linalg

1.3.0

Generated by Doxygen 1.8.11

## **Contents**

1	Main	Page			1
	1.1	Introdu	uction		1
2	Mod	ules Ind	dex		3
	2.1	Module	es List		3
3	Data	Type I	ndex		5
	3.1	Data T	ypes List		5
4	Mod	ule Doc	cumentatio	on	7
	4.1	lapack	Module Re	eference	7
		4.1.1	Detailed	Description	7
	4.2	linalg_	c_binding	Module Reference	7
		4.2.1	Detailed	Description	9
		4.2.2	Function	Subroutine Documentation	9
			4.2.2.1	cholesky_factor_c(n, a, upper, err)	9
			4.2.2.2	cholesky_rank1_downdate_c(n, r, u, err)	10
			4.2.2.3	cholesky_rank1_update_c(n, r, u, err)	10
			4.2.2.4	cmtx_mult_c(transa, transb, m, n, k, alpha, a, lda, b, ldb, beta, c)	11
			4.2.2.5	det_c(n, a, err)	11
			4.2.2.6	diag_cmtx_mult_left_c(m, n, k, alpha, a, b, beta, c)	12
			4.2.2.7	diag_cmtx_mult_right_c(m, n, k, alpha, a, b, beta, c)	12
			4.2.2.8	diag_mtx_mult_cmplx_left_c(m, n, k, alpha, a, b, beta, c)	13
			4.2.2.9	diag_mtx_mult_cmplx_right_c(m, n, k, alpha, a, b, beta, c)	13
			42210	diag mtx mult left c(m n k alpha a h beta c)	13

iv CONTENTS

4.2.2.11	disg_mtx_mult_right_c(m, n, k, alpha, a, b, beta, c)	14
4.2.2.12	eigen_asymm_c(n, a, vals, vecs, err)	14
4.2.2.13	eigen_gen_c(n, a, b, alpha, beta, vecs, err)	15
4.2.2.14	eigen_symm_c(n, vecs, a, vals, err)	15
4.2.2.15	$form\_lu\_c(n, lu, ipvt, u, p)  .  .  .  .  .  .  .  .  .  $	16
4.2.2.16	form_qr_c(m, n, r, tau, q, err)	17
4.2.2.17	form_qr_pivot_c(m, n, r, tau, pvt, q, p, err)	17
4.2.2.18	lu_factor_c(m, n, a, ipvt, err)	18
4.2.2.19	mtx_inverse_c(n, a, err)	18
4.2.2.20	mtx_mult_c(transa, transb, m, n, k, alpha, a, lda, b, ldb, beta, c)	19
4.2.2.21	mtx_pinverse_c(m, n, a, ainv, err)	19
4.2.2.22	mtx_rank_c(m, n, a, err)	20
4.2.2.23	mult_qr_c(trans, m, n, q, tau, c, err)	20
4.2.2.24	mult_rz_c(trans, m, n, I, a, tau, c, err)	21
4.2.2.25	qr_factor_c(m, n, a, tau, err)	22
4.2.2.26	qr_factor_pivot_c(m, n, a, tau, jpvt, err)	22
4.2.2.27	qr_rank1_update_c(m, n, q, r, u, v, err)	23
4.2.2.28	rank1_update_c(m, n, alpha, x, y, a)	23
4.2.2.29	rz_factor_c(m, n, a, tau, err)	24
4.2.2.30	solve_cholesky_c(upper, n, nrhs, a, b)	24
4.2.2.31	solve_least_squares_c(m, n, nrhs, a, b, err)	24
4.2.2.32	solve_lu_c(n, nrhs, a, ipvt, b)	25
4.2.2.33	solve_qr_c(m, n, nrhs, a, tau, b, err)	25
4.2.2.34	solve_qr_pivot_c(m, n, nrhs, a, tau, jpvt, b, err)	26
4.2.2.35	solve_tri_mtx_c(upper, trans, nounit, n, nrhs, alpha, a, b)	26
4.2.2.36	sort_cmplx_ind_c(ascend, n, x, ind)	27
4.2.2.37	sort_dbl_ind_c(ascend, n, x, ind)	27
4.2.2.38	sort_eigen_cmplx_c(ascend, n, vals, vecs)	28
4.2.2.39	sort_eigen_dbl_c(ascend, n, vals, vecs)	28
4.2.2.40	svd_c(m, n, a, s, u, vt, err)	28

CONTENTS

		4.2.2.41	swap_c(n, x, y)	29
		4.2.2.42	$trace\_c(m,n,x) \qquad \dots \qquad \dots \qquad \dots \qquad \dots \qquad \dots \qquad \dots$	29
4.3	linalg_	constants	Module Reference	30
	4.3.1	Detailed	Description	30
4.4	linalg_	core Modu	le Reference	30
	4.4.1	Detailed	Description	31
	4.4.2	Function	Subroutine Documentation	31
		4.4.2.1	det(a, iwork, err)	31
		4.4.2.2	diag_mtx_mult_mtx(lside, trans, alpha, a, b, beta, c, err)	32
		4.4.2.3	diag_mtx_mult_mtx2(lside, alpha, a, b, err)	33
		4.4.2.4	diag_mtx_mult_mtx3(lside, trans, alpha, a, b, beta, c, err)	33
		4.4.2.5	diag_mtx_mult_mtx4(lside, trans, alpha, a, b, beta, c, err)	34
		4.4.2.6	mtx_mult_mtx(transa, transb, alpha, a, b, beta, c, err)	34
		4.4.2.7	mtx_mult_vec(trans, alpha, a, b, beta, c, err)	35
		4.4.2.8	mtx_rank(a, tol, work, olwork, err)	36
		4.4.2.9	rank1_update(alpha, x, y, a, err)	36
		4.4.2.10	recip_mult_array(a, x)	37
		4.4.2.11	swap(x, y, err)	37
		4.4.2.12	trace(x)	38
		4.4.2.13	tri_mtx_mult(upper, alpha, a, beta, b, err)	38
4.5	linalg_	eigen Mod	ule Reference	38
	4.5.1	Detailed	Description	39
	4.5.2	Function	Subroutine Documentation	39
		4.5.2.1	eigen_asymm(a, vals, vecs, work, olwork, err)	39
		4.5.2.2	eigen_gen(a, b, alpha, beta, vecs, work, olwork, err)	40
		4.5.2.3	eigen_symm(vecs, a, vals, work, olwork, err)	41
4.6	linalg_	factor Mod	ule Reference	42
	4.6.1	Detailed	Description	43
	4.6.2	Function	Subroutine Documentation	44
		4.6.2.1	cholesky_factor(a, upper, err)	44

vi

	4.6.2.2	cholesky_rank1_downdate(r, u, work, err)	44
	4.6.2.3	cholesky_rank1_update(r, u, work, err)	45
	4.6.2.4	form_lu_all(lu, ipvt, u, p, err)	46
	4.6.2.5	form_lu_only(lu, u, err)	46
	4.6.2.6	form_qr_no_pivot(r, tau, q, work, olwork, err)	47
	4.6.2.7	form_qr_pivot(r, tau, pvt, q, p, work, olwork, err)	48
	4.6.2.8	lu_factor(a, ipvt, err)	48
	4.6.2.9	mult_qr_mtx(lside, trans, a, tau, c, work, olwork, err)	49
	4.6.2.10	mult_qr_vec(trans, a, tau, c, work, olwork, err)	50
	4.6.2.11	mult_rz_mtx(lside, trans, I, a, tau, c, work, olwork, err)	51
	4.6.2.12	mult_rz_vec(trans, I, a, tau, c, work, olwork, err)	52
	4.6.2.13	qr_factor_no_pivot(a, tau, work, olwork, err)	52
	4.6.2.14	qr_factor_pivot(a, tau, jpvt, work, olwork, err)	54
	4.6.2.15	qr_rank1_update(q, r, u, v, work, err)	55
	4.6.2.16	rz_factor(a, tau, work, olwork, err)	56
	4.6.2.17	svd(a, s, u, vt, work, olwork, err)	57
linalg_	solve Mod	ule Reference	59
4.7.1	Detailed	Description	60
4.7.2	Function	Subroutine Documentation	60
	4.7.2.1	mtx_inverse(a, iwork, work, olwork, err)	60
	4.7.2.2	mtx_pinverse(a, ainv, tol, work, olwork, err)	61
	4.7.2.3	solve_cholesky_mtx(upper, a, b, err)	62
	4.7.2.4	solve_cholesky_vec(upper, a, b, err)	63
	4.7.2.5	solve_least_squares_mtx(a, b, work, olwork, err)	63
	4.7.2.6	solve_least_squares_mtx_pvt(a, b, ipvt, arnk, work, olwork, err)	64
	4.7.2.7	solve_least_squares_mtx_svd(a, b, arnk, s, work, olwork, err)	65
	4.7.2.7 4.7.2.8	solve_least_squares_mtx_svd(a, b, arnk, s, work, olwork, err)	66
	4.7.2.8	solve_least_squares_vec(a, b, work, olwork, err)	66
	4.7.1	4.6.2.4 4.6.2.5 4.6.2.6 4.6.2.7 4.6.2.8 4.6.2.9 4.6.2.10 4.6.2.11 4.6.2.12 4.6.2.13 4.6.2.14 4.6.2.15 4.6.2.16 4.6.2.17 linalg_solve Model 4.7.1 Detailed 4.7.2 Function/ 4.7.2.1 4.7.2.2 4.7.2.3 4.7.2.4 4.7.2.5 4.7.2.6	4.6.2.3 cholesky_rank1_update(r, u, work, err) 4.6.2.4 form_lu_all(lu, ipvt, u, p, err) 4.6.2.5 form_lu_only(lu, u, err) 4.6.2.6 form_gr_no_pivot(r, tau, q, work, olwork, err) 4.6.2.7 form_gr_pivot(r, tau, pvt, q, p, work, olwork, err) 4.6.2.8 lu_factor(a, ipvt, err) 4.6.2.9 mult_gr_mtx(lside, trans, a, tau, c, work, olwork, err) 4.6.2.10 mult_gr_vec(trans, a, tau, c, work, olwork, err) 4.6.2.11 mult_rz_mtx(lside, trans, l, a, tau, c, work, olwork, err) 4.6.2.12 mult_rz_vec(trans, l, a, tau, c, work, olwork, err) 4.6.2.13 gr_factor_no_pivot(a, tau, work, olwork, err) 4.6.2.14 qr_factor_pivot(a, tau, ipvt, work, olwork, err) 4.6.2.15 gr_rank1_update(q, r, u, v, work, err) 4.6.2.16 rz_factor(a, tau, work, olwork, err) 4.6.2.17 svd(a, s, u, vt, work, olwork, err) 4.6.2.18 polyte Module Reference 4.7.1 Detailed Description 4.7.2.1 mtx_inverse(a, iwork, work, olwork, err) 4.7.2.2 mtx_pinverse(a, iwork, work, olwork, err) 4.7.2.3 solve_cholesky_mtx(upper, a, b, err) 4.7.2.4 solve_cholesky_mtx(upper, a, b, err) 4.7.2.5 solve_least_squares_mtx_pvt(a, b, ipvt, arnk, work, olwork, err)

CONTENTS vii

		4.7.2.12	solve_lu_vec(a, ipvt, b, err)	69
		4.7.2.13	solve_qr_no_pivot_mtx(a, tau, b, work, olwork, err)	69
		4.7.2.14	solve_qr_no_pivot_vec(a, tau, b, work, olwork, err)	70
		4.7.2.15	solve_qr_pivot_mtx(a, tau, jpvt, b, work, olwork, err)	70
		4.7.2.16	solve_qr_pivot_vec(a, tau, jpvt, b, work, olwork, err)	71
		4.7.2.17	solve_tri_mtx(lside, upper, trans, nounit, alpha, a, b, err)	72
		4.7.2.18	solve_tri_vec(upper, trans, nounit, a, x, err)	73
4.8	linalg_	sorting Mo	dule Reference	74
	4.8.1	Detailed I	Description	74
	4.8.2	Function/	Subroutine Documentation	75
		4.8.2.1	cmplx_partition(ascend, x, marker)	75
		4.8.2.2	cmplx_partition_ind(ascend, x, ind, marker)	75
		4.8.2.3	dbl_partition_ind(ascend, x, ind, marker)	76
		4.8.2.4	qsort_cmplx(ascend, x)	76
		4.8.2.5	qsort_cmplx_ind(ascend, x, ind)	76
		4.8.2.6	qsort_dbl_ind(ascend, x, ind)	77
		4.8.2.7	sort_cmplx_array(x, ascend)	77
		4.8.2.8	sort_cmplx_array_ind(x, ind, ascend, err)	78
		4.8.2.9	sort_dbl_array(x, ascend)	78
		4.8.2.10	sort_dbl_array_ind(x, ind, ascend, err)	79
		4.8.2.11	sort_eigen_cmplx(vals, vecs, ascend, err)	79
		4.8.2.12	sort_eigen_dbl(vals, vecs, ascend, err)	80

viii CONTENTS

5	Data	Type [	ocumentation		81
	5.1	linalg_	ore::diag_mtx_mult Inte	erface Reference	81
		5.1.1	Detailed Description		81
		5.1.2	Member Function/Subi	routine Documentation	81
			5.1.2.1 diag_mtx_m	ult_mtx(Iside, trans, alpha, a, b, beta, c, err)	81
			5.1.2.2 diag_mtx_m	ult_mtx2(lside, alpha, a, b, err)	82
			5.1.2.3 diag_mtx_m	ult_mtx3(lside, trans, alpha, a, b, beta, c, err)	83
			5.1.2.4 diag_mtx_m	ult_mtx4(lside, trans, alpha, a, b, beta, c, err)	83
	5.2	lapack	:DLAMCH Interface Ref	erence	84
		5.2.1	Detailed Description		84
	5.3	linalg_	eigen::eigen Interface R	eference	84
		5.3.1	Detailed Description		85
		5.3.2	Member Function/Subi	routine Documentation	85
			5.3.2.1 eigen_asym	m(a, vals, vecs, work, olwork, err)	85
			5.3.2.2 eigen_gen(a	ı, b, alpha, beta, vecs, work, olwork, err)	86
			5.3.2.3 eigen_symm	n(vecs, a, vals, work, olwork, err)	87
	5.4	linalg_	actor::form_lu Interface	Reference	88
		5.4.1	Detailed Description		88
		5.4.2	Member Function/Subi	routine Documentation	89
			5.4.2.1 form_lu_all(l	u, ipvt, u, p, err)	89
			5.4.2.2 form_lu_only	/(lu, u, err)	89
	5.5	linalg_	actor::form_qr Interface	Reference	90
		5.5.1	Detailed Description		90
		5.5.2	Member Function/Subi	routine Documentation	90
			5.5.2.1 form_qr_no_	_pivot(r, tau, q, work, olwork, err)	90
			5.5.2.2 form_qr_pivo	ot(r, tau, pvt, q, p, work, olwork, err)	91
	5.6	linalg_	ore::mtx_mult Interface	Reference	92
		5.6.1	Detailed Description		92
		5.6.2	Member Function/Subi	routine Documentation	92
			5.6.2.1 mtx_mult_m	tx(transa, transb, alpha, a, b, beta, c, err)	92

