat
 - matdatastruct.c, 47
 - matrix.h, 181
- mat_int_priorityqueue_dequeue
 - matdatastruct.c, 47
 - matrix.h, 181
- mat_int_priorityqueue_enqueue
 - matdatastruct.c, 47
 - matrix.h, 182
- mat_int_priorityqueue_free
 - matdatastruct.c, 47
 - matrix.h, 182
- mat_int_priorityqueue_is_empty
 - matdatastruct.c, 47
 - matrix.h, 182
- mat_int_priorityqueue_update
 - matdatastruct.c, 48
 - matrix.h, 182
- mat_int_qadrat
 - matintegrate.c, 81
 - matrix.h, 182
- mat_int_queue, 14
 - head, 15
 - matrix.h, 129
 - p, 15
 - tail, 15
- mat_int_queue_creat
 - matdatastruct.c, 48
 - matrix.h, 183
- mat_int_queue_dequeue
 - matdatastruct.c, 48
 - matrix.h, 183
- mat_int_queue_enqueue
 - matdatastruct.c, 48
 - matrix.h, 183
- mat_int_queue_free
 - matdatastruct.c, 48
 - matrix.h, 183
- mat_int_queue_is_empty
 - matdatastruct.c, 48
 - matrix.h, 183
- mat_int_simpson
 - matintegrate.c, 82
 - matrix.h, 183
- mat_int_stack, 15
 - length, 16
 - matrix.h, 130
 - p, 16
 - stack, 16
- mat_int_stack_creat
 - matdatastruct.c, 49
 - matrix.h, 184
- mat_int_stack_free
 - matdatastruct.c, 49
 - matrix.h, 184
- mat_int_stack_is_empty
 - matdatastruct.c, 49
 - matrix.h, 184
- mat_int_stack_pop
 - matdatastruct.c, 49
 - matrix.h, 184
- mat_int_stack_push
 - matdatastruct.c, 49
 - matrix.h, 184
- mat_int_trapezoid
 - matintegrate.c, 82
 - matrix.h, 184
- mat_intpqnode, 16
 - data, 16
 - matrix.h, 130
 - priority, 17
- mat_inv
 - matinv.c, 83
 - matrix.h, 185
- mat_inv_dot
 - matdiv.c, 57
 - matrix.h, 185
- mat_isnumeric
 - matrix.h, 186
 - mattext.c, 237
- mat_kdnode, 17
 - idx, 17
 - left, 17
 - matrix.h, 130
 - right, 17
 - x, 18
- mat_kdtree, 18
 - _is_allocated, 18

- data, 18
- kdtree, 18
- length, 19
- matrix.h, 131
- ndims, 19
- mat_kdtree_free
 - matkdtree.c, 84
 - matrix.h, 186
- mat_kdtree_k_nearest
 - matkdtree.c, 84
 - matrix.h, 186
- mat_kdtree_make_tree
 - matkdtree.c, 85
 - matrix.h, 187
- mat_kdtree_nearest
 - matkdtree.c, 85
 - matrix.h, 187
- mat_kmeans
 - matprec.c, 112
 - matrix.h, 187
- mat_least_squares
 - matfit.c, 67
 - matrix.h, 188
- mat_legendre
 - matpoly.c, 106
 - matrix.h, 188
- mat_legendre_init
 - matpoly.c, 106
 - matrix.h, 188
- mat_legendre_series_table
 - matpoly.c, 111
 - matrix.h, 221
- mat_linear_ls_fit
 - matfit.c, 67
 - matrix.h, 188
- mat_ksolve
 - matrix.h, 189
 - matsolve.c, 224
- mat_ksolve_durbin
 - matdurbn.c, 61
 - matrix.h, 189
- mat_lu
 - matrix.h, 190
 - matsolve.c, 224
- mat_max
 - matmaxmin.c, 86
 - matrix.h, 190
- mat_mds
 - matmds.c, 87
 - matrix.h, 190
- mat_mean
 - matmean.c, 88
 - matrix.h, 191
- mat_mean_col
 - matmean.c, 88
 - matrix.h, 191
- mat_mean_row
 - matmean.c, 88
- matrix.h, 191
- mat_median
 - matrix.h, 192
 - matsort.c, 225
- mat_min
 - matmaxmin.c, 86
 - matrix.h, 192
- mat_minor
 - matdet.c, 53
 - matrix.h, 192
- mat_mtype_priorityqueue, 19
 - element, 19
 - length, 20
 - matrix.h, 131
 - p, 20
 - type, 20
- mat_mtype_priorityqueue_creat