CONTENTS

		5.6.2.2	mtx_mult_vec(trans, alpha, a, b, beta, c, err)	93
5.7	linalg_f	actor::mul	t_qr Interface Reference	94
	5.7.1	Detailed	Description	94
	5.7.2	Member	Function/Subroutine Documentation	94
		5.7.2.1	mult_qr_mtx(lside, trans, a, tau, c, work, olwork, err)	94
		5.7.2.2	mult_qr_vec(trans, a, tau, c, work, olwork, err)	95
5.8	linalg_f	actor::mul	t_rz Interface Reference	96
	5.8.1	Detailed	Description	96
	5.8.2	Member	Function/Subroutine Documentation	96
		5.8.2.1	mult_rz_mtx(lside, trans, I, a, tau, c, work, olwork, err)	96
		5.8.2.2	mult_rz_vec(trans, I, a, tau, c, work, olwork, err)	97
5.9	linalg_f	actor::qr_f	actor Interface Reference	98
	5.9.1	Detailed	Description	98
	5.9.2	Member	Function/Subroutine Documentation	98
		5.9.2.1	qr_factor_no_pivot(a, tau, work, olwork, err)	98
		5.9.2.2	qr_factor_pivot(a, tau, jpvt, work, olwork, err)	100
5.10	linalg_s	solve::solv	e_cholesky Interface Reference	101
	5.10.1	Detailed	Description	101
	5.10.2	Member	Function/Subroutine Documentation	102
		5.10.2.1	solve_cholesky_mtx(upper, a, b, err)	102
		5.10.2.2	solve_cholesky_vec(upper, a, b, err)	102
5.11	linalg_s	solve::solv	e_least_squares Interface Reference	103
	5.11.1	Detailed	Description	103
	5.11.2	Member	Function/Subroutine Documentation	103
		5.11.2.1	solve_least_squares_mtx(a, b, work, olwork, err)	103
		5.11.2.2	solve_least_squares_vec(a, b, work, olwork, err)	104
5.12	linalg_s	solve::solv	e_least_squares_full Interface Reference	104
	5.12.1	Detailed	Description	105
	5.12.2	Member	Function/Subroutine Documentation	105
		5.12.2.1	solve_least_squares_mtx_pvt(a, b, ipvt, arnk, work, olwork, err)	105

CONTENTS

		5.12.2.2	solve_least_squares_vec_pvt(a, b, ipvt, arnk, work, olwork, err)	06
5.13 lii	nalg_s	olve::solve	e_least_squares_svd Interface Reference	07
5	.13.1	Detailed I	Description	07
5	.13.2	Member I	Function/Subroutine Documentation	07
		5.13.2.1	solve_least_squares_mtx_svd(a, b, arnk, s, work, olwork, err)	07
		5.13.2.2	solve_least_squares_vec_svd(a, b, arnk, s, work, olwork, err)	80
5.14 lii	nalg_s	olve::solve	e_lu Interface Reference	09
5	.14.1	Detailed I	Description	09
5	.14.2	Member I	Function/Subroutine Documentation	09
		5.14.2.1	solve_lu_mtx(a, ipvt, b, err)	09
		5.14.2.2	solve_lu_vec(a, ipvt, b, err)	10
5.15 lii	nalg_s	olve::solve	e_qr Interface Reference	10
5	.15.1	Detailed I	Description	11
5	.15.2	Member I	Function/Subroutine Documentation	11
		5.15.2.1	solve_qr_no_pivot_mtx(a, tau, b, work, olwork, err)	11
		5.15.2.2	solve_qr_no_pivot_vec(a, tau, b, work, olwork, err)	12
		5.15.2.3	solve_qr_pivot_mtx(a, tau, jpvt, b, work, olwork, err)	12
		5.15.2.4	solve_qr_pivot_vec(a, tau, jpvt, b, work, olwork, err)	13
5.16 lii	nalg_s	olve::solve	e_triangular_system Interface Reference	14
5	.16.1	Detailed I	Description	14
5	.16.2	Member I	Function/Subroutine Documentation	14
		5.16.2.1	solve_tri_mtx(lside, upper, trans, nounit, alpha, a, b, err)	14
		5.16.2.2	solve_tri_vec(upper, trans, nounit, a, x, err)	15
5.17 lii	nalg_s	orting::so	rt Interface Reference	16
5	.17.1	Detailed I	Description	17
5	.17.2	Member I	Function/Subroutine Documentation	17
		5.17.2.1	sort_cmplx_array(x, ascend)	17
		5.17.2.2	sort_cmplx_array_ind(x, ind, ascend, err)	18
		5.17.2.3	sort_dbl_array(x, ascend)	18
		5.17.2.4	sort_dbl_array_ind(x, ind, ascend, err)	19
		5.17.2.5	sort_eigen_cmplx(vals, vecs, ascend, err)	19
		5.17.2.6	sort_eigen_dbl(vals, vecs, ascend, err)	20

Index

121

## **Chapter 1**

# Main Page

## 1.1 Introduction

LINALG is a linear algebra library that provides a user-friendly interface to several BLAS and LAPACK routines.

Author

Jason Christopherson

Version

1.3.0

2 Main Page

## **Chapter 2**

## **Modules Index**

## 2.1 Modules List

Here is a list of all documented modules with brief descriptions:

lapack	
lapack	7
linalg_c_binding	
linalg_c_binding	7
linalg_constants	
linalg_constants	30
linalg_core	
linalg_core	30
linalg_eigen	
linalg_eigen	38
linalg_factor	
linalg_factor	12
linalg_solve	
linalg_solve	59
linalg_sorting	
linalg_sorting	74

4 Modules Index

## **Chapter 3**

# **Data Type Index**

## 3.1 Data Types List

Here are the data types with brief descriptions:

linalg_core::diag_mtx_mult	
Multiplies a diagonal matrix with another matrix or array	81
lapack::DLAMCH	84
linalg_eigen::eigen	
Computes the eigenvalues, and optionally the eigenvectors, of a matrix	84
linalg_factor::form_lu	
Extracts the L and U matrices from the condensed [L\U] storage format used by the lu_factor .	88
linalg_factor::form_qr	00
Forms the full M-by-M orthogonal matrix Q from the elementary reflectors returned by the base	
QR factorization algorithm	90
linalg core::mtx mult	
Performs the matrix operation: $C = alpha * op(A) * op(B) + beta * C \dots \dots \dots \dots$	92
linalg_factor::mult_qr	
Multiplies a general matrix by the orthogonal matrix Q from a QR factorization	94
linalg factor::mult rz	
Multiplies a general matrix by the orthogonal matrix Z from an RZ factorization	96
linalg factor::qr factor	
Computes the QR factorization of an M-by-N matrix	98
linalg solve::solve cholesky	
· · · · · · · · · · · · · · · · · · ·	101
linalg_solve::solve_least_squares	
Solves the overdetermined or underdetermined system (A*X = B) of M equations of N unknowns	103
linalg_solve::solve_least_squares_full	
Solves the overdetermined or underdetermined system (A*X = B) of M equations of N unknowns,	
but uses a full orthogonal factorization of the system	104
linalg_solve::solve_least_squares_svd	
Solves the overdetermined or underdetermined system $(A*X = B)$ of M equations of N unknowns	
using a singular value decomposition of matrix A	107
linalg_solve::solve_lu	
	109
linalg_solve::solve_qr	
, i	110
linalg_solve::solve_triangular_system	
9	114
linalg_sorting::sort	
Corto an array	116

6 Data Type Index

## **Chapter 4**

## **Module Documentation**

## 4.1 lapack Module Reference

#### lapack

### **Data Types**

• interface **DLAMCH** 

## 4.1.1 Detailed Description

### lapack

#### **Purpose**

Provides interfaces to various LAPACK routines.

## 4.2 linalg\_c\_binding Module Reference

### linalg\_c\_binding

#### **Functions/Subroutines**

- subroutine mtx\_mult\_c (transa, transb, m, n, k, alpha, a, lda, b, ldb, beta, c)
  - Performs the matrix operation: C = alpha \* op(A) \* op(B) + beta \* C.
- subroutine cmtx\_mult\_c (transa, transb, m, n, k, alpha, a, lda, b, ldb, beta, c)
  - Performs the matrix operation: C = alpha \* op(A) \* op(B) + beta \* C.
- subroutine diag\_mtx\_mult\_left\_c (m, n, k, alpha, a, b, beta, c)
  - Computes the matirx operation: C = alpha \* A \* B + beta \* C, where A is a diagonal amtrix.
- subroutine disg\_mtx\_mult\_right\_c (m, n, k, alpha, a, b, beta, c)
  - Comptues the matrix operation: C = alpha \* A \* B + beta \* C, where B is a diagonal matrix.
- subroutine diag\_mtx\_mult\_cmplx\_left\_c (m, n, k, alpha, a, b, beta, c)

Computes the matirx operation: C = alpha \* A \* B + beta \* C, where A is a diagonal amtrix.

• subroutine diag\_mtx\_mult\_cmplx\_right\_c (m, n, k, alpha, a, b, beta, c)

Comptues the matrix operation: C = alpha \* A \* B + beta \* C, where B is a diagonal matrix.

• subroutine diag\_cmtx\_mult\_left\_c (m, n, k, alpha, a, b, beta, c)

Computes the matirx operation: C = alpha \* A \* B + beta \* C, where A is a diagonal amtrix.

• subroutine diag\_cmtx\_mult\_right\_c (m, n, k, alpha, a, b, beta, c)

Comptues the matrix operation: C = alpha \* A \* B + beta \* C, where B is a diagonal matrix.

• subroutine rank1 update c (m, n, alpha, x, y, a)

Performs the rank-1 update to matrix A such that: A = alpha \* X \* Y \* \* T + A, where A is an M-by-N matrix, alpha is a scalar, X is an M-element array, and N is an N-element array.

• pure real(dp) function trace c (m, n, x)

Computes the trace of a matrix (the sum of the main diagonal elements).

• integer(i32) function mtx\_rank\_c (m, n, a, err)

Computes the rank of a matrix.

real(dp) function det\_c (n, a, err)

Computes the determinant of a square matrix.

subroutine swap\_c (n, x, y)

Swaps the contents of two arrays.

- subroutine tri\_mtx\_mult\_c (upper, n, alpha, a, beta, b, err)
- subroutine lu factor c (m, n, a, ipvt, err)

Computes the LU factorization of an M-by-N matrix.

subroutine form\_lu\_c (n, lu, ipvt, u, p)

Extracts the L, U, and P matrices from the output of the lu\_factor routine.

• subroutine qr\_factor\_c (m, n, a, tau, err)

Computes the QR factorization of an M-by-N matrix without pivoting.

• subroutine qr\_factor\_pivot\_c (m, n, a, tau, jpvt, err)

Computes the QR factorization of an M-by-N matrix with column pivoting such that A \* P = Q \* R.

• subroutine form\_qr\_c (m, n, r, tau, q, err)

Forms the full M-by-M orthogonal matrix Q from the elementary reflectors returned by the base QR factorization algorithm.

• subroutine form\_qr\_pivot\_c (m, n, r, tau, pvt, q, p, err)

Forms the full M-by-M orthogonal matrix Q from the elementary reflectors returned by the base QR factorization algorithm.

• subroutine mult qr c (trans, m, n, q, tau, c, err)

Multiplies a general matrix by the orthogonal matrix Q from a QR factorization such that: C = op(Q) \* C.

• subroutine qr\_rank1\_update\_c (m, n, q, r, u, v, err)

Computes the rank 1 update to an M-by-N QR factored matrix A (M >= N) where A = Q \* R, and A1 = A + U \* V \*\* T such that A1 = Q1 \* R1.

• subroutine cholesky\_factor\_c (n, a, upper, err)

Computes the Cholesky factorization of a symmetric, positive definite matrix.

• subroutine cholesky\_rank1\_update\_c (n, r, u, err)

Computes the rank 1 update to a Cholesky factored matrix (upper triangular).

• subroutine cholesky\_rank1\_downdate\_c (n, r, u, err)

Computes the rank 1 downdate to a Cholesky factored matrix (upper triangular).

• subroutine rz\_factor\_c (m, n, a, tau, err)

Factors an upper trapezoidal matrix by means of orthogonal transformations such that  $A = R * Z = (R \ 0) * Z$ . Z is an orthogonal matrix of dimension N-by-N, and R is an M-by-M upper triangular matrix.

subroutine mult\_rz\_c (trans, m, n, I, a, tau, c, err)

Multiplies a general matrix by the orthogonal matrix Z from an RZ factorization such that: C = op(Z) \* C.

• subroutine svd c (m, n, a, s, u, vt, err)

Computes the singular value decomposition of a matrix A. The SVD is defined as: A = U \* S \* V \* \* T, where U is an M-by-M orthogonal matrix, S is an M-by-N diagonal matrix, and V is an N-by-N orthogonal matrix.

subroutine solve\_tri\_mtx\_c (upper, trans, nounit, n, nrhs, alpha, a, b)

Solves one of the matrix equations: op(A) \* X = alpha \* B, where A is a triangular matrix.

• subroutine solve\_lu\_c (n, nrhs, a, ipvt, b)

Solves a system of LU-factored equations.

• subroutine solve\_qr\_c (m, n, nrhs, a, tau, b, err)

Solves a system of M QR-factored equations of N unknowns where M >= N.

• subroutine solve gr pivot c (m, n, nrhs, a, tau, jpvt, b, err)

Solves a system of M QR-factored equations of N unknowns where the QR factorization made use of column pivoting.

subroutine solve\_cholesky\_c (upper, n, nrhs, a, b)

Solves a system of Cholesky factored equations.

• subroutine mtx\_inverse\_c (n, a, err)

Computes the inverse of a square matrix.

• subroutine mtx\_pinverse\_c (m, n, a, ainv, err)

Computes the Moore-Penrose pseudo-inverse of a M-by-N matrix using the singular value decomposition of the matrix.

• subroutine solve least squares c (m, n, nrhs, a, b, err)

Solves the overdetermined or underdetermined system (A\*X = B) of M equations of N unknowns using a QR or LQ factorization of the matrix A. Notice, it is assumed that matrix A has full rank.

• subroutine eigen\_symm\_c (n, vecs, a, vals, err)

Computes the eigenvalues, and optionally the eigenvectors of a real, symmetric matrix.

• subroutine eigen asymm c (n, a, vals, vecs, err)

Computes the eigenvalues, and the right eigenvectors of a square matrix.

subroutine eigen\_gen\_c (n, a, b, alpha, beta, vecs, err)

Computes the eigenvalues, and optionally the right eigenvectors of a square matrix assuming the structure of the eigenvalue problem is A\*X = lambda\*B\*X.

• subroutine sort\_dbl\_ind\_c (ascend, n, x, ind)

Sorts an array of double-precision values.

• subroutine sort\_cmplx\_ind\_c (ascend, n, x, ind)

Sorts an array of complex values according to their real components.

• subroutine sort\_eigen\_cmplx\_c (ascend, n, vals, vecs)

A sorting routine specifically tailored for sorting of eigenvalues and their associated eigenvectors using a quick-sort approach.

• subroutine sort\_eigen\_dbl\_c (ascend, n, vals, vecs)

A sorting routine specifically tailored for sorting of eigenvalues and their associated eigenvectors using a quick-sort approach.

### 4.2.1 Detailed Description

#### linalg\_c\_binding

**Purpose** 

Provides a C friendly interface to the LINALG library.

## 4.2.2 Function/Subroutine Documentation

4.2.2.1 subroutine linalg\_c\_binding::cholesky\_factor\_c ( integer(i32), intent(in), value *n*, real(dp), dimension(n,n), intent(inout) *a*, logical(c\_bool), intent(in), value *upper*, type(errorhandler), intent(inout) *err* )

Computes the Cholesky factorization of a symmetric, positive definite matrix.