 - matdatastruct.c, 49
 - matrix.h, 193
- mat_mtype_priorityqueue_dequeue
 - matdatastruct.c, 49
 - matrix.h, 193
- mat_mtype_priorityqueue_enqueue
 - matdatastruct.c, 50
 - matrix.h, 193
- mat_mtype_priorityqueue_free
 - matdatastruct.c, 50
 - matrix.h, 193
- mat_mtype_priorityqueue_is_empty
 - matdatastruct.c, 50
 - matrix.h, 193
- mat_mtype_priorityqueue_update
 - matdatastruct.c, 50
 - matrix.h, 193
- mat_mtype_queue, 20
 - head, 21
 - matrix.h, 131
 - p, 21
 - tail, 21
- mat_mtype_queue_creat
 - matdatastruct.c, 50
 - matrix.h, 194
- mat_mtype_queue_dequeue
 - matdatastruct.c, 50
 - matrix.h, 194
- mat_mtype_queue_enqueue
 - matdatastruct.c, 51
 - matrix.h, 194
- mat_mtype_queue_free
 - matdatastruct.c, 51
 - matrix.h, 194
- mat_mtype_queue_is_empty
 - matdatastruct.c, 51
 - matrix.h, 194
- mat_mtype_stack, 21
 - length, 21
 - matrix.h, 131
 - p, 22

- stack, 22
- mat_mtype_stack_creat
 - matdatastruct.c, 51
 - matrix.h, 194
- mat_mtype_stack_free
 - matdatastruct.c, 51
 - matrix.h, 195
- mat_mtype_stack_is_empty
 - matdatastruct.c, 51
 - matrix.h, 195
- mat_mtype_stack_pop
 - matdatastruct.c, 51
 - matrix.h, 195
- mat_mtype_stack_push
 - matdatastruct.c, 52
 - matrix.h, 195
- mat_mtypepqnode, 22
 - data, 22
 - matrix.h, 132
 - priority, 22
- mat_mul
 - matmul.c, 98
 - matrix.h, 195
- mat_mul_dot
 - matmul.c, 98
 - matrix.h, 196
- mat_mul_fast
 - matmul.c, 100
 - matrix.h, 196
- mat_muls
 - matmul.c, 100
 - matrix.h, 196
- mat_nextline
 - matmisc.c, 94
 - matrix.h, 197
- mat_norm_inf
 - matinnerprod.c, 80
 - matrix.h, 197
- mat_norm_one
 - matinnerprod.c, 80
 - matrix.h, 197
- mat_norm_p
 - matinnerprod.c, 80
 - matrix.h, 197
- mat_omp
 - matpursuit.c, 113
 - matrix.h, 197
- mat_order_statistic
 - matrix.h, 198
 - matsort.c, 225
- mat_pca
 - matpca.c, 101
 - matrix.h, 198
- mat_perceptron, 23
 - class_labels, 23
 - class_weights, 23
 - istrained, 23
 - matrix.h, 132
 - num_of_classes, 24
 - num_of_features, 24
 - num_of_iterations, 24
- mat_perceptron_creat
 - matcreat.c, 41
 - matrix.h, 198
- mat_perceptron_free
 - matcreat.c, 41
 - matrix.h, 198
- mat_perceptron_test
 - matprec.c, 112
 - matrix.h, 200
- mat_perceptron_train
 - matprec.c, 112
 - matrix.h, 200
- mat_perceptron_train_
 - matprec.c, 113
 - matrix.h, 200
- mat_pick_col
 - matmisc.c, 94
 - matrix.h, 200
- mat_pick_row
 - matmisc.c, 95
 - matrix.h, 201
- mat_pinv
 - matpinv.c, 102
 - matrix.h, 201
- mat_pol2cart
 - matmisc.c, 95
 - matrix.h, 201
- mat_poly_add
 - matpoly.c, 107
 - matrix.h, 202
- mat_poly_diff
 - matpoly.c, 107
 - matrix.h, 202
- mat_poly_diff_eval
 - matpoly.c, 107
 - matrix.h, 203
- mat_poly_div
 - matpoly.c, 108
 - matrix.h, 203
- mat_poly_eval
 - matpoly.c, 108
 - matrix.h, 203
- mat_poly_mul
 - matpoly.c, 110
 - matrix.h, 204
- mat_poly_scale
 - matpoly.c, 110
 - matrix.h, 204
- mat_poly_shift
 - matpoly.c, 111
 - matrix.h, 205
- mat_qintnode, 24
 - data, 25
 - matrix.h, 132
 - next, 25

- mat_qmtyphenode, 25
 - data, 25
 - matrix.h, 133
 - next, 25
- mat_qr
 - matdurbn.c, 62
 - matrix.h, 205
- mat_qsort
 - matrix.h, 206
 - matsort.c, 226
- mat_rand
 - matrand.c, 115
 - matrix.h, 206
- mat_randexp
 - matrand.c, 115
 - matrix.h, 206
- mat_randfun
 - matrand.c, 115
 - matrix.h, 206
- mat_randn
 - matrand.c, 115
 - matrix.h, 207
- mat_randperm
 - matrand.c, 116
 - matrix.h, 207
- mat_randperm_n
 - matrand.c, 116
 - matrix.h, 207
- mat_read_word
 - matrix.h, 207
 - mattext.c, 237
- mat_reg_inv
 - matinv.c, 83
 - matrix.h, 208
- mat_rob_least_squares
 - matfit.c, 68
 - matrix.h, 208
- mat_robust_fit
 - matfit.c, 68
 - matrix.h, 208
- mat_rowcopy
 - matcreat.c, 42
 - matrix.h, 209
- mat_scpcol
 - matpca.c, 102
 - matrix.h, 209
- mat_set_seed
 - matrand.c, 116
 - matrix.h, 210
- mat_sub
 - matrix.h, 210
 - matsub.c, 229
- mat_submat
 - matrix.h, 210
 - matsubx.c, 231
- mat_subs
 - matrix.h, 211
 - matsub.c, 230
- mat_subs_neg
 - matrix.h, 211
 - matsub.c, 230
- mat_sum
 - matrix.h, 211
 - matsum.c, 232
- mat_sum_col
 - matrix.h, 212
 - matsum.c, 232
- mat_sum_row
 - matrix.h, 212
 - matsum.c, 233
- mat_svd
 - matrix.h, 213
 - matsvd.c, 233
- mat_symtoeplz
 - matrix.h, 213
 - mattoepz.c, 239
- mat_tic
 - matrix.h, 213
- mat_toc
 - matrix.h, 213
 - mattimers.c, 238
- mat_toc_print
 - matrix.h, 214
 - mattimers.c, 238
- mat_tqli
 - matpca.c, 102
 - matrix.h, 214
- mat_tran
 - matrix.h, 214
 - mattran.c, 240
- mat_tred2
 - matpca.c, 102
 - matrix.h, 214
- mat_tree_node, 26
 - element, 26
 - left, 26
 - matrix.h, 133
 - right, 26
- mat_vectorize
 - matconv.c, 32
 - matrix.h, 215
- mat_vectorize_tr
 - matconv.c, 33
 - matrix.h, 215
- mat_w_least_squares
 - matfit.c, 69
 - matrix.h, 215
- mat_wpinv
 - matpinv.c, 103
 - matrix.h, 216
- mat_xcopy
 - matcreat.c, 42
 - matrix.h, 216
- mat_xjoin
 - matcreat.c, 43
 - matrix.h, 217

- matabs.c, 27
 - int_vec_abs, 27
 - mat_abs, 27
- matadd.c, 28
 - int_vec_add, 28
 - int_vec_adds, 29
 - mat_add, 29
 - mat_adds, 30
- matcompress.c, 30
- matconcat.c, 30
 - int_vec_concat, 30
 - mat_concat, 31
- matconv.c, 31
 - int_vec2_mat, 32
 - mat_2int_vec, 32
 - mat_vectorize, 32
 - mat_vectorize_tr, 33
- matcreat.c, 33
 - int_vec_append, 34
 - int_vec_copy, 35
 - int_vec_creat, 35
 - int_vec_fill, 36
 - int_vec_fill_type, 36
 - int_vec_free, 36
 - int_vecstack_creat, 37
 - int_vecstack_free, 37
 - mat_bayes_model_creat, 37
 - mat_bayes_model_free, 38
 - mat_colcopy, 38
 - mat_copy, 38
 - mat_creat, 39
 - mat_creat_diag, 39
 - mat_fgetmat, 40
 - mat_fill, 40
 - mat_fill_type, 40
 - mat_free, 41
 - mat_perceptron_creat, 41
 - mat_perceptron_free, 41
 - mat_rowcopy, 42
 - mat_xcopy, 42
 - mat_xjoin, 43
 - matstack_append, 43
 - matstack_creat, 43
 - matstack_free, 44
 - matvec_creat, 44
 - matvec_free, 44
- matdatastruct.c, 45
 - mat_bs_delete, 46
 - mat_bs_find, 46
 - mat_bs_find_max, 46
 - mat_bs_find_min, 46