#### **Parameters**

in	n	The dimension of the matrix.
in,out	а	On input, the N-by-N matrix to factor. On output, the factored matrix is returned in either
		the upper or lower triangular portion of the matrix, dependent upon the value of upper.
in	upper	An optional input that, if specified, provides control over whether the factorization is
		computed as $A = U**T*U$ (set to true), or as $A = L*L**T$ (set to false). The default
		value is true such that $A = U**T*U$ .
in,out	err	The errorhandler object. If no error handling is desired, simply pass NULL, and errors will
		be dealt with by the default internal error handler. Possible errors that may be encountered
		are as follows.
		LA_MATRIX_FORMAT_ERROR: Occurs if a is not positive definite.

Definition at line 814 of file linalg\_c\_binding.f90.

4.2.2.2 subroutine linalg\_c\_binding::cholesky\_rank1\_downdate\_c ( integer(i32), intent(in), value *n*, real(dp), dimension(n,n), intent(inout) *r*, real(dp), dimension(n), intent(inout) *u*, type(errorhandler), intent(inout) *err* )

Computes the rank 1 downdate to a Cholesky factored matrix (upper triangular).

#### **Parameters**

in	n	The dimension of the matrix.
in,out	r	On input, the N-by-N upper triangular matrix R. On output, the updated matrix R1.
in,out	и	On input, the N-element update vector U. On output, the rotation sines used to transform R to
		R1.
in,out	err	The errorhandler object. If no error handling is desired, simply pass NULL, and errors will be dealt with by the default internal error handler. Possible errors that may be encountered are as follows.
		LA_ARRAY_SIZE_ERROR: Occurs if any of the input array sizes are incorrect.
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>
		<ul> <li>LA_MATRIX_FORMAT_ERROR: Occurs if the downdated matrix is not positive definite.</li> </ul>
		• LA_SINGULAR_MATRIX_ERROR: Occurs if r is singular.

Definition at line 888 of file linalg\_c\_binding.f90.

4.2.2.3 subroutine linalg\_c\_binding::cholesky\_rank1\_update\_c ( integer(i32), intent(in), value *n*, real(dp), dimension(n,n), intent(inout) *r*, real(dp), dimension(n), intent(inout) *u*, type(errorhandler), intent(inout) *err* )

Computes the rank 1 update to a Cholesky factored matrix (upper triangular).

in	n	The dimension of the matrix.
in,out	r	On input, the N-by-N upper triangular matrix R. On output, the updated matrix R1.

#### **Parameters**

in,out	и	On input, the N-element update vector U. On output, the rotation sines used to transform R to
		R1.
in,out	err	The errorhandler object. If no error handling is desired, simply pass NULL, and errors will be dealt with by the default internal error handler. Possible errors that may be encountered are as follows.
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>

Definition at line 849 of file linalg\_c\_binding.f90.

4.2.2.4 subroutine linalg\_c\_binding::cmtx\_mult\_c ( logical(c\_bool), intent(in), value transa, logical(c\_bool), intent(in), value transb, integer(i32), intent(in), value m, integer(i32), intent(in), value n, integer(i32), intent(in), value k, real(dp), intent(in), value alpha, complex(dp), dimension(lda,\*), intent(in) a, integer(i32), intent(in), value lda, complex(dp), dimension(ldb,\*), intent(in) b, integer(i32), intent(in), value ldb, real(dp), intent(in), value beta, complex(dp), dimension(m,n), intent(inout) c)

Performs the matrix operation: C = alpha \* op(A) \* op(B) + beta \* C.

#### **Parameters**

transa	Set to true if $op(A) == A**T$ ; else, set to false if $op(A) == A$ .
transb	Set to true if $op(B) == B**T$ ; else, set to false if $op(B) == B$ .
m	The number of rows in matrix C, and the number of rows in matrix op(A).
n	The number of columns in matrix C, and the number of columns in matrix op(B).
k	The number of columns in matrix op(A), and the number of rows in the matrix op(B).
alpha	The scalar multiplier to matrix A.
а	The M-by-K matrix A.
lda	The leading dimension of matrix A. If transa is true, this value must be at least MAX(1, K); else, if transa is false, this value must be at least MAX(1, M).
b	The K-by-N matrix B.
ldb	The leading dimension of matrix B. If transb is true, this value must be at least MAX(1, N); else, if transb is false, this value must be at least MAX(1, K).
beta	The scalar multiplier to matrix C.
С	The M-by-N matrix C.
	transb m n k alpha a Ida b Idb

Definition at line 96 of file linalg c binding.f90.

4.2.2.5 real(dp) function linalg\_c\_binding::det\_c ( integer(i32), intent(in), value *n*, real(dp), dimension(n,n), intent(inout) *a*, type(errorhandler), intent(inout) *err* )

Computes the determinant of a square matrix.

in	n	The dimension of the matrix.
in,out	а	On input, the N-by-N matrix on which to operate. On output the contents are overwritten by
		the LU factorization of the original matrix.

#### **Parameters**

in,ou	t <i>eri</i>	The errorhandler object. If no error handling is desired, simply pass NULL, and errors will be dealt with by the default internal error handler. Possible errors that may be encountered are as follows.
		LA_ARRAY_SIZE_ERROR: Occurs if the input matrix is not square.
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>

Definition at line 374 of file linalg\_c\_binding.f90.

4.2.2.6 subroutine linalg\_c\_binding::diag\_cmtx\_mult\_left\_c ( integer(i32), intent(in), value *m*, integer(i32), intent(in), value *n*, integer(i32), intent(in), value *k*, real(dp), intent(in), value *alpha*, complex(dp), dimension(min(m, k)), intent(in) *a*, complex(dp), dimension(k, n), intent(in) *b*, real(dp), intent(in), value *beta*, complex(dp), dimension(m, n), intent(inout) *c* )

Computes the matirx operation: C = alpha \* A \* B + beta \* C, where A is a diagonal amtrix.

#### **Parameters**

in	m	The number of rows in matrix C.
in	n	The number of columns in matrix C.
in	k	The number of rows in matrix B.
in	alpha	The scalar multiplier to matrix A.
in	а	A MIN(M,K)-element array containing the diagonal elements of matrix A.
in	b	The K-by-N matrix B.
in	beta	The scalar multiplier to matrix C.
in,out	С	The M-by-N matrix C.

Definition at line 240 of file linalg\_c\_binding.f90.

4.2.2.7 subroutine linalg\_c\_binding::diag\_cmtx\_mult\_right\_c ( integer(i32), intent(in), value *m*, integer(i32), intent(in), value *k*, real(dp), intent(in), value *alpha*, complex(dp), dimension(m, k), intent(in) *a*, complex(dp), dimension(m, n), intent(in) *b*, real(dp), intent(in), value *beta*, complex(dp), dimension(m, n), intent(inout) *c* )

Comptues the matrix operation: C = alpha \* A \* B + beta \* C, where B is a diagonal matrix.

in	m	The number of rows in matrix C.
in	n	The number of columns in matrix C.
in	k	The number of columns in matrix A.
in	alpha	The scalar multiplier to matrix A.
in	а	The M-by-K matrix A.
in	b	A MIN(K,N)-element array containing the diagonal elements of matrix B.
in	beta	The scalar multiplier to matrix C.
in,out	С	The M-by-N matrix C.

Definition at line 265 of file linalg\_c\_binding.f90.

4.2.2.8 subroutine linalg\_c\_binding::diag\_mtx\_mult\_cmplx\_left\_c ( integer(i32), intent(in), value *m*, integer(i32), intent(in), value *n*, integer(i32), intent(in), value *k*, real(dp), intent(in), value *alpha*, complex(dp), dimension(min(m, k)), intent(in) *a*, real(dp), dimension(k, n), intent(in) *b*, real(dp), intent(in), value *beta*, complex(dp), dimension(m, n), intent(inout) *c* )

Computes the matirx operation: C = alpha \* A \* B + beta \* C, where A is a diagonal amtrix.

#### **Parameters**

in	m	The number of rows in matrix C.
in	n	The number of columns in matrix C.
in	k	The number of rows in matrix B.
in	alpha	The scalar multiplier to matrix A.
in	а	A MIN(M,K)-element array containing the diagonal elements of matrix A.
in	b	The K-by-N matrix B.
in	beta	The scalar multiplier to matrix C.
in,out	С	The M-by-N matrix C.

Definition at line 188 of file linalg\_c\_binding.f90.

4.2.2.9 subroutine linalg\_c\_binding::diag\_mtx\_mult\_cmplx\_right\_c ( integer(i32), intent(in), value *m*, integer(i32), intent(in), value *k*, real(dp), intent(in), value *alpha*, real(dp), dimension(m, k), intent(in) *a*, complex(dp), dimension(min(k, n)), intent(in) *b*, real(dp), intent(in), value *beta*, complex(dp), dimension(m, n), intent(inout) *c* )

Comptues the matrix operation: C = alpha \* A \* B + beta \* C, where B is a diagonal matrix.

#### **Parameters**

in	m	The number of rows in matrix C.
in	n	The number of columns in matrix C.
in	k	The number of columns in matrix A.
in	alpha	The scalar multiplier to matrix A.
in	а	The M-by-K matrix A.
in	b	A MIN(K,N)-element array containing the diagonal elements of matrix B.
in	beta	The scalar multiplier to matrix C.
in,out	С	The M-by-N matrix C.

Definition at line 214 of file linalg\_c\_binding.f90.

4.2.2.10 subroutine linalg\_c\_binding::diag\_mtx\_mult\_left\_c ( integer(i32), intent(in), value *m*, integer(i32), intent(in), value *k*, real(dp), intent(in), value *alpha*, real(dp), dimension(min(m,k)), intent(in) *a*, real(dp), dimension(k,n), intent(in) *b*, real(dp), intent(in), value *beta*, real(dp), dimension(m,n), intent(inout) *c* )

Computes the matirx operation: C = alpha \* A \* B + beta \* C, where A is a diagonal amtrix.

#### **Parameters**

in	m	The number of rows in matrix C.
in	n	The number of columns in matrix C.
in	k	The number of rows in matrix B.
in	alpha	The scalar multiplier to matrix A.
in	а	A MIN(M,K)-element array containing the diagonal elements of matrix A.
in	b	The K-by-N matrix B.
in	beta	The scalar multiplier to matrix C.
in,out	С	The M-by-N matrix C.

Definition at line 138 of file linalg\_c\_binding.f90.

4.2.2.11 subroutine linalg\_c\_binding::disg\_mtx\_mult\_right\_c ( integer(i32), intent(in), value *m*, integer(i32), intent(in), value *n*, integer(i32), intent(in), value *k*, real(dp), intent(in), value *alpha*, real(dp), dimension(m, k), intent(in) *a*, real(dp), dimension(m, n), intent(inout) *c* )

Comptues the matrix operation: C = alpha \* A \* B + beta \* C, where B is a diagonal matrix.

#### **Parameters**

in	m	The number of rows in matrix C.
in	n	The number of columns in matrix C.
in	k	The number of columns in matrix A.
in	alpha	The scalar multiplier to matrix A.
in	а	The M-by-K matrix A.
in	b	A MIN(K,N)-element array containing the diagonal elements of matrix B.
in	beta	The scalar multiplier to matrix C.
in,out	С	The M-by-N matrix C.

Definition at line 163 of file linalg\_c\_binding.f90.

4.2.2.12 subroutine linalg\_c\_binding::eigen\_asymm\_c ( integer(i32), intent(in), value n, real(dp), dimension(n,n), intent(inout) a, complex(dp), dimension(n), intent(out) vals, complex(dp), dimension(n,n), intent(out) vecs, type(errorhandler), intent(inout) err )

Computes the eigenvalues, and the right eigenvectors of a square matrix.

in	n	The dimension of the matrix.
in,out	а	On input, the N-by-N matrix on which to operate. On output, the contents of this matrix are
		overwritten.
out	vals	An N-element array containing the eigenvalues of the matrix on output. The eigenvalues
		are not sorted.
out	vecs	An N-by-N matrix containing the right eigenvectors (one per column) on output.

#### **Parameters**

in,out	err	The errorhandler object. If no error handling is desired, simply pass NULL, and errors will be dealt with by the default internal error handler. Possible errors that may be encountered are as follows.
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>
		LA_CONVERGENCE_ERROR: Occurs if the algorithm failed to converge.

Definition at line 1372 of file linalg\_c\_binding.f90.

4.2.2.13 subroutine linalg\_c\_binding::eigen\_gen\_c ( integer(i32), intent(in), value *n*, real(dp), dimension(n, n), intent(inout) *a*, real(dp), dimension(n, n), intent(inout) *b*, complex(dp), dimension(n), intent(out) *alpha*, real(dp), dimension(n), intent(out) *beta*, complex(dp), dimension(n,n), intent(out) *vecs*, type(errorhandler), intent(inout) *err* )

Computes the eigenvalues, and optionally the right eigenvectors of a square matrix assuming the structure of the eigenvalue problem is A\*X = lambda\*B\*X.

#### **Parameters**

in	n	The dimension of the matrix.
in,out	а	On input, the N-by-N matrix A. On output, the contents of this matrix are overwritten.
in,out	b	On input, the N-by-N matrix B. On output, the contents of this matrix are overwritten.
out	alpha	An N-element array that, on output, contains the numerator of the eigenvalue ration ALPHA / BETA. Computation of this ratio isn't necessarily as trivial as it seems as it is entirely possible, and likely, that ALPHA / BETA can overflow or underflow. With that said, the values in ALPHA will always be less than and usually comparable with the NORM(A).
out	beta	An N-element array that, on output, contains the denominator used to determine the eigenvalues as ALPHA / BETA. The values in this array will always be less than and usually comparable with the NORM(B).
out	vecs	An N-by-N matrix containing the right eigenvectors (one per column) on output.
in,out	err	The errorhandler object. If no error handling is desired, simply pass NULL, and errors will be dealt with by the default internal error handler. Possible errors that may be encountered are as follows.  • LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.  • LA_CONVERGENCE_ERROR: Occurs if the algorithm failed to converge.

Definition at line 1421 of file linalg\_c\_binding.f90.

4.2.2.14 subroutine linalg\_c\_binding::eigen\_symm\_c ( integer(i32), intent(in), value *n*, logical(c\_bool), intent(in), value *vecs*, real(dp), dimension(n,n), intent(inout) *a*, real(dp), dimension(n), intent(out) *vals*, type(errorhandler), intent(inout) *err* )

Computes the eigenvalues, and optionally the eigenvectors of a real, symmetric matrix.

#### **Parameters**

in	n	The dimension of the matrix.
in	vecs	Set to true to compute the eigenvectors as well as the eigenvalues; else, set to false to just compute the eigenvalues.
in,out	а	On input, the N-by-N symmetric matrix on which to operate. On output, and if vecs is set to true, the matrix will contain the eigenvectors (one per column) corresponding to each eigenvalue in vals. If vecs is set to false, the lower triangular portion of the matrix is overwritten.
out	vals	An N-element array that will contain the eigenvalues sorted into ascending order.
in,out	err	The errorhandler object. If no error handling is desired, simply pass NULL, and errors will be dealt with by the default internal error handler. Possible errors that may be encountered are as follows.
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>
		<ul> <li>LA_CONVERGENCE_ERROR: Occurs if the algorithm failed to converge.</li> </ul>

Definition at line 1333 of file linalg\_c\_binding.f90.

4.2.2.15 subroutine linalg\_c\_binding::form\_lu\_c ( integer(i32), intent(in), value *n*, real(dp), dimension(n,n), intent(inout) *lu*, integer(i32), dimension(n), intent(in) *ipvt*, real(dp), dimension(n,n), intent(out) *u*, real(dp), dimension(n,n), intent(out) *p* 

Extracts the L, U, and P matrices from the output of the lu\_factor routine.