 - mat_bs_free, 46
 - mat_bs_inorder, 46
 - mat_bs_insert, 46
 - mat_bs_make_null, 47
 - mat_int_priorityqueue_creat, 47
 - mat_int_priorityqueue_dequeue, 47
 - mat_int_priorityqueue_enqueue, 47
 - mat_int_priorityqueue_free, 47
 - mat_int_priorityqueue_is_empty, 47
 - mat_int_priorityqueue_update, 48
 - mat_int_queue_creat, 48
 - mat_int_queue_dequeue, 48
 - mat_int_queue_enqueue, 48
 - mat_int_queue_free, 48
 - mat_int_queue_is_empty, 48
 - mat_int_stack_creat, 49
 - mat_int_stack_free, 49
 - mat_int_stack_is_empty, 49
 - mat_int_stack_pop, 49
 - mat_int_stack_push, 49
 - mat_mtype_priorityqueue_creat, 49
 - mat_mtype_priorityqueue_dequeue, 49
 - mat_mtype_priorityqueue_enqueue, 50
 - mat_mtype_priorityqueue_free, 50
 - mat_mtype_priorityqueue_is_empty, 50
 - mat_mtype_priorityqueue_update, 50
 - mat_mtype_queue_creat, 50
 - mat_mtype_queue_dequeue, 50
 - mat_mtype_queue_enqueue, 51
 - mat_mtype_queue_free, 51
 - mat_mtype_queue_is_empty, 51
 - mat_mtype_stack_creat, 51
 - mat_mtype_stack_free, 51
 - mat_mtype_stack_is_empty, 51
 - mat_mtype_stack_pop, 51
 - mat_mtype_stack_push, 52
- matdet.c, 52
 - mat_cofact, 52
 - mat_det, 53
 - mat_minor, 53
- matdiv.c, 53
 - int_vec_div, 54
 - int_vec_divs, 54
 - int_vec_divs_inv, 55
 - int_vec_inv, 55
 - mat_div_dot, 56
 - mat_divs, 56
 - mat_divs_inv, 56
 - mat_inv_dot, 57
- matdump.c, 57
 - int_vec_dump, 58
 - int_vec_dumpf, 58
 - int_vec_fdump, 58
 - int_vec_fdumpf, 59
 - mat_dump, 59
 - mat_dumpf, 59
 - mat_fdump, 60
 - mat_fdumpf, 60
- matdurbin.c, 60
 - mat_durbin, 61
 - mat_solve_durbin, 61
 - mat_qr, 62
- materr.c, 62
 - gen_error, 63
 - graph_error, 63

- int_vec_error, 63
- int_vecstack_error, 63
- mat_error, 64
- matstack_error, 64
- pq_error, 64
- queue_error, 65
- stack_error, 65
- matfft.c, 65
 - mat_fft2, 65
- matfilter.c, 66
 - mat_conv2, 66
- matfit.c, 67
 - mat_least_squares, 67
 - mat_linear_ls_fit, 67
 - mat_rob_least_squares, 68
 - mat_robust_fit, 68
 - mat_w_least_squares, 69
- matflip.c, 69
 - mat_fliplr, 70
 - mat_flipud, 70
- matfuncs.c, 70
 - __mat_addfunc, 71
 - __mat_arccosh, 71
 - __mat_arcsinh, 72
 - __mat_arctanh, 72
 - __mat_bisquare_wt, 72
 - __mat_divfunc, 73
 - __mat_huber_wt, 73
 - __mat_logplusone, 73
 - __mat_mulfunc, 74
 - __mat_round_away_zero, 74
 - __mat_round_towards_zero, 75
 - __mat_sqrfunc, 75
 - __mat_sqrtfunc, 75
 - __mat_subfunc, 76
 - mat_bisquare_wt, 76
 - mat_gfunc, 76
 - mat_huber_wt, 77
- matgraph.c, 77
 - mat_graph_adjlist, 78
 - mat_graph_adjm_to_adjl, 78
 - mat_graph_creat, 78
 - mat_graph_dump, 78
 - mat_graph_dumpf, 78
 - mat_graph_reverse, 79
 - mat_graph_search, 79
 - mat_graph_visit, 79
- matinnerprod.c, 79
 - mat_innerprod, 80
 - mat_norm_inf, 80
 - mat_norm_one, 80
 - mat_norm_p, 80
- matintegrate.c, 80
 - mat_int_qadrat, 81
 - mat_int_simpson, 82
 - mat_int_trapezoid, 82
- matinv.c, 83
 - mat_inv, 83
 - mat_reg_inv, 83
- matkdtree.c, 84
 - mat_kdtree_free, 84