#### **Parameters**

in	n	The dimension of the original matrix.
in,out	lu	On input, the N-by-N matrix as output by lu_factor. On output, the N-by-N lower triangular
		matrix L.
in	ipvt	The N-element pivot array as output by lu_factor.
out	и	An N-by-N matrix where the U matrix will be written.
out	р	An N-by-N matrix where the row permutation matrix will be written.

#### Remarks

This routine allows extraction of the actual "L", "U", and "P" matrices of the decomposition. To use these matrices to solve the system A\*X = B, the following approach is used.

- 1. First, solve the linear system: L\*Y = P\*B for Y.
- 2. Second, solve the linear system: U\*X = Y for X.

Notice, as both L and U are triangular in structure, the above equations can be solved by forward and backward substitution.

#### See Also

- Wikipedia
- Wolfram MathWorld

Definition at line 521 of file linalg\_c\_binding.f90.

4.2.2.16 subroutine linalg\_c\_binding::form\_qr\_c ( integer(i32), intent(in), value *m*, integer(i32), intent(in), value *n*, real(dp), dimension(m,n), intent(inout) *r*, real(dp), dimension(min(m,n)), intent(in) *tau*, real(dp), dimension(m,m), intent(out) *q*, type(errorhandler), intent(inout) *err* )

Forms the full M-by-M orthogonal matrix Q from the elementary reflectors returned by the base QR factorization algorithm.

#### **Parameters**

in	m	The number of rows in the original matrix.
in	n	The number of columns in the original matrix.
in,out	r	On input, an M-by-N matrix where the elements below the diagonal contain the elementary reflectors generated from the QR factorization. On and above the diagonal, the matrix contains the matrix R. On output, the elements below the diagonal are zeroed such that the remaining matrix is simply the M-by-N matrix R.
in	tau	A MIN(M, N)-element array containing the scalar factors of each elementary reflector defined in r.
out	q	An M-by-M matrix where the full orthogonal matrix Q will be written. In the event that $M > N$ , Q may be supplied as M-by-N, and therefore only return the useful submatrix Q1 (Q = [Q1, Q2]) as the factorization can be written as Q * R = [Q1, Q2] * [R1; 0].
in,out	err	The errorhandler object. If no error handling is desired, simply pass NULL, and errors will be dealt with by the default internal error handler. Possible errors that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if the scalar factor array is not sized appropriately.  • LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.

Definition at line 643 of file linalg\_c\_binding.f90.

4.2.2.17 subroutine linalg\_c\_binding::form\_qr\_pivot\_c ( integer(i32), intent(in), value *m*, integer(i32), intent(in), value *n*, real(dp), dimension(m,n), intent(inout) *r*, real(dp), dimension(min(m,n)), intent(in) *tau*, integer(i32), dimension(n), intent(in) *pvt*, real(dp), dimension(m,m), intent(out) *q*, real(dp), dimension(n,n), intent(out) *p*, type(errorhandler), intent(inout) *err* )

Forms the full M-by-M orthogonal matrix Q from the elementary reflectors returned by the base QR factorization algorithm.

in	m	The number of rows in the original matrix.
in	n	The number of columns in the original matrix.
in,out	r	On input, an M-by-N matrix where the elements below the diagonal contain the elementary reflectors generated from the QR factorization. On and above the diagonal, the matrix contains the matrix R. On output, the elements below the diagonal are zeroed such that the remaining matrix is simply the M-by-N matrix R.
in	tau	A MIN(M, N)-element array containing the scalar factors of each elementary reflector defined
		in r.
in	pvt	An N-element column pivot array as returned by the QR factorization.
out	q	An M-by-M matrix where the full orthogonal matrix Q will be written. In the event that $M>N$ , Q may be supplied as M-by-N, and therefore only return the useful submatrix Q1 (Q = [Q1, Q2]) as the factorization can be written as Q * R = [Q1, Q2] * [R1; 0].

## **Parameters**

out	p	An N-by-N matrix where the pivot matrix will be written.
in,out	err	The errorhandler object. If no error handling is desired, simply pass NULL, and errors will be dealt with by the default internal error handler. Possible errors that may be encountered are as follows.
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if the scalar factor array is not sized appropriately.</li> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>

Definition at line 692 of file linalg\_c\_binding.f90.

4.2.2.18 subroutine linalg\_c\_binding::lu\_factor\_c ( integer(i32), intent(in), value *m*, integer(i32), intent(in), value *n*, real(dp), dimension(m,n), intent(inout) *a*, integer(i32), dimension(min(m,n)), intent(out) *ipvt*, type(errorhandler), intent(inout) *err* 

Computes the LU factorization of an M-by-N matrix.

#### **Parameters**

in	m	The number of rows in the matrix.
in	n	The number of columns in the matrix.
in,out	а	On input, the M-by-N matrix on which to operate. On output, the LU factored matrix in the form [L\U] where the unit diagonal elements of L are not stored.
out	ipvt	An MIN(M, N)-element array used to track row-pivot operations. The array stored pivot information such that row I is interchanged with row IPVT(I).
in,out	err	The errorhandler object. If no error handling is desired, simply pass NULL, and errors will be dealt with by the default internal error handler. Possible errors that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if the pivot array is not sized appropriately.  • LA_SINGULAR_MATRIX_ERROR: Occurs as a warning if a is found to be singular.
		English Color and English Color and a warning in a 15 loans to be singular.

Definition at line 475 of file linalg\_c\_binding.f90.

4.2.2.19 subroutine linalg\_c\_binding::mtx\_inverse\_c ( integer(i32), intent(in), value *n*, real(dp), dimension(n,n), intent(inout) *a*, type(errorhandler), intent(inout) *err* )

Computes the inverse of a square matrix.

in	n	The dimension of the matrix.
in,out	а	On input, the N-by-N matrix to invert. On output, the inverted matrix.

#### **Parameters**

in,out	err	The errorhandler object. If no error handling is desired, simply pass NULL, and errors will be dealt with by the default internal error handler. Possible errors that may be encountered are as follows.
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>
		LA_SINGULAR_MATRIX_ERROR: Occurs if the input matrix is singular.

Definition at line 1214 of file linalg\_c\_binding.f90.

4.2.2.20 subroutine linalg\_c\_binding::mtx\_mult\_c ( logical(c\_bool), intent(in), value transa, logical(c\_bool), intent(in), value transb, integer(i32), intent(in), value m, integer(i32), intent(in), value n, integer(i32), intent(in), value k, real(dp), intent(in), value alpha, real(dp), dimension(lda,\*), intent(in) a, integer(i32), intent(in), value lda, real(dp), dimension(ldb,\*), intent(in) b, integer(i32), intent(in), value ldb, real(dp), intent(in), value beta, real(dp), dimension(m,n), intent(inout) c)

Performs the matrix operation: C = alpha \* op(A) \* op(B) + beta \* C.

#### **Parameters**

in	transa	Set to true if $op(A) == A**T$ ; else, set to false if $op(A) == A$ .
in	transb	Set to true if $op(B) == B**T$ ; else, set to false if $op(B) == B$ .
in	т	The number of rows in matrix C, and the number of rows in matrix op(A).
in	n	The number of columns in matrix C, and the number of columns in matrix op(B).
in	k	The number of columns in matrix op(A), and the number of rows in the matrix op(B).
in	alpha	The scalar multiplier to matrix A.
in	а	The M-by-K matrix A.
in	lda	The leading dimension of matrix A. If transa is true, this value must be at least MAX(1, K); else, if transa is false, this value must be at least MAX(1, M).
in	b	The K-by-N matrix B.
in	ldb	The leading dimension of matrix B. If transb is true, this value must be at least MAX(1, N); else, if transb is false, this value must be at least MAX(1, K).
in	beta	The scalar multiplier to matrix C.
in,out	С	The M-by-N matrix C.

Definition at line 47 of file linalg\_c\_binding.f90.

4.2.2.21 subroutine linalg\_c\_binding::mtx\_pinverse\_c ( integer(i32), intent(in), value *m*, integer(i32), intent(in), value *n*, real(dp), dimension(m,n), intent(inout) *ainv*, type(errorhandler), intent(inout) *err* )

Computes the Moore-Penrose pseudo-inverse of a M-by-N matrix using the singular value decomposition of the matrix.

in	m	The number of rows in the matrix to invert.
in	n	The number of columns in the matrix to invert.

#### **Parameters**

in,out	а	On input, the M-by-N matrix to invert. The matrix is overwritten on output.
out	ainv	The N-by-M matrix where the pseudo-inverse of a will be written.
in,out	err	The errorhandler object. If no error handling is desired, simply pass NULL, and errors will be dealt with by the default internal error handler. Possible errors that may be encountered are as follows.  • LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.  • LA_CONVERGENCE_ERROR: Occurs as a warning if the QR iteration process
		could not converge to a zero value.

Definition at line 1251 of file linalg\_c\_binding.f90.

4.2.2.22 integer(i32) function linalg\_c\_binding::mtx\_rank\_c ( integer(i32), intent(in), value *m*, integer(i32), intent(in), value *n*, real(dp), dimension(m,n), intent(inout) *a*, type(errorhandler), intent(inout) *err* )

Computes the rank of a matrix.

#### **Parameters**

in	m	The number of rows in the matrix.
in	n	The number of columns in the matrix.
in,out	а	On input, the M-by-N matrix of interest. On output, the contents of the matrix are overwritten.
in,out	err	The errorhandler object. If no error handling is desired, simply pass NULL, and errors will be dealt with by the default internal error handler. Possible errors that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> <li>LA_CONVERGENCE_ERROR: Occurs as a warning if the QR iteration process could not converge to a zero value.</li> </ul>

Definition at line 341 of file linalg\_c\_binding.f90.

4.2.2.23 subroutine linalg\_c\_binding::mult\_qr\_c ( logical(c\_bool), intent(in), value *trans*, integer(i32), intent(in), value *m*, integer(i32), intent(in), value *n*, real(dp), dimension(m,m), intent(inout) *q*, real(dp), dimension(min(m,n)), intent(in) *tau*, real(dp), dimension(m,n), intent(inout) *c*, type(errorhandler), intent(inout) *err* )

Multiplies a general matrix by the orthogonal matrix Q from a QR factorization such that: C = op(Q) \* C.

in	trans	Set to true to apply Q**T; else, set to false.
in	m	The number of rows in the matrix c.

#### **Parameters**

in	n	The number of columns in the matrix c.
in	q	On input, an M-by-M matrix containing the elementary reflectors output from the QR factorization. Notice, the contents of this matrix are restored on exit. that the remaining matrix is simply the M-by-N matrix R.
in	tau	A MIN(M,N)-element array containing the scalar factors of each elementary reflector defined in a.
in,out	С	On input, the M-by-N matrix C. On output, the product of the orthogonal matrix Q and the original matrix C.
in,out	err	The errorhandler object. If no error handling is desired, simply pass NULL, and errors will be dealt with by the default internal error handler. Possible errors that may be encountered are as follows.
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if the scalar factor array is not sized appropriately.</li> </ul>
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>

Definition at line 737 of file linalg\_c\_binding.f90.

4.2.2.24 subroutine linalg\_c\_binding::mult\_rz\_c ( logical(c\_bool), intent(in), value *trans*, integer(i32), intent(in), value *m*, integer(i32), intent(in), value *n*, integer(i32), intent(in), value *l*, real(dp), dimension(m,m), intent(inout) *a*, real(dp), dimension(m), intent(in) *tau*, real(dp), dimension(m,n), intent(inout) *c*, type(errorhandler), intent(inout) *err* )

Multiplies a general matrix by the orthogonal matrix Z from an RZ factorization such that: C = op(Z) \* C.

#### **Parameters**

in	trans	Set to true to apply Z**T; else, set to false.
in	m	The number of rows in the matrix c.
in	n	The number of columns in the matrix c.
in	1	The number of columns in matrix a containing the meaningful part of the Householder vectors (M $>=$ L $>=$ 0).
in,out	а	On input, the M-by-M matrix Z as output by $rz\_factor$ . The matrix is used as in-place storage during execution; however, the contents of the matrix are restored on exit.
in	tau	An M-element array containing the scalar factors of the elementary reflectors found in a.
in,out	С	On input, the M-by-N matrix C. On output, the product of the orthogonal matrix Z and the original matrix C.
in,out	err	The errorhandler object. If no error handling is desired, simply pass NULL, and errors will be dealt with by the default internal error handler. Possible errors that may be encountered are as follows.  • LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.

Definition at line 969 of file linalg\_c\_binding.f90.

4.2.2.25 subroutine linalg\_c\_binding::qr\_factor\_c ( integer(i32), intent(in), value *m*, integer(i32), intent(in), value *n*, real(dp), dimension(m,n), intent(inout) *a*, real(dp), dimension(min(m,n)), intent(out) *tau*, type(errorhandler), intent(inout) *err* )

Computes the QR factorization of an M-by-N matrix without pivoting.

#### **Parameters**

in	m	The number of rows in the matrix.
in	n	The number of columns in the matrix.
in,out	а	On input, the M-by-N matrix to factor. On output, the elements on and above the diagonal contain the MIN(M, N)-by-N upper trapezoidal matrix R (R is upper triangular if $M \ge N$ ). The elements below the diagonal, along with the array tau, represent the orthogonal matrix Q as a product of elementary reflectors.
out	tau	A MIN(M, N)-element array used to store the scalar factors of the elementary reflectors.
in,out	err	The errorhandler object. If no error handling is desired, simply pass NULL, and errors will be dealt with by the default internal error handler. Possible errors that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if the scalar factor array is not sized appropriately.  • LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and
		there is insufficient memory available.

Definition at line 553 of file linalg\_c\_binding.f90.

4.2.2.26 subroutine linalg\_c\_binding::qr\_factor\_pivot\_c ( integer(i32), intent(in), value *m*, integer(i32), intent(in), value *n*, real(dp), dimension(m,n), intent(inout) *a*, real(dp), dimension(min(m,n)), intent(out) *tau*, integer(i32), dimension(n), intent(inout) *jpvt*, type(errorhandler), intent(inout) *err* )

Computes the QR factorization of an M-by-N matrix with column pivoting such that A \* P = Q \* R.

in	m	The number of rows in the matrix.
in	n	The number of columns in the matrix.
in,out	а	On input, the M-by-N matrix to factor. On output, the elements on and above the diagonal contain the MIN(M, N)-by-N upper trapezoidal matrix R (R is upper triangular if $M \ge N$ ). The elements below the diagonal, along with the array tau, represent the orthogonal matrix Q as a product of elementary reflectors.
out	tau	A MIN(M, N)-element array used to store the scalar factors of the elementary reflectors.
in,out	jpvt	On input, an N-element array that if $JPVT(I)$ .ne. 0, the I-th column of A is permuted to the front of A $*$ P; if $JPVT(I) = 0$ , the I-th column of A is a free column. On output, if $JPVT(I) = K$ , then the I-th column of A $*$ P was the K-th column of A.
in,out	err	The errorhandler object. If no error handling is desired, simply pass NULL, and errors will be dealt with by the default internal error handler. Possible errors that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if the scalar factor array is not sized appropriately.  • LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.

Definition at line 598 of file linalg\_c\_binding.f90.

4.2.2.27 subroutine linalg\_c\_binding::qr\_rank1\_update\_c ( integer(i32), intent(in), value *m*, integer(i32), intent(in), value *n*, real(dp), dimension(m,m), intent(inout) *q*, real(dp), dimension(m,n), intent(inout) *r*, real(dp), dimension(m), intent(inout) *u*, real(dp), dimension(n), intent(inout) *v*, type(errorhandler), intent(inout) *err* )

Computes the rank 1 update to an M-by-N QR factored matrix A (M >= N) where A = Q \* R, and A1 = A + U \* V\*\*T such that A1 = Q1 \* R1.