 - mat_kdtree_k_nearest, 84
 - mat_kdtree_make_tree, 85
 - mat_kdtree_nearest, 85
- matmaxmin.c, 86
 - mat_max, 86
 - mat_min, 86
- matmds.c, 86
 - __mat_mds_metric, 86
 - __mat_mds_nonmetric, 87
 - mat_mds, 87
- matmean.c, 87
 - int_vec_mean, 87
 - mat_mean, 88
 - mat_mean_col, 88
 - mat_mean_row, 88
- matmisc.c, 89
 - __mat_cart2pol, 90
 - __mat_pol2cart, 90
 - int_vec_permute_vect, 90
 - int_vec_unique, 91
 - mat_bsxfun, 91
 - mat_calc_dist_sq, 91
 - mat_cart2pol, 92
 - mat_find_within_dist, 92
 - mat_fnextline, 93
 - mat_get_sub_matrix_from_cols, 93
 - mat_get_sub_matrix_from_rows, 93
 - mat_get_sub_vector, 94
 - mat_nextline, 94
 - mat_pick_col, 94
 - mat_pick_row, 95
 - mat_pol2cart, 95
 - mats_isinf, 96
 - mats_isnan, 96
- matmul.c, 96
 - int_vec_mul, 97
 - int_vec_muls, 97
 - mat_diagmul, 98
 - mat_mul, 98
 - mat_mul_dot, 98
 - mat_mul_fast, 100
 - mat_muls, 100
- matpca.c, 101
 - mat_corcol, 101
 - mat_covcol, 101
 - mat_eig_sym, 101
 - mat_pca, 101
 - mat_scpcol, 102
 - mat_tqli, 102
 - mat_tred2, 102
- matpinv.c, 102
 - mat_pinv, 102
 - mat_wpinv, 103
- matpoly.c, 103
 - mat_binom, 104

- mat_binom_init, 105
- mat_binom_series_table, 111
- mat_cheby, 105
- mat_cheby_approx, 105
- mat_cheby_coeffs_to_poly, 106
- mat_cheby_init, 106
- mat_cheby_series_table, 111
- mat_legendre, 106
- mat_legendre_init, 106
- mat_legendre_series_table, 111
- mat_poly_add, 107
- mat_poly_diff, 107
- mat_poly_diff_eval, 107
- mat_poly_div, 108
- mat_poly_eval, 108
- mat_poly_mul, 110
- mat_poly_scale, 110
- mat_poly_shift, 111
- matprec.c, 112
 - mat_bayes_classifier_test, 112
 - mat_bayes_classifier_train, 112
 - mat_kmeans, 112
 - mat_perceptron_test, 112
 - mat_perceptron_train, 112
 - mat_perceptron_train_, 113
- matpursuit.c, 113
 - mat_omp, 113
- matrand.c, 114
 - __mat_rand, 114
 - __mat_randexp, 114
 - __mat_randfun, 114
 - __mat_randn, 114
 - int_vec_randperm, 115
 - MAT_SEED, 116
 - MAT_SET_SEED, 116
 - mat_rand, 115
 - mat_randexp, 115
 - mat_randfun, 115
 - mat_randn, 115
 - mat_randperm, 116
 - mat_randperm_n, 116
 - mat_set_seed, 116
- matrix.c, 117
- matrix.h, 117
 - __declspec, 134
 - __int_vec_creat, 134
 - __int_vecstack_creat, 134
 - __mat_addfunc, 134
 - __mat_arccosh, 135
 - __mat_arcsinh, 135
 - __mat_arctanh, 135
 - __mat_bisquare_wt, 136
 - __mat_cart2pol, 136
 - __mat_creat, 136
 - __mat_divfunc, 136
 - __mat_fft, 137
 - __mat_huber_wt, 137
 - __mat_kd_find_median, 137
 - __mat_kdtree_k_nearest, 138
 - __mat_kdtree_make_tree, 138
 - __mat_kdtree_nearest, 138
 - __mat_lint, 138
 - __mat_logplusone, 139
 - __mat_mds_metric, 139
 - __mat_mds_nonmetric, 139
 - __mat_mulfunc, 139
 - __mat_pol2cart, 140
 - __mat_powerof2, 140
 - __mat_quicksort, 140
 - __mat_rand, 140
 - __mat_randexp, 141
 - __mat_randfun, 141
 - __mat_randn, 141
 - __mat_round_away_zero, 141
 - __mat_round_towards_zero, 141
 - __mat_sqrfunc, 142
 - __mat_sqrtfunc, 142
 - __mat_subfunc, 143