#### **Parameters**

in	m	The number of rows in the original matrix.
in	n	The number of columns in the original matrix.
in,out	q	On input, the original M-by-M orthogonal matrix Q. On output, the updated matrix Q1.
in,out	r	On input, the M-by-N matrix R. On output, the updated matrix R1.
in,out	и	On input, the M-element U update vector. On output, the original content of the array is overwritten.
in,out	V	On input, the N-element V update vector. On output, the original content of the array is
		overwritten.
in,out	err	The errorhandler object. If no error handling is desired, simply pass NULL, and errors will be dealt with by the default internal error handler. Possible errors that may be encountered are as follows.
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>

Definition at line 778 of file linalg c binding.f90.

4.2.2.28 subroutine linalg\_c\_binding::rank1\_update\_c ( integer(i32), intent(in), value *m*, integer(i32), intent(in), value *n*, real(dp), intent(in), value *alpha*, real(dp), dimension(m), intent(in) *x*, real(dp), dimension(n), intent(in) *y*, real(dp), dimension(m,n), intent(inout) *a* )

Performs the rank-1 update to matrix A such that: A = alpha \* X \* Y \* \* T + A, where A is an M-by-N matrix, alpha is a scalar, X is an M-element array, and N is an N-element array.

#### Parameters

in	m	The number of elements in $\mathbf{x}$ , and the number of rows in matrix $\mathbf{a}$ .
in	n	The number of elements in $y$ , and the number of columns in matrix $a$ .
in	alpha	The scalar multiplier.
in	X	An M-element array.
in	У	An N-element array.
in,out	а	On input, the M-by-N matrix to update. On output, the updated M-by-N matrix.

### Notes

This routine is based upon the BLAS routine DGER.

Definition at line 294 of file linalg\_c\_binding.f90.

4.2.2.29 subroutine linalg\_c\_binding::rz\_factor\_c ( integer(i32), intent(in), value *m*, integer(i32), intent(in), value *n*, real(dp), dimension(m,n), intent(inout) *a*, real(dp), dimension(m), intent(out) *tau*, type(errorhandler), intent(inout) *err* )

Factors an upper trapezoidal matrix by means of orthogonal transformations such that  $A = R * Z = (R \ 0) * Z$ . Z is an orthogonal matrix of dimension N-by-N, and R is an M-by-M upper triangular matrix.

#### **Parameters**

in	m	The number of rows in the original matrix.
in	n	The number of columns in the original matrix.
in,out	а	On input, the M-by-N upper trapezoidal matrix to factor. On output, the leading M-by-M upper triangular part of the matrix contains the upper triangular matrix R, and elements N-L+1 to N of the first M rows of A, with the array tau, represent the orthogonal matrix Z as a product of M elementary reflectors.
out	tau	An M-element array used to store the scalar factors of the elementary reflectors.
in,out	err	The errorhandler object. If no error handling is desired, simply pass NULL, and errors will be dealt with by the default internal error handler. Possible errors that may be encountered are as follows.  • LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.
		there is insumition in the mory available.

Definition at line 927 of file linalg\_c\_binding.f90.

4.2.2.30 subroutine linalg\_c\_binding::solve\_cholesky\_c ( logical(c\_bool), intent(in), value *upper*, integer(i32), intent(in), value *n*, integer(i32), intent(in), value *n*, real(dp), dimension(n,n), intent(in) *a*, real(dp), dimension(n,nrhs), intent(inout) *b* )

Solves a system of Cholesky factored equations.

#### **Parameters**

in	upper	Set to true if the original matrix A was factored such that $A = U**T*U$ ; else, set to false if
		the factorization of A was $A = L**T*L$ .
in	n	The dimension of the original matrix a.
in	nrhs	The number of right-hand-side vectors (number of columns in matrix b).
in	а	The N-by-N Cholesky factored matrix.
in,out	b	On input, the N-by-NRHS right-hand-side matrix B. On output, the solution matrix X.

Definition at line 1190 of file linalg\_c\_binding.f90.

4.2.2.31 subroutine linalg\_c\_binding::solve\_least\_squares\_c ( integer(i32), intent(in), value *m*, integer(i32), intent(in), value *n*, integer(i32), intent(in), value *n*, real(dp), dimension(m, n), intent(inout) *a*, real(dp), dimension(max(m,n), nrhs), intent(inout) *b*, type(errorhandler), intent(inout) *err* )

Solves the overdetermined or underdetermined system (A\*X = B) of M equations of N unknowns using a QR or LQ factorization of the matrix A. Notice, it is assumed that matrix A has full rank.

The number of tows in the original coefficient matrix 7.	in /	m	The number of rows in the original coefficient matrix A.
--	------	---	--

#### **Parameters**

in	n	The number of columns in the original coefficient matrix A.
in	nrhs	The number of right-hand-side vectors (number of columns in matrix b).
in,out	а	On input, the M-by-N matrix A. On output, the matrix is overwritten by the details of its complete orthogonal factorization.
in,out	b	If $M >= N$ , the M-by-NRHS matrix B. On output, the first N rows contain the N-by-NRHS solution matrix X. If $M < N$ , an N-by-NRHS matrix with the first M rows containing the matrix B. On output, the N-by-NRHS solution matrix X.
in,out	err	The errorhandler object. If no error handling is desired, simply pass NULL, and errors will be dealt with by the default internal error handler. Possible errors that may be encountered are as follows.  • LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.

Definition at line 1292 of file linalg\_c\_binding.f90.

4.2.2.32 subroutine linalg\_c\_binding::solve\_lu\_c ( integer(i32), intent(in), value *n*, integer(i32), intent(in), value *n*, real(dp), dimension(n,n), intent(in) *a*, integer(i32), dimension(n), intent(in) *ipvt*, real(dp), dimension(n,nrhs), intent(inout) *b* )

Solves a system of LU-factored equations.

#### **Parameters**

in	n	The dimension of the original matrix a.
in	nrhs	The number of right-hand-side vectors (number of columns in matrix b).
in	а	The N-by-N LU factored matrix as output by lu_factor.
in	ipvt	The N-element pivot array as output by lu_factor.
in,out	b	On input, the N-by-NRHS right-hand-side matrix. On output, the N-by-NRHS solution matrix.

Definition at line 1079 of file linalg\_c\_binding.f90.

4.2.2.33 subroutine linalg\_c\_binding::solve\_qr\_c ( integer(i32), intent(in), value *m*, integer(i32), intent(in), value *nrhs*, real(dp), dimension(m,n), intent(inout) *a*, real(dp), dimension(min(m,n)), intent(in) *tau*, real(dp), dimension(m,nrhs), intent(inout) *b*, type(errorhandler), intent(inout) *err* )

Solves a system of M QR-factored equations of N unknowns where  $M \ge N$ .

in	m	The number of rows in the original coefficient matrix A.
in	n	The number of columns in the original coefficient matrix A.
in	nrhs	The number of right-hand-side vectors (number of columns in matrix b).
in	а	On input, the M-by-N QR factored matrix as returned by qr_factor. On output, the contents of this matrix are restored. Notice, M must be greater than or equal to N.
in	tau	A MIN(M, N)-element array containing the scalar factors of the elementary reflectors as returned by qr_factor.

#### **Parameters**

in	b	On input, the M-by-NRHS right-hand-side matrix. On output, the first N columns are overwritten by the solution matrix X.
in,out	err	The errorhandler object. If no error handling is desired, simply pass NULL, and errors will be dealt with by the default internal error handler. Possible errors that may be encountered are as follows.
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>

Definition at line 1112 of file linalg\_c\_binding.f90.

4.2.2.34 subroutine linalg\_c\_binding::solve\_qr\_pivot\_c ( integer(i32), intent(in), value *m*, integer(i32), intent(in), value *n*, integer(i32), intent(in), value *nrhs*, real(dp), dimension(m,n), intent(inout) *a*, real(dp), dimension(min(m,n)), intent(in) *tau*, integer(i32), dimension(n), intent(in) *jpvt*, real(dp), dimension(max(m,n),nrhs), intent(inout) *b*, type(errorhandler), intent(inout) *err* )

Solves a system of M QR-factored equations of N unknowns where the QR factorization made use of column pivoting.

#### **Parameters**

in	m	The number of rows in the original coefficient matrix A.
in	n	The number of columns in the original coefficient matrix A.
in	nrhs	The number of right-hand-side vectors (number of columns in matrix b).
in	а	On input, the M-by-N QR factored matrix as returned by qr_factor. On output, the contents of this matrix are altered.
in	tau	A MIN(M, N)-element array containing the scalar factors of the elementary reflectors as returned by qr_factor.
in	jpvt	An N-element array, as output by qr_factor, used to track the column pivots.
in	b	On input, the MAX(M, N)-by-NRHS matrix where the first M rows contain the right-hand-side matrix B. On output, the first N rows are overwritten by the solution matrix X.
in,out	err	The errorhandler object. If no error handling is desired, simply pass NULL, and errors will be dealt with by the default internal error handler. Possible errors that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.  • LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.

Definition at line 1157 of file linalg\_c\_binding.f90.

4.2.2.35 subroutine linalg\_c\_binding::solve\_tri\_mtx\_c ( logical(c\_bool), intent(in), value *upper,* logical(c\_bool), intent(in), value *trans,* logical(c\_bool), intent(in), value *nounit,* integer(i32), intent(in), value *n,* integer(i32), intent(in), value *nrhs,* real(dp), intent(in), value *alpha,* real(dp), dimension(n,n), intent(in) *a,* real(dp), dimension(n,nrhs), intent(inout) *b* )

Solves one of the matrix equations: op(A) \* X = alpha \* B, where A is a triangular matrix.

in	upper	Set to true if A is an upper triangular matrix; else, set to false if A is a lower triangular matrix.
in	trans	Set to true if $op(A) = A**T$ ; else, set to false if $op(A) = A$ .
in	nounit	Set to true if A is not a unit-diagonal matrix (ones on every diagonal element); else, set to false if A is a unit-diagonal matrix.
in	n	The dimension of the triangular matrix a.
in	nrhs	The number of right-hand-side vectors (number of columns in matrix b).
in	alpha	The scalar multiplier to B.
in	а	N-by-N triangular matrix on which to operate.
in,out	b	On input, the N-by-NRHS right-hand-side. On output, the N-by-NRHS solution.

Definition at line 1056 of file linalg\_c\_binding.f90.

4.2.2.36 subroutine linalg\_c\_binding::sort\_cmplx\_ind\_c ( logical(c\_bool), intent(in), value ascend, integer(i32), intent(in), value n, complex(dp), dimension(n), intent(inout) x, type(c\_ptr), intent(in), value ind )

Sorts an array of complex values according to their real components.

#### **Parameters**

in	ascend	Set to true to sort in ascending order; else, false to sort in descending order.
in	n	The number of elements in the array.
in,out	X	On input, the N-element array to sort. On output, the sorted array.
in,out	ind	On input, a pointer to an integer array. If NULL, this argument is ignored, and $x$ is sorted as expected. However, if used, on output, the contents of this array are shifted in the same order as that of $x$ as a means of tracking the sorting operation. It is often useful to set this array to an ascending group of values $(1, 2, n)$ such that this array tracks the original positions of the sorted array. Such an array can then be used to align other arrays. This array must be the same size as $x$ .

Definition at line 1495 of file linalg\_c\_binding.f90.

4.2.2.37 subroutine linalg\_c\_binding::sort\_dbl\_ind\_c ( logical(c\_bool), intent(in), value ascend, integer(i32), intent(in), value n, real(dp), dimension(n), intent(inout) x, type(c\_ptr), intent(in), value ind )

Sorts an array of double-precision values.

in	ascend	Set to true to sort in ascending order; else, false to sort in descending order.
in	n	The number of elements in the array.
in,out	X	On input, the N-element array to sort. On output, the sorted array.
in,out	ind	On input, a pointer to an integer array. If NULL, this argument is ignored, and $x$ is sorted as expected. However, if used, on output, the contents of this array are shifted in the same order as that of $x$ as a means of tracking the sorting operation. It is often useful to set this array to an ascending group of values $(1, 2, n)$ such that this array tracks the original positions of the sorted array. Such an array can then be used to align other arrays. This array must be the same size as $x$ .

Definition at line 1459 of file linalg\_c\_binding.f90.

4.2.2.38 subroutine linalg\_c\_binding::sort\_eigen\_cmplx\_c ( logical(c\_bool), intent(in), value *ascend*, integer(i32), intent(in), value *n*, complex(dp), dimension(n), intent(inout) *vals*, complex(dp), dimension(n,n), intent(inout) *vecs* )

A sorting routine specifically tailored for sorting of eigenvalues and their associated eigenvectors using a quick-sort approach.

#### **Parameters**

in	ascend	Set to true to sort in ascending order; else, false to sort in descending order.	
in	n	The number of eigenvalues.	
in,out	vals	On input, an N-element array containing the eigenvalues. On output, the sorted eigenvalues.	
in,out	vecs	On input, an N-by-N matrix containing the eigenvectors associated with $vals$ (one vector per column). On output, the sorted eigenvector matrix.	

Definition at line 1527 of file linalg\_c\_binding.f90.

4.2.2.39 subroutine linalg\_c\_binding::sort\_eigen\_dbl\_c ( logical(c\_bool), intent(in), value ascend, integer(i32), intent(in), value n, real(dp), dimension(n), intent(inout) vals, real(dp), dimension(n,n), intent(inout) vecs )

A sorting routine specifically tailored for sorting of eigenvalues and their associated eigenvectors using a quick-sort approach.

# **Parameters**

in	ascend	Set to true to sort in ascending order; else, false to sort in descending order.	
in	n	The number of eigenvalues.	
in,out	vals	On input, an N-element array containing the eigenvalues. On output, the sorted eigenvalues.	
in,out	vecs	On input, an N-by-N matrix containing the eigenvectors associated with $vals$ (one vector per column). On output, the sorted eigenvector matrix.	

Definition at line 1550 of file linalg\_c\_binding.f90.

4.2.2.40 subroutine linalg\_c\_binding::svd\_c ( integer(i32), intent(in), value *m*, integer(i32), intent(in), value *n*, real(dp), dimension(m,n), intent(inout) *a*, real(dp), dimension(min(m,n)), intent(out) *s*, real(dp), dimension(m,m), intent(out) *vt*, type(errorhandler), intent(inout) *err* )

Computes the singular value decomposition of a matrix A. The SVD is defined as: A = U \* S \* V \* \* T, where U is an M-by-M orthogonal matrix, S is an M-by-N diagonal matrix, and V is an N-by-N orthogonal matrix.

in	m	The number of rows in the original matrix.
in	n	The number of columns in the original matrix.
in,out	а	On input, the M-by-N matrix to factor. The matrix is overwritten on output. that the remaining matrix is simply the M-by-N matrix R.

out	s	A MIN(M, N)-element array containing the singular values of a sorted in descending order.
out	и	An M-by-M matrix that on output contains the left singular vectors (matrix U in the
		decomposition: $A = U * S * V * * T$ )
out	vt	An N-by-N matrix that on output contains the right singular vectors (matrix V**T in the
		decomposition: $A = U * S * V * * T$ ).
in,out	err	The errorhandler object. If no error handling is desired, simply pass NULL, and errors will be dealt with by the default internal error handler. Possible errors that may be encountered are as follows.
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if the singular value array is not sized appropriately.</li> </ul>
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>
		<ul> <li>LA_CONVERGENCE_ERROR: Occurs as a warning if the QR iteration process could not converge to a zero value.</li> </ul>

Definition at line 1016 of file linalg\_c\_binding.f90.

4.2.2.41 subroutine linalg\_c\_binding::swap\_c ( integer(i32), intent(in), value n, real(dp), dimension(n), intent(inout) x, real(dp), dimension(n), intent(inout) y )

Swaps the contents of two arrays.

# **Parameters**

in	n	The number of elements either array.
in,out	Х	One of the N-element arrays.
in,out	у	The other N-element array.

Definition at line 399 of file linalg\_c\_binding.f90.

4.2.2.42 pure real(dp) function linalg\_c\_binding::trace\_c ( integer(i32), intent(in), value *m*, integer(i32), intent(in), value *n*, real(dp), dimension(m,n), intent(in) *x* )

Computes the trace of a matrix (the sum of the main diagonal elements).

#### **Parameters**

in	m	The number of rows in the matrix.
in	n	The number of columns in the matrix.
in	X	The matrix on which to operate.

# Returns

The trace of  $\mathbf{x}$ .

Definition at line 314 of file linalg\_c\_binding.f90.

# 4.3 linalg\_constants Module Reference

# linalg\_constants

# **Variables**

• integer, parameter dp = real64

Defines a double-precision (64-bit) floating-point type.

• integer, parameter i32 = int32

Defines a 32-bit signed integer type.

• integer, parameter la\_invalid\_input\_error = 101

An error flag denoting an invalid input.

• integer, parameter la\_array\_size\_error = 102

An error flag denoting an improperly sized array.

• integer, parameter la\_singular\_matrix\_error = 103

An error flag denoting a singular matrix.

integer, parameter la\_matrix\_format\_error = 104

An error flag denoting an issue with the matrix format.

• integer, parameter la\_out\_of\_memory\_error = 105

An error flag denoting that there is insufficient memory available.

• integer, parameter la\_convergence\_error = 106

An error flag denoting a convergence failure.

• integer, parameter la\_invalid\_operation\_error = 107

An error resulting from an invalid operation.

# 4.3.1 Detailed Description

#### linalg\_constants

# **Purpose**

Provides a set of constants and error flags for the library.

# 4.4 linalg\_core Module Reference

# linalg\_core

# **Data Types**

· interface diag\_mtx\_mult

Multiplies a diagonal matrix with another matrix or array.

interface mtx\_mult

Performs the matrix operation: C = alpha \* op(A) \* op(B) + beta \* C.

#### **Functions/Subroutines**

• subroutine mtx\_mult\_mtx (transa, transb, alpha, a, b, beta, c, err)

Performs the matrix operation: C = alpha \* op(A) \* op(B) + beta \* C.

• subroutine mtx\_mult\_vec (trans, alpha, a, b, beta, c, err)

Performs the matrix-vector operation: c = alpha \* op(A) \* b + beta \* c.

• subroutine, public rank1\_update (alpha, x, y, a, err)

Performs the rank-1 update to matrix A such that: A = alpha \* X \* Y \* \* T + A, where A is an M-by-N matrix, alpha is a scalar, X is an M-element array, and N is an N-element array.

• subroutine diag\_mtx\_mult\_mtx (Iside, trans, alpha, a, b, beta, c, err)

Computes the matrix operation: C = alpha \* A \* op(B) + beta \* C, or C = alpha \* op(B) \* A + beta \* C.

• subroutine diag mtx mult mtx2 (Iside, alpha, a, b, err)

Computes the matrix operation: B = alpha \* A \* op(B), or B = alpha \* op(B) \* A.

• subroutine diag mtx mult mtx3 (Iside, trans, alpha, a, b, beta, c, err)

Computes the matrix operation: C = alpha \* A \* op(B) + beta \* C, or C = alpha \* op(B) \* A + beta \* C, where A and C are complex-valued.

• subroutine diag\_mtx\_mult\_mtx4 (Iside, trans, alpha, a, b, beta, c, err)

Computes the matrix operation: C = alpha \* A \* op(B) + beta \* C, or C = alpha \* op(B) \* A + beta \* C, where A, B, and C are complex-valued.

pure real(dp) function, public trace (x)

Computes the trace of a matrix (the sum of the main diagonal elements).

integer(i32) function, public mtx rank (a, tol, work, olwork, err)

Computes the rank of a matrix.

real(dp) function, public det (a, iwork, err)

Computes the determinant of a square matrix.

• subroutine, public swap (x, y, err)

Swaps the contents of two arrays.

• subroutine, public recip\_mult\_array (a, x)

Multiplies a vector by the reciprocal of a real scalar.

• subroutine, public tri mtx mult (upper, alpha, a, beta, b, err)

Computes the triangular matrix operation: B = alpha \* A\*\*T \* A + beta \* B, or B = alpha \* A \* A\*\*T + beta \* B, where A is a triangular matrix.

# 4.4.1 Detailed Description

### linalg core

#### **Purpose**

Provides common "core" linear algebra routines.

### 4.4.2 Function/Subroutine Documentation

4.4.2.1 real(dp) function, public linalg\_core::det ( real(dp), dimension(:,:), intent(inout) a, integer(i32), dimension(:), intent(out), optional, pointer iwork, class(errors), intent(inout), optional, target err )

Computes the determinant of a square matrix.

# **Parameters**

in,out	а	On input, the N-by-N matrix on which to operate. On output the contents are overwritten by the LU factorization of the original matrix.
out	iwork	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least N-elements.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.</li> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>

# Returns

The determinant of a.

Definition at line 1088 of file linalg\_core.f90.

4.4.2.2 subroutine linalg\_core::diag\_mtx\_mult\_mtx ( logical, intent(in) *lside*, logical, intent(in) *trans*, real(dp) *alpha*, real(dp), dimension(:), intent(in) a, real(dp), dimension(:,:), intent(in) b, real(dp) beta, real(dp), dimension(:,:), intent(inout) c, class(errors), intent(inout), optional, target err ) [private]

Computes the matrix operation: C = alpha \* A \* op(B) + beta \* C, or C = alpha \* op(B) \* A + beta \* C.

### **Parameters**

in	Iside	Set to true to apply matrix A from the left; else, set to false to apply matrix A from the left.	
in	trans	Set to true if $op(B) == B**T$ ; else, set to false if $op(B) == B$ .	
in	alpha	A scalar multiplier.	
in	а	A K-element array containing the diagonal elements of A where $K = MIN(M,P)$ if lside is true; else, if lside is false, $K = MIN(N,P)$ .	
in	b	The LDB-by-TDB matrix B where (LDB = leading dimension of B, and TDB = trailing dimension of B):  • lside == true & trans == true: LDB = N, TDB = P  • lside == true & trans == false: LDB = P, TDB = N  • lside == false & trans == true: LDB = P, TDB = M  • lside == false & trans == false: LDB = M, TDB = P	
in	beta	A scalar multiplier.	
in,out	С	On input, the M-by-N matrix C. On output, the resulting M-by-N matrix.	
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if any of the input array sizes are incorrect.	
		Generated by Doxygen	

Generated by Doxygen

Definition at line 329 of file linalg\_core.f90.

4.4.2.3 subroutine linalg\_core::diag\_mtx\_mult\_mtx2 ( logical, intent(in) *lside*, real(dp), intent(in) *alpha*, real(dp), dimension(:), intent(in) *a*, real(dp), dimension(:,:), intent(inout) *b*, class(errors), intent(inout), optional, target *err* ) [private]

Computes the matrix operation: B = alpha \* A \* op(B), or B = alpha \* op(B) \* A.

#### **Parameters**

in	Iside	Set to true to apply matrix A from the left; else, set to false to apply matrix A from the left.
in	alpha	A scalar multiplier.
in	а	A K-element array containing the diagonal elements of A where $K = MIN(M,P)$ if lside is true; else, if lside is false, $K = MIN(N,P)$ .
in	b	On input, the M-by-N matrix B. On output, the resulting M-by-N matrix.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if any of the input array sizes are incorrect.

Definition at line 496 of file linalg\_core.f90.

4.4.2.4 subroutine linalg\_core::diag\_mtx\_mult\_mtx3 ( logical, intent(in) *lside,* logical, intent(in) *trans,* real(dp) *alpha,* complex(dp), dimension(:), intent(in) *a,* real(dp), dimension(:,:), intent(in) *b,* real(dp) *beta,* complex(dp), dimension(:,:), intent(inout) *c,* class(errors), intent(inout), optional, target *err* ) [private]

Computes the matrix operation: C = alpha \* A \* op(B) + beta \* C, or C = alpha \* op(B) \* A + beta \* C, where A and C are complex-valued.

Iside	Cat to true to apply matrix A from the left, also not to falso to apply matrix A from the left	
	Set to true to apply matrix A from the left; else, set to false to apply matrix A from the left.	
trans	Set to true if $op(B) == B**T$ ; else, set to false if $op(B) == B$ .	
alpha	A scalar multiplier.	
а	A K-element array containing the diagonal elements of A where $K = MIN(M,P)$ if lside is true; else, if lside is false, $K = MIN(N,P)$ .	
b	The LDB-by-TDB matrix B where (LDB = leading dimension of B, and TDB = trailing dimension of B):  • lside == true & trans == true: LDB = N, TDB = P  • lside == true & trans == false: LDB = P, TDB = N  • lside == false & trans == true: LDB = P, TDB = M  • lside == false & trans == false: LDB = M, TDB = P	
beta	A scalar multiplier.	
С	On input, the M-by-N matrix C. On output, the resulting M-by-N matrix.	
	alpha a b	

#### **Parameters**

out	err	An optional errors-based object that if provided can be used to retrieve information relating
		to any errors encountered during execution. If not provided, a default implementation of
		the errors class is used internally to provide error handling. Possible errors and warning
		messages that may be encountered are as follows.
		LA_ARRAY_SIZE_ERROR: Occurs if any of the input array sizes are incorrect.

Definition at line 579 of file linalg\_core.f90.

4.4.2.5 subroutine linalg\_core::diag\_mtx\_mult\_mtx4 ( logical, intent(in) *lside*, logical, intent(in) *trans*, real(dp) *alpha*, complex(dp), dimension(:,:), intent(in) *b*, real(dp) *beta*, complex(dp), dimension(:,:), intent(inout) *c*, class(errors), intent(inout), optional, target *err* ) [private]

Computes the matrix operation: C = alpha \* A \* op(B) + beta \* C, or C = alpha \* op(B) \* A + beta \* C, where A, B, and C are complex-valued.

#### **Parameters**

in	Iside	Set to true to apply matrix A from the left; else, set to false to apply matrix A from the left.	
in	trans	Set to true if $op(B) == B**T$ ; else, set to false if $op(B) == B$ .	
in	alpha	A scalar multiplier.	
in	а	A K-element array containing the diagonal elements of A where $K = MIN(M,P)$ if lside is true; else, if lside is false, $K = MIN(N,P)$ .	
in	b	The LDB-by-TDB matrix B where:	
		• lside == true & trans == true: LDA = N, TDB = P	
		• lside == true & trans == false: LDA = P, TDB = N	
		• lside == false & trans == true: LDA = P, TDB = M	
		• lside == false & trans == false: LDA = M, TDB = P	
in	beta	A scalar multiplier.	
in,out	С	On input, the M-by-N matrix C. On output, the resulting M-by-N matrix.	
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if any of the input array sizes are incorrect.	

Definition at line 755 of file linalg\_core.f90.

4.4.2.6 subroutine linalg\_core::mtx\_mult\_mtx ( logical, intent(in) transa, logical, intent(in) transb, real(dp), intent(in) alpha, real(dp), dimension(:,:), intent(in) a, real(dp), dimension(:,:), intent(in) b, real(dp), intent(in) beta, real(dp), dimension(:,:), intent(inout) c, class(errors), intent(inout), optional, target err ) [private]

Performs the matrix operation: C = alpha \* op(A) \* op(B) + beta \* C.

in	transa	Set to true if $op(A) = A**T$ ; else, set to false for $op(A) = A$ .
in	transb	Set to true if $op(B) = B**T$ ; else, set to false for $op(B) = B$ .
in	alpha	A scalar multiplier.
in	а	If transa is set to true, an K-by-M matrix; else, if transa is set to false, an M-by-K matrix.
in	b	If transb is set to true, an N-by-K matrix; else, if transb is set to false, a K-by-N matrix.
in	beta	A scalar multiplier.
in,out	С	On input, the M-by-N matrix C. On output, the M-by-N result.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if any of the input array sizes are incorrect.

# Notes

This routine utilizes the BLAS routine DGEMM.

Definition at line 85 of file linalg\_core.f90.

4.4.2.7 subroutine linalg\_core::mtx\_mult\_vec ( logical, intent(in) trans, real(dp), intent(in) alpha, real(dp), dimension(:,:), intent(in) a, real(dp), dimension(:), intent(in) b, real(dp), intent(in) beta, real(dp), dimension(:), intent(inout) c, class(errors), intent(inout), optional, target err ) [private]

Performs the matrix-vector operation: c = alpha \* op(A) \* b + beta \* c.

# **Parameters**

in	trans	Set to true if $op(A) = A**T$ ; else, set to false for $op(A) = A$ .
in	alpha	A scalar multiplier.
in	a The M-by-N matrix A.	
in	b	If trans is set to true, an M-element array; else, if trans is set to false, an N-element array.
in	beta	A scalar multiplier.
in,out	С	On input, if trans is set to true, an N-element array; else, if trans is set to false, an M-element array. On output, the results of the operation.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if any of the input array sizes are incorrect.

# Notes

This routine utilizes the BLAS routine DGEMV.

Definition at line 179 of file linalg\_core.f90.

4.4.2.8 integer(i32) function, public linalg\_core::mtx\_rank ( real(dp), dimension(:,:), intent(inout) *a,* real(dp), intent(in), optional *tol,* real(dp), dimension(:), intent(out), optional, pointer *work,* integer(i32), intent(out), optional *olwork,* class(errors), intent(inout), optional, target *err* )

Computes the rank of a matrix.

#### **Parameters**

in,out	а	On input, the M-by-N matrix of interest. On output, the contents of the matrix are overwritten.
in	tol	An optional input, that if supplied, overrides the default tolerance on singular values such that singular values less than this tolerance are treated as zero. The default tolerance is: MAX(M, N) * EPS * MAX(S). If the supplied value is less than the smallest value that causes an overflow if inverted, the tolerance reverts back to its default value, and the operation continues; however, a warning message is issued.
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.
out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for work, and returns without performing any actual calculations.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.</li> </ul>
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>
		<ul> <li>LA_CONVERGENCE_ERROR: Occurs as a warning if the QR iteration process could not converge to a zero value.</li> </ul>

### See Also

• Wolfram MathWorld

Definition at line 968 of file linalg\_core.f90.

4.4.2.9 subroutine, public linalg\_core::rank1\_update ( real(dp), intent(in) alpha, real(dp), dimension(:), intent(in) x, real(dp), dimension(:), intent(in) y, real(dp), dimension(:,:), intent(inout) a, class(errors), intent(inout), optional, target err )

Performs the rank-1 update to matrix A such that: A = alpha \* X \* Y \* \* T + A, where A is an M-by-N matrix, alpha is a scalar, X is an M-element array, and N is an N-element array.

in	alpha	The scalar multiplier.
in	X	An M-element array.

in	У	An N-element array.
in,out	а	On input, the M-by-N matrix to update. On output, the updated M-by-N matrix.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if the size of a does not match with x and y.

# Notes

This routine is based upon the BLAS routine DGER.

Definition at line 257 of file linalg\_core.f90.