 - __matstack_creat, 143
 - gen_eq, 143
 - gen_error, 143
 - gen_gt, 144
 - gen_lt, 144
 - graph_error, 144
 - INT_VECSTACK, 128
 - INT_VECTOR, 128
 - int_vec2_mat, 145
 - int_vec_abs, 145
 - int_vec_add, 145
 - int_vec_adds, 146
 - int_vec_append, 146
 - int_vec_concat, 147
 - int_vec_copy, 147
 - int_vec_creat, 147
 - int_vec_div, 148
 - int_vec_divs, 148
 - int_vec_divs_inv, 149
 - int_vec_dump, 149
 - int_vec_dumpf, 149
 - int_vec_error, 150
 - int_vec_fdump, 150
 - int_vec_fdumpf, 150
 - int_vec_fill, 151
 - int_vec_fill_type, 151
 - int_vec_find, 151
 - int_vec_free, 152
 - int_vec_inv, 152
 - int_vec_mean, 152
 - int_vec_mul, 153
 - int_vec_muls, 153
 - int_vec_permute_vect, 154
 - int_vec_randperm, 154
 - int_vec_sub, 154
 - int_vec_subs, 155
 - int_vec_subs_neg, 155
 - int_vec_sum, 155

int_vec_unique, 156
int_vecstack_creat, 156
int_vecstack_error, 157
int_vecstack_free, 157
MAT_BAYES_MODEL, 128
MAT_CLOCK_TIME, 221
MAT_GNODE, 129
MAT_GRAPH, 129
MAT_INT_PRIORITYQUEUE, 129
MAT_INT_QUEUE, 130
MAT_INT_STACK, 130
MAT_INTPQNODE, 130
MAT_KDNODE, 130
MAT_KDTREE, 131
MAT_MTYPE_PRIORITYQUEUE, 131
MAT_MTYPE_QUEUE, 131
MAT_MTYPE_STACK, 132
MAT_MTYPEPQNODE, 132
MAT_PERCEPTRON, 132
MAT_QINTNODE, 132
MAT_QMTYPENODE, 133
MAT_SEED, 221
MAT_SET_SEED, 222
MAT_TREE_NODE, 133
MAT_TREE, 133
MATRIX, 133
MATSTACK, 133
MATVEC_DPOINTER, 134
mat_2int_vec, 157
mat_abs, 158
mat_add, 158
mat_adds, 158
mat_backsubs1, 159
mat_bayes_classifier_test, 159
mat_bayes_classifier_train, 159
mat_bayes_model, 128
mat_bayes_model_creat, 159
mat_bayes_model_free, 159
mat_binom, 160
mat_binom_init, 160
mat_binom_series_table, 221
mat_bisquare_wt, 160
mat_bs_delete, 161
mat_bs_find, 161
mat_bs_find_max, 161
mat_bs_find_min, 161
mat_bs_free, 161
mat_bs_inorder, 161
mat_bs_insert, 162
mat_bs_make_null, 162
mat_bsxfun, 162
mat_calc_dist_sq, 162
mat_cart2pol, 163
mat_cheby, 163
mat_cheby_approx, 164
mat_cheby_coeffs_to_poly, 164
mat_cheby_init, 165
mat_cheby_series_table, 221
mat_cholesky, 165
mat_cofact, 165
mat_colcopy, 166
mat_concat, 166
mat_conjgrad, 166
mat_conv2, 167
mat_copy, 167
mat_corcol, 168
mat_count_words_in_line, 168
mat_covcol, 168
mat_creat, 168
mat_creat_diag, 169
mat_det, 169
mat_diagmul, 170
mat_div_dot, 170
mat_divs, 170
mat_divs_inv, 171
mat_dlmread, 171
mat_dlmwrite, 172
mat_dump, 172
mat_dumpf, 172
mat_durbin, 173
mat_eig_sym, 173
mat_error, 173
mat_fdump, 174
mat_fdumpf, 174
mat_fft2, 174
mat_fgetmat, 175
mat_fill, 175
mat_fill_type, 175
mat_find, 176
mat_find_within_dist, 176
mat_fliplr, 176
mat_flipud, 176
mat_fnextline, 177
mat_free, 177
mat_get_sub_matrix_from_cols, 177
mat_get_sub_matrix_from_rows, 178
mat_get_sub_vector, 178
mat_gfunc, 179
mat_gnode, 129
mat_go_next_word, 179
mat_graph, 129
mat_graph_adjlist, 179
mat_graph_adjm_to_adjl, 179
mat_graph_creat, 180
mat_graph_dump, 180
mat_graph_dumpf, 180
mat_graph_reverse, 180
mat_graph_search, 180
mat_graph_visit, 180