4.4.2.10 subroutine, public linalg\_core::recip\_mult\_array ( real(dp), intent(in) a, real(dp), dimension(:), intent(inout) x )

Multiplies a vector by the reciprocal of a real scalar.

# **Parameters**

in	а	The scalar which is used to divide each component of $X$ . The value must be $>=0$ , or the subroutine will divide by zero.
in,out	X	The vector.

# Notes

This routine is based upon the LAPACK routine DRSCL.

Definition at line 1239 of file linalg\_core.f90.

4.4.2.11 subroutine, public linalg\_core::swap ( real(dp), dimension(:), intent(inout) x, real(dp), dimension(:), intent(inout) y, class(errors), intent(inout), optional, target err )

Swaps the contents of two arrays.

in,out	X	One of the N-element arrays.
in,out	У	The other N-element array.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		• LA_ARRAY_SIZE_ERROR: Occurs if $\boldsymbol{x}$ and $\boldsymbol{y}$ are not the same size.

Definition at line 1193 of file linalg\_core.f90.

4.4.2.12 pure real(dp) function, public linalg\_core::trace ( real(dp), dimension(:,:), intent(in) x )

Computes the trace of a matrix (the sum of the main diagonal elements).

#### **Parameters**

in	X	The matrix on which to operate.
----	---	---------------------------------

#### Returns

The trace of x.

Definition at line 912 of file linalg\_core.f90.

4.4.2.13 subroutine, public linalg\_core::tri\_mtx\_mult ( logical, intent(in) upper, real(dp), intent(in) alpha, real(dp), dimension(:,:), intent(in) a, real(dp), intent(in) beta, real(dp), dimension(:,:), intent(inout) b, class(errors), intent(inout), optional, target err )

Computes the triangular matrix operation: B = alpha \* A\*\*T \* A + beta \* B, or B = alpha \* A \* A\*\*T + beta \* B, where A is a triangular matrix.

### **Parameters**

in	upper	Set to true if matrix A is upper triangular, and B = alpha $*$ A**T $*$ A + beta $*$ B is to be calculated; else, set to false if A is lower triangular, and B = alpha $*$ A $*$ A**T + beta $*$ B is to be computed.
in	alpha	A scalar multiplier.
in	а	The N-by-N triangular matrix. Notice, if upper is true only the upper triangular portion of this matrix is referenced; else, if upper is false, only the lower triangular portion of this matrix is referenced.
in	beta	A scalar multiplier.
in,out	b	On input, the N-by-N matrix B. On output, the N-by-N solution matrix.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized
		appropriately.

Definition at line 1316 of file linalg\_core.f90.

# 4.5 linalg\_eigen Module Reference

# linalg\_eigen

# **Data Types**

• interface eigen

Computes the eigenvalues, and optionally the eigenvectors, of a matrix.

# **Functions/Subroutines**

• subroutine eigen\_symm (vecs, a, vals, work, olwork, err)

Computes the eigenvalues, and optionally the eigenvectors of a real, symmetric matrix.

• subroutine eigen\_asymm (a, vals, vecs, work, olwork, err)

Computes the eigenvalues, and optionally the right eigenvectors of a square matrix.

• subroutine eigen\_gen (a, b, alpha, beta, vecs, work, olwork, err)

Computes the eigenvalues, and optionally the right eigenvectors of a square matrix assuming the structure of the eigenvalue problem is A\*X = lambda\*B\*X.

# 4.5.1 Detailed Description

# linalg\_eigen

#### **Purpose**

Provides routines for computing the eigenvalues and eigenvectors of matrices.

# 4.5.2 Function/Subroutine Documentation

4.5.2.1 subroutine linalg\_eigen::eigen\_asymm ( real(dp), dimension(:,:), intent(inout) a, complex(dp), dimension(:), intent(out) vals, complex(dp), dimension(:,:), intent(out), optional vecs, real(dp), dimension(:), intent(out), optional, pointer work, integer(i32), intent(out), optional olwork, class(errors), intent(inout), optional, target err ) [private]

Computes the eigenvalues, and optionally the right eigenvectors of a square matrix.

in,out	а	On input, the N-by-N matrix on which to operate. On output, the contents of this matrix are overwritten.
out	vals	An N-element array containing the eigenvalues of the matrix. The eigenvalues are not sorted.
out	vecs	An optional N-by-N matrix, that if supplied, signals to compute the right eigenvectors (one per column). If not provided, only the eigenvalues will be computed.
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.
out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for work, and returns without performing any actual calculations.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.</li> </ul>
Generated by D	oxygen	<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>
		<ul> <li>LA_CONVERGENCE_ERROR: Occurs if the algorithm failed to converge.</li> </ul>

#### Notes

This routine utilizes the LAPACK routine DGEEV.

Definition at line 185 of file linalg\_eigen.f90.

4.5.2.2 subroutine linalg\_eigen::eigen\_gen ( real(dp), dimension(:,:), intent(inout) a, real(dp), dimension(:,:), intent(inout) b, complex(dp), dimension(:), intent(out) alpha, real(dp), dimension(:), intent(out), optional beta, complex(dp), dimension(:,:), intent(out), optional vecs, real(dp), dimension(:), intent(out), optional, pointer work, integer(i32), intent(out), optional olwork, class(errors), intent(inout), optional, target err ) [private]

Computes the eigenvalues, and optionally the right eigenvectors of a square matrix assuming the structure of the eigenvalue problem is A\*X = lambda\*B\*X.

#### **Parameters**

in,out	а	On input, the N-by-N matrix A. On output, the contents of this matrix are overwritten.
in,out	b	On input, the N-by-N matrix B. On output, the contents of this matrix are overwritten.
out	alpha	An N-element array that, if beta is not supplied, contains the eigenvalues. If beta is supplied however, the eigenvalues must be computed as ALPHA / BETA. This however, is not as trivial as it seems as it is entirely possible, and likely, that ALPHA / BETA can overflow or underflow. With that said, the values in ALPHA will always be less than and usually comparable with the NORM(A).
out	beta	An optional N-element array that if provided forces alpha to return the numerator, and this array contains the denominator used to determine the eigenvalues as ALPHA / BETA. If used, the values in this array will always be less than and usually comparable with the NORM(B).
out	vecs	An optional N-by-N matrix, that if supplied, signals to compute the right eigenvectors (one per column). If not provided, only the eigenvalues will be computed.
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.
out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for work, and returns without performing any actual calculations.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.</li> </ul>
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>
		LA_CONVERGENCE_ERROR: Occurs if the algorithm failed to converge.

#### Usage

As an example, consider the eigenvalue problem arising from a mechanical system of masses and springs such that the masses are described by a mass matrix M, and the arrangement of springs are described by a stiffness matrix K.

- ! This is an example illustrating the use of the eigenvalue and eigenvector
- ! routines to solve a free vibration problem of 3 masses connected by springs.

```
As illustrated above, the system consists of 3 masses connected by springs. Spring k1 and spring k4 connect the end masses to ground. The equations of
        motion for this system are as follows.
 ! | m1 0 0 | |x1" | | k1+k2 -k2
! | 0 m2 0 | |x2" | + | -k2 k2+k3
! | 0 0 m3 | |x3" | 0 -k3
                                                                                               | k1+k2 -k2
                                                                                                                                                                          0 | |x1|
                                                                                                                                                                        -k3 \mid |x2| = |0|
                                                                                                                                                                   k3+k4| |x3|
       Notice: x1" = the second time derivative of x1.
program example
                use linalg_constants, only : dp, i32
                 use linalg_eigen
                implicit none
                ! Define the model parameters
                 real(dp), parameter :: pi = 3.14159265359d0
                 real(dp), parameter :: m1 = 0.5d0
                 real(dp), parameter :: m2 = 2.5d0
                 real(dp), parameter :: m3 = 0.75d0
                 real(dp), parameter :: k1 = 5.0d6
                 real(dp), parameter :: k2 = 10.0d6
                 real(dp), parameter :: k3 = 10.0d6
                 real(dp), parameter :: k4 = 5.0d6
                 ! Local Variables
                integer(i32) :: i, j
real(dp) :: m(3,3), k(3,3), natfreq(3)
             complex(dp) :: vals(3), modeshapes(3,3)
                 ! Define the mass matrix % \left( 1\right) =\left( 1\right) \left( 1\right) 
               m = reshape([m1, 0.0d0, 0.0d0, 0.0d0, m2, 0.0d0, 0.0d0, 0.0d0, m3], [3, 3])
                 ! Define the stiffness matrix
                 k = reshape([k1 + k2, -k2, 0.0d0, -k2, k2 + k3, -k3, 0.0d0, -k3, k3 + k4], &
                 ! Compute the eigenvalues and eigenvectors.
                call eigen(k, m, vals, vecs = modeshapes)
                 ! Compute the natural frequency values, and return them with units of Hz.
                  ! Notice, all eigenvalues and eigenvectors are real for this example.
                 natfreq = sqrt(real(vals)) / (2.0d0 * pi)
                 ! Display the natural frequency and mode shape values. Notice, the eigen
               ! routine does not necessarily sort the values. print ' (A)', "Modal Information (Not Sorted):"
                do i = 1, size(natfreq)
print '(AIOAF8.4A)', "Mode ", i, ": (", natfreq(i), " Hz)"
                                 print '(F10.3)', (real(modeShapes(j,i)), j = 1, size(natfreq))
                 end do
end program
```

#### **Notes**

This routine utilizes the LAPACK routine DGGEV.

Definition at line 465 of file linalg\_eigen.f90.

4.5.2.3 subroutine linalg\_eigen::eigen\_symm ( logical, intent(in) *vecs*, real(dp), dimension(:,:), intent(inout) *a*, real(dp), dimension(:), intent(out), optional, pointer *work*, integer(i32), intent(out), optional *olwork*, class(errors), intent(inout), optional, target *err* ) [private]

Computes the eigenvalues, and optionally the eigenvectors of a real, symmetric matrix.

in	vecs	Set to true to compute the eigenvectors as well as the eigenvalues; else, set to false to
		just compute the eigenvalues.

#### **Parameters**

in,out	а	On input, the N-by-N symmetric matrix on which to operate. On output, and if vecs is set to true, the matrix will contain the eigenvectors (one per column) corresponding to each eigenvalue in vals. If vecs is set to false, the lower triangular portion of the matrix is overwritten.
out	vals	An N-element array that will contain the eigenvalues sorted into ascending order.
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.
out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for $work$ , and returns without performing any actual calculations.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.</li> </ul>
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>
		LA_CONVERGENCE_ERROR: Occurs if the algorithm failed to converge.

#### **Notes**

This routine utilizes the LAPACK routine DSYEV.

Definition at line 68 of file linalg\_eigen.f90.

# 4.6 linalg\_factor Module Reference

# linalg\_factor

# **Data Types**

· interface form\_lu

Extracts the L and U matrices from the condensed [L\U] storage format used by the lu\_factor.

interface form\_qr

Forms the full M-by-M orthogonal matrix Q from the elementary reflectors returned by the base QR factorization algorithm.

• interface mult\_qr

Multiplies a general matrix by the orthogonal matrix Q from a QR factorization.

· interface mult\_rz

Multiplies a general matrix by the orthogonal matrix Z from an RZ factorization.

• interface qr\_factor

Computes the QR factorization of an M-by-N matrix.

#### **Functions/Subroutines**

• subroutine, public lu\_factor (a, ipvt, err)

Computes the LU factorization of an M-by-N matrix.

subroutine form\_lu\_all (lu, ipvt, u, p, err)

Extracts the L, U, and P matrices from the output of the lu\_factor routine.

• subroutine form\_lu\_only (lu, u, err)

Extracts the L, and U matrices from the output of the lu\_factor routine.

subroutine qr\_factor\_no\_pivot (a, tau, work, olwork, err)

Computes the QR factorization of an M-by-N matrix without pivoting.

• subroutine qr\_factor\_pivot (a, tau, jpvt, work, olwork, err)

Computes the QR factorization of an M-by-N matrix with column pivoting such that A \* P = Q \* R.

subroutine form\_qr\_no\_pivot (r, tau, q, work, olwork, err)

Forms the full M-by-M orthogonal matrix Q from the elementary reflectors returned by the base QR factorization algorithm.

• subroutine form\_qr\_pivot (r, tau, pvt, q, p, work, olwork, err)

Forms the full M-by-M orthogonal matrix Q from the elementary reflectors returned by the base QR factorization algorithm.

subroutine mult\_qr\_mtx (Iside, trans, a, tau, c, work, olwork, err)

Multiplies a general matrix by the orthogonal matrix Q from a QR factorization such that: C = op(Q) \* C, or C = C \* op(Q).

• subroutine mult gr vec (trans, a, tau, c, work, olwork, err)

Multiplies a vector by the orthogonal matrix Q from a QR factorization such that: C = op(Q) \* C.

• subroutine, public qr\_rank1\_update (q, r, u, v, work, err)

Computes the rank 1 update to an M-by-N QR factored matrix A (M >= N) where A = Q \* R, and A1 = A + U \* V \*\* T such that A1 = Q1 \* R1.

subroutine, public cholesky\_factor (a, upper, err)

Computes the Cholesky factorization of a symmetric, positive definite matrix.

subroutine, public cholesky\_rank1\_update (r, u, work, err)

Computes the rank 1 update to a Cholesky factored matrix (upper triangular).

subroutine, public cholesky\_rank1\_downdate (r, u, work, err)

Computes the rank 1 downdate to a Cholesky factored matrix (upper triangular).

• subroutine, public rz factor (a, tau, work, olwork, err)

Factors an upper trapezoidal matrix by means of orthogonal transformations such that  $A = R * Z = (R \ 0) * Z$ . Z is an orthogonal matrix of dimension N-by-N, and R is an M-by-M upper triangular matrix.

subroutine mult\_rz\_mtx (Iside, trans, I, a, tau, c, work, olwork, err)

Multiplies a general matrix by the orthogonal matrix Z from an RZ factorization such that: C = op(Z) \* C, or C = C \* op(Z).

• subroutine mult\_rz\_vec (trans, I, a, tau, c, work, olwork, err)

Multiplies a vector by the orthogonal matrix Z from an RZ factorization such that: C = op(Z) \* C.

• subroutine, public svd (a, s, u, vt, work, olwork, err)

Computes the singular value decomposition of a matrix A. The SVD is defined as: A = U \* S \* V \* \* T, where U is an M-by-M orthogonal matrix, S is an M-by-N diagonal matrix, and V is an N-by-N orthogonal matrix.

# 4.6.1 Detailed Description

# linalg\_factor

# **Purpose**

Provides a set of matrix factorization routines.

# 4.6.2 Function/Subroutine Documentation

4.6.2.1 subroutine, public linalg\_factor::cholesky\_factor ( real(dp), dimension(:,:), intent(inout) *a,* logical, intent(in), optional *upper,* class(errors), intent(inout), optional, target *err* )

Computes the Cholesky factorization of a symmetric, positive definite matrix.

#### **Parameters**

in,out	а	On input, the N-by-N matrix to factor. On output, the factored matrix is returned in either the upper or lower triangular portion of the matrix, dependent upon the value of upper.
in	upper	An optional input that, if specified, provides control over whether the factorization is computed as $A = U**T*U$ (set to true), or as $A = L*L**T$ (set to false). The default value is true such that $A = U**T*U$ .
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if a is not square.  • LA_MATRIX_FORMAT_ERROR: Occurs if a is not positive definite.

# Usage

To solve a system of N equations of N unknowns using Cholesky factorization, the following code will suffice. Notice, the system of equations must be positive definite.

```
! Solve the system: A*X = B, where A is an N-by-N matrix, and B and X are
! N-by-NRHS in size.
! Variables
real(dp), dimension(n, n) :: a
real(dp), dimension(n, nrhs) :: b
logical :: upper
! Initialize A and B...
! Specify that we're using the upper portion of A (remember positive
! definite matrices are symmetric)
upper = .true.
! Compute the factorization of A.
call cholesky_factor(a, upper)
! Solve A*X = B for X - Note: X overwrites B.
call solve_cholesky(upper, a, b)
```

#### **Notes**

This routine utilizes the LAPACK routine DPOTRF.

Definition at line 1308 of file linalg factor.f90.

4.6.2.2 subroutine, public linalg\_factor::cholesky\_rank1\_downdate ( real(dp), dimension(:,:), intent(inout) *r*, real(dp), dimension(:), intent(inout) *u*, real(dp), dimension(:), intent(out), optional, target *work*, class(errors), intent(inout), optional, target *err* )

Computes the rank 1 downdate to a Cholesky factored matrix (upper triangular).

in,out	r	On input, the N-by-N upper triangular matrix R. On output, the updated matrix R1.
in,out	и	On input, the N-element update vector U. On output, the rotation sines used to transform R to R1.
out	work	An optional argument that if supplied prevents local memory allocation. If provided, the array must have at least N elements. Additionally, this workspace array is used to contain the rotation cosines used to transform R to R1.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if any of the input array sizes are incorrect.
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>
		<ul> <li>LA_MATRIX_FORMAT_ERROR: Occurs if the downdated matrix is not positive definite.</li> </ul>
		LA_SINGULAR_MATRIX_ERROR: Occurs if r is singular.

#### Notes

This routine utilizes the QRUPDATE routine DCH1DN.

# See Also

#### Source

Definition at line 1494 of file linalg\_factor.f90.

4.6.2.3 subroutine, public linalg\_factor::cholesky\_rank1\_update ( real(dp), dimension(:,:), intent(inout) *r*, real(dp), dimension(:), intent(inout) *u*, real(dp), dimension(:), intent(out), optional, target *work*, class(errors), intent(inout), optional, target *err* )

Computes the rank 1 update to a Cholesky factored matrix (upper triangular).

in,out	r	On input, the N-by-N upper triangular matrix R. On output, the updated matrix R1.
in,out	и	On input, the N-element update vector U. On output, the rotation sines used to transform R
		to R1.
out	work	An optional argument that if supplied prevents local memory allocation. If provided, the
		array must have at least N elements. Additionally, this workspace array is used to contain
		the rotation cosines used to transform R to R1.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if any of the input array sizes are incorrect.
		LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.

#### Notes

This routine utilizes the QRUPDATE routine DCH1UP.

#### See Also

Source

Definition at line 1400 of file linalg\_factor.f90.

4.6.2.4 subroutine linalg\_factor::form\_lu\_all ( real(dp), dimension(:,:), intent(inout) *lu*, integer(i32), dimension(:), intent(in) *ipvt*, real(dp), dimension(:,:), intent(out) *u*, real(dp), dimension(:,:), intent(out) *p*, class(errors), intent(inout), optional, target *err* ) [private]

Extracts the L, U, and P matrices from the output of the lu factor routine.

#### **Parameters**

in, out	lu	On input, the N-by-N matrix as output by lu_factor. On output, the N-by-N lower triangular
		matrix L.
in	ipvt	The N-element pivot array as output by lu_factor.
out	и	An N-by-N matrix where the U matrix will be written.
out	р	An N-by-N matrix where the row permutation matrix will be written.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if any of the input array sizes are incorrect.

# Remarks

This routine allows extraction of the actual "L", "U", and "P" matrices of the decomposition. To use these matrices to solve the system A\*X = B, the following approach is used.

- 1. First, solve the linear system: L\*Y = P\*B for Y.
- 2. Second, solve the linear system: U\*X = Y for X.

Notice, as both L and U are triangular in structure, the above equations can be solved by forward and backward substitution.

#### See Also

- Wikipedia
- Wolfram MathWorld

Definition at line 211 of file linalg\_factor.f90.

4.6.2.5 subroutine linalg\_factor::form\_lu\_only ( real(dp), dimension(:,:), intent(inout) *lu*, real(dp), dimension(:,:), intent(out) *u*, class(errors), intent(inout), optional, target *err* ) [private]

Extracts the L, and U matrices from the output of the lu\_factor routine.

in,out	lu	On input, the N-by-N matrix as output by lu_factor. On output, the N-by-N lower triangular
		matrix L.
out	и	An N-by-N matrix where the U matrix will be written.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if any of the input array sizes are incorrect.

Definition at line 288 of file linalg\_factor.f90.

4.6.2.6 subroutine linalg\_factor::form\_qr\_no\_pivot ( real(dp), dimension(:,:), intent(inout) r, real(dp), dimension(:), intent(in) tau, real(dp), dimension(:,:), intent(out) q, real(dp), dimension(:), intent(out), optional, target work, integer(i32), intent(out), optional olwork, class(errors), intent(inout), optional, target err ) [private]

Forms the full M-by-M orthogonal matrix Q from the elementary reflectors returned by the base QR factorization algorithm.

#### **Parameters**

in,out	r	On input, an M-by-N matrix where the elements below the diagonal contain the elementary reflectors generated from the QR factorization. On and above the diagonal, the matrix contains the matrix R. On output, the elements below the diagonal are zeroed such that the remaining matrix is simply the M-by-N matrix R.
in	tau	A MIN(M, N)-element array containing the scalar factors of each elementary reflector defined in r.
out	q	An M-by-M matrix where the full orthogonal matrix Q will be written. In the event that $M > N$ , Q may be supplied as M-by-N, and therefore only return the useful submatrix Q1 (Q = [Q1, Q2]) as the factorization can be written as Q * R = [Q1, Q2] * [R1; 0].
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.
out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for work, and returns without performing any actual calculations.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized
		appropriately.  • LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and
		there is insufficient memory available.

# Notes

This routine utilizes the LAPACK routine DORGQR.

Definition at line 694 of file linalg\_factor.f90.

4.6.2.7 subroutine linalg\_factor::form\_qr\_pivot ( real(dp), dimension(:,:), intent(inout) r, real(dp), dimension(:), intent(in) tau, integer(i32), dimension(:), intent(in) pvt, real(dp), dimension(:,:), intent(out) q, real(dp), dimension(:,:), intent(out) p, real(dp), dimension(:), intent(out), optional, target work, integer(i32), intent(out), optional olwork, class(errors), intent(inout), optional, target err ) [private]

Forms the full M-by-M orthogonal matrix Q from the elementary reflectors returned by the base QR factorization algorithm.

# **Parameters**

in,out	r	On input, an M-by-N matrix where the elements below the diagonal contain the elementary reflectors generated from the QR factorization. On and above the diagonal, the matrix contains the matrix R. On output, the elements below the diagonal are zeroed such that the remaining matrix is simply the M-by-N matrix R.
in	tau	A MIN(M, N)-element array containing the scalar factors of each elementary reflector defined in r.
in	pvt	An N-element column pivot array as returned by the QR factorization.
out	q	An M-by-M matrix where the full orthogonal matrix Q will be written. In the event that $M > N$ , Q may be supplied as M-by-N, and therefore only return the useful submatrix Q1 (Q = [Q1, Q2]) as the factorization can be written as $Q * R = [Q1, Q2] * [R1; 0]$ .
out	р	An N-by-N matrix where the pivot matrix will be written.
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.
out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for $work$ , and returns without performing any actual calculations.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.</li> </ul>
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>

# Notes

This routine utilizes the LAPACK routine DORGQR.

Definition at line 825 of file linalg\_factor.f90.

4.6.2.8 subroutine, public linalg\_factor::lu\_factor ( real(dp), dimension(:,:), intent(inout) a, integer(i32), dimension(:), intent(out) ipvt, class(errors), intent(inout), optional, target err )

Computes the LU factorization of an M-by-N matrix.

in,out	а	On input, the M-by-N matrix on which to operate. On output, the LU factored matrix in the
		form [L\U] where the unit diagonal elements of L are not stored.

out	ipvt	An MIN(M, N)-element array used to track row-pivot operations. The array stored pivot information such that row I is interchanged with row IPVT(I).
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if ipvt is not sized appropriately.  • LA_SINGULAR_MATRIX_ERROR: Occurs as a warning if a is found to be singular.

#### Usage

To solve a system of N equations of N unknowns using LU factorization, the following code will suffice.

```
! Solve the system: A*X = B, where A is an N-by-N matrix, and B and X are
! N-by-NRHS in size.
! Variables
real(dp), dimension(n, n) :: a
real(dp), dimension(n, nrhs) :: b
! Define the array used to track row pivots.
integer(i32), dimension(n) :: pvt
! Initialize A and B...
! Compute the LU factorization of A. On output, A contains [L\U].
call lu_factor(a, pvt)
! Solve A*X = B for X - Note: X overwrites B.
call solve_lu(a, pvt, b)
```

#### Notes

This routine utilizes the LAPACK routine DGETRF.

# See Also

- Wikipedia
- Wolfram MathWorld

Definition at line 127 of file linalg\_factor.f90.

4.6.2.9 subroutine linalg\_factor::mult\_qr\_mtx ( logical, intent(in) *lside*, logical, intent(in) *trans*, real(dp), dimension(:,:), intent(inout) *a*, real(dp), dimension(:), intent(in) *tau*, real(dp), dimension(:,:), intent(inout) *c*, real(dp), dimension(:), intent(out), optional, target *work*, integer(i32), intent(out), optional *olwork*, class(errors), intent(inout), optional, target *err* ) [private]

Multiplies a general matrix by the orthogonal matrix Q from a QR factorization such that: C = op(Q) \* C, or C = C \* op(Q).

in	Iside	Set to true to apply Q or Q**T from the left; else, set to false to apply Q or Q**T from the right.
in	trans	Set to true to apply Q**T; else, set to false.

#### **Parameters**

in	а	On input, an LDA-by-K matrix containing the elementary reflectors output from the QR factorization. If lside is set to true, LDA = M, and $M >= K >= 0$ ; else, if lside is set to false, LDA = N, and $N >= K >= 0$ . Notice, the contents of this matrix are restored on exit.
in	tau	A K-element array containing the scalar factors of each elementary reflector defined in a.
in,out	С	On input, the M-by-N matrix C. On output, the product of the orthogonal matrix Q and the original matrix C.
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.
out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for work, and returns without performing any actual calculations.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>

# Notes

This routine utilizes the LAPACK routine DORMQR.

Definition at line 927 of file linalg\_factor.f90.

4.6.2.10 subroutine linalg\_factor::mult\_qr\_vec ( logical, intent(in) *trans*, real(dp), dimension(:,:), intent(inout) *a*, real(dp), dimension(:), intent(in) *tau*, real(dp), dimension(:), intent(inout) *c*, real(dp), dimension(:), intent(out), optional, target work, integer(i32), intent(out), optional olwork, class(errors), intent(inout), optional, target err ) [private]

Multiplies a vector by the orthogonal matrix Q from a QR factorization such that: C = op(Q) \* C.

in	trans	Set to true to apply Q**T; else, set to false.
in	а	On input, an M-by-K matrix containing the elementary reflectors output from the QR factorization. Notice, the contents of this matrix are restored on exit.
in	tau	A K-element array containing the scalar factors of each elementary reflector defined in a.
in,out	С	On input, the M-element vector C. On output, the product of the orthogonal matrix Q and the original vector C.
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.
out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for work, and returns without performing any actual calculations.

out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.</li> </ul>
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>

#### Notes

This routine is based upon the LAPACK routine DORM2R.

Definition at line 1054 of file linalg\_factor.f90.

4.6.2.11 subroutine linalg\_factor::mult\_rz\_mtx ( logical, intent(in) *lside,* logical, intent(in) *trans,* integer(i32), intent(in) *l,* real(dp), dimension(:,:), intent(inout) *a,* real(dp), dimension(:), intent(inout) *tau,* real(dp), dimension(:,:), intent(inout) *c,* real(dp), dimension(:), intent(out), optional, target *work,* integer(i32), intent(out), optional *olwork,* class(errors), intent(inout), optional, target *err* ) [private]

Multiplies a general matrix by the orthogonal matrix Z from an RZ factorization such that: C = op(Z) \* C, or C = C \* op(Z).

in	Iside	Set to true to apply Z or Z**T from the left; else, set to false to apply Z or Z**T from the right.
in	trans	Set to true to apply Z**T; else, set to false.
in	1	The number of columns in matrix a containing the meaningful part of the Householder vectors. If $lside$ is true, $M >= L >= 0$ ; else, if $lside$ is false, $N >= L >= 0$ .
in,out	а	On input the K-by-LTA matrix Z, where LTA = M if lside is true; else, LTA = N if lside is false. The I-th row must contain the Householder vector in the last k rows. Notice, the contents of this matrix are restored on exit.
in	tau	A K-element array containing the scalar factors of the elementary reflectors, where $M >= K >= 0$ if lside is true; else, $N >= K >= 0$ if lside is false.
in,out	С	On input, the M-by-N matrix C. On output, the product of the orthogonal matrix Z and the original matrix C.
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.
out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for work, and returns without performing any actual calculations.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.</li> </ul>
		LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and
enerated by Do	oxygen	there is insufficient memory available.

#### Notes

This routine utilizes the LAPACK routine DORMRZ.

Definition at line 1745 of file linalg\_factor.f90.

4.6.2.12 subroutine linalg\_factor::mult\_rz\_vec ( logical, intent(in) trans, integer(i32), intent(in) I, real(dp), dimension(:,:), intent(inout) a, real(dp), dimension(:), intent(in) tau, real(dp), dimension(:), intent(inout) c, real(dp), dimension(:), intent(out), optional, target work, integer(i32), intent(out), optional olwork, class(errors), intent(inout), optional, target err ) [private]

Multiplies a vector by the orthogonal matrix Z from an RZ factorization such that: C = op(Z) \* C.

#### **Parameters**

in	trans	Set to true to apply $Z**T$ ; else, set to false.
in	1	The number of columns in matrix a containing the meaningful part of the Householder vectors. If $lside$ is true, $M >= L >= 0$ ; else, if $lside$ is false, $N >= L >= 0$ .
in,out	а	On input the K-by-LTA matrix Z, where LTA = M if lside is true; else, LTA = N if lside is false. The I-th row must contain the Householder vector in the last k rows. Notice, the contents of this matrix are restored on exit.
in	tau	A K-element array containing the scalar factors of the elementary reflectors, where $M \ge K \ge 0$ if lside is true; else, $N \ge K \ge 0$ if lside is false.
in,out	С	On input, the M-element array C. On output, the product of the orthogonal matrix Z and the original array C.
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.
out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for work, and returns without performing any actual calculations.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.  • LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.

# Notes

This routine utilizes the LAPACK routine DORMRZ.

Definition at line 1887 of file linalg\_factor.f90.

4.6.2.13 subroutine linalg\_factor::qr\_factor\_no\_pivot ( real(dp), dimension(:,:), intent(inout) a, real(dp), dimension(:), intent(out) tau, real(dp), dimension(:), intent(out), optional, target work, integer(i32), intent(out), optional olwork, class(errors), intent(inout), optional, target err ) [private]

Computes the QR factorization of an M-by-N matrix without pivoting.

in,out	а	On input, the M-by-N matrix to factor. On output, the elements on and above the diagonal contain the MIN(M, N)-by-N upper trapezoidal matrix R (R is upper triangular if $M \ge N$ ). The elements below the diagonal, along with the array tau, represent the orthogonal matrix Q as a product of elementary reflectors