mat_huber_wt, 181
mat_innerprod, 181
mat_int_priorityqueue, 129
mat_int_priorityqueue_creat, 181
mat_int_priorityqueue_dequeue, 181
mat_int_priorityqueue_enqueue, 182
mat_int_priorityqueue_free, 182

`mat_int_priorityqueue_is_empty`, 182
`mat_int_priorityqueue_update`, 182
`mat_int_qadrat`, 182
`mat_int_queue`, 129
`mat_int_queue_creat`, 183
`mat_int_queue_dequeue`, 183
`mat_int_queue_enqueue`, 183
`mat_int_queue_free`, 183
`mat_int_queue_is_empty`, 183
`mat_int_simpson`, 183
`mat_int_stack`, 130
`mat_int_stack_creat`, 184
`mat_int_stack_free`, 184
`mat_int_stack_is_empty`, 184
`mat_int_stack_pop`, 184
`mat_int_stack_push`, 184
`mat_int_trapezoid`, 184
`mat_intpqnode`, 130
`mat_inv`, 185
`mat_inv_dot`, 185
`mat_isnumeric`, 186
`mat_kdnode`, 130
`mat_kdtree`, 131
`mat_kdtree_free`, 186
`mat_kdtree_k_nearest`, 186
`mat_kdtree_make_tree`, 187
`mat_kdtree_nearest`, 187
`mat_kmeans`, 187
`mat_least_squares`, 188
`mat_legendre`, 188
`mat_legendre_init`, 188
`mat_legendre_series_table`, 221
`mat_linear_ls_fit`, 188
`mat_lsolve`, 189
`mat_lsolve_durbin`, 189
`mat_lu`, 190
`mat_max`, 190
`mat_mds`, 190
`mat_mean`, 191
`mat_mean_col`, 191
`mat_mean_row`, 191
`mat_median`, 192
`mat_min`, 192
`mat_minor`, 192
`mat_mtype_priorityqueue`, 131
`mat_mtype_priorityqueue_creat`, 193
`mat_mtype_priorityqueue_dequeue`, 193
`mat_mtype_priorityqueue_enqueue`, 193
`mat_mtype_priorityqueue_free`, 193
`mat_mtype_priorityqueue_is_empty`, 193
`mat_mtype_priorityqueue_update`, 193
`mat_mtype_queue`, 131
`mat_mtype_queue_creat`, 194
`mat_mtype_queue_dequeue`, 194
`mat_mtype_queue_enqueue`, 194
`mat_mtype_queue_free`, 194
`mat_mtype_queue_is_empty`, 194
`mat_mtype_stack`, 131
`mat_mtype_stack_creat`, 194
`mat_mtype_stack_free`, 195
`mat_mtype_stack_is_empty`, 195
`mat_mtype_stack_pop`, 195
`mat_mtype_stack_push`, 195
`mat_mtypepqnode`, 132
`mat_mul`, 195
`mat_mul_dot`, 196
`mat_mul_fast`, 196
`mat_muls`, 196
`mat_nextline`, 197
`mat_norm_inf`, 197
`mat_norm_one`, 197
`mat_norm_p`, 197
`mat_omp`, 197
`mat_order_statistic`, 198
`mat_pca`, 198
`mat_perceptron`, 132
`mat_perceptron_creat`, 198
`mat_perceptron_free`, 198
`mat_perceptron_test`, 200
`mat_perceptron_train`, 200
`mat_perceptron_train_`, 200
`mat_pick_col`, 200
`mat_pick_row`, 201
`mat_pinv`, 201
`mat_pol2cart`, 201
`mat_poly_add`, 202
`mat_poly_diff`, 202
`mat_poly_diff_eval`, 203
`mat_poly_div`, 203
`mat_poly_eval`, 203
`mat_poly_mul`, 204
`mat_poly_scale`, 204
`mat_poly_shift`, 205
`mat_qintnode`, 132
`mat_qmtypernode`, 133
`mat_qr`, 205
`mat_qsort`, 206
`mat_rand`, 206
`mat_randexp`, 206
`mat_randfun`, 206
`mat_randh`, 207
`mat_randperm`, 207
`mat_randperm_n`, 207
`mat_read_word`, 207
`mat_reg_inv`, 208
`mat_rob_least_squares`, 208
`mat_robust_fit`, 208
`mat_rowcopy`, 209
`mat_scpcol`, 209
`mat_set_seed`, 210
`mat_sub`, 210
`mat_submat`, 210
`mat_subs`, 211
`mat_subs_neg`, 211
`mat_sum`, 211
`mat_sum_col`, 212

- mat_sum_row, 212
- mat_svd, 213
- mat_symtoeplz, 213
- mat_tic, 213
- mat_toc, 213
- mat_toc_print, 214
- mat_tqli, 214
- mat_tran, 214
- mat_tred2, 214
- mat_tree_node, 133
- mat_vectorize, 215
- mat_vectorize_tr, 215
- mat_w_least_squares, 215
- mat_wpinv, 216
- mat_xcopy, 216
- mat_xjoin, 217
- mats_isinf, 217
- mats_isnan, 218
- matstack_append, 218
- matstack_creat, 218
- matstack_error, 219
- matstack_free, 219
- matvec_creat, 219
- matvec_free, 220
- pq_error, 220
- queue_error, 220
- stack_error, 221
- mats_isinf
 - matmisc.c, 96
 - matrix.h, 217
- mats_isnan
 - matmisc.c, 96
 - matrix.h, 218
- matsearch.c, 222
 - int_vec_find, 222
 - mat_find, 222
- matsolve.c, 223
 - mat_backsubs1, 223
 - mat_cholesky, 223
 - mat_conjgrad, 223
 - mat_lsolve, 224
 - mat_lu, 224
- matsort.c, 225
 - mat_median, 225
 - mat_order_statistic, 225
 - mat_qsort, 226
- matstack_append
 - matcreat.c, 43
 - matrix.h, 218
- matstack_creat
 - matcreat.c, 43
 - matrix.h, 218
- matstack_error
 - materr.c, 64
 - matrix.h, 219
- matstack_free
 - matcreat.c, 44
 - matrix.h, 219
- matstdrels.c, 226
 - gen_eq, 227
 - gen_gt, 227
 - gen_lt, 227
- matsub.c, 228
 - int_vec_sub, 228
 - int_vec_subs, 228
 - int_vec_subs_neg, 229
 - mat_sub, 229
 - mat_subs, 230
 - mat_subs_neg, 230
- matsubx.c, 231
 - mat_submat, 231
- matsum.c, 231
 - int_vec_sum, 232
 - mat_sum, 232
 - mat_sum_col, 232
 - mat_sum_row, 233
- matsvd.c, 233
 - mat_svd, 233
- mattext.c, 234
 - mat_count_words_in_line, 234
 - mat_dlmread, 235
 - mat_dlmwrite, 235
 - mat_go_next_word, 235
 - mat_isnumeric, 237
 - mat_read_word, 237
- mattimers.c, 238
 - __declspec, 238
 - MAT_CLOCK_TIME, 239
 - mat_toc, 238
 - mat_toc_print, 238
- mattoepz.c, 239
 - mat_symtoeplz, 239
- mattran.c, 239
 - mat_tran, 240
- matvec_creat
 - matcreat.c, 44
 - matrix.h, 219
- matvec_free
 - matcreat.c, 44
 - matrix.h, 220
- ndims
 - mat_kdtree, 19
- nedges
 - mat_graph, 12
- next
 - mat_gnode, 11
 - mat_qintnode, 25
 - mat_qmtypenode, 25
- num_of_classes
 - mat_bayes_model, 10
 - mat_perceptron, 24
- num_of_features
 - mat_bayes_model, 10
 - mat_perceptron, 24
- num_of_iterations
 - mat_perceptron, 24

- nvertices
 - mat_graph, 12
- p
 - mat_int_priorityqueue, 14
 - mat_int_queue, 15
 - mat_int_stack, 16
 - mat_mtype_priorityqueue, 20
 - mat_mtype_queue, 21
 - mat_mtype_stack, 22
- pq
 - mat_graph, 12
- pq_error
 - materr.c, 64
 - matrix.h, 220
- priority
 - mat_intpqnode, 17
 - mat_mtypepqnode, 22
- queue_error
 - materr.c, 65
 - matrix.h, 220
- README.md, 240
- right
 - mat_kdnode, 17
 - mat_tree_node, 26
- stack
 - mat_int_stack, 16
 - mat_mtype_stack, 22
- stack_error
 - materr.c, 65
 - matrix.h, 221
- tail
 - mat_int_queue, 15
 - mat_mtype_queue, 21
- type
 - mat_int_priorityqueue, 14
 - mat_mtype_priorityqueue, 20
- v
 - mat_gnode, 11
- val
 - mat_graph, 12
- vseq
 - mat_graph, 13
- weight
 - mat_gnode, 11
- weighted
 - mat_graph, 13
- x
 - mat_kdnode, 18
- z
 - mat_graph, 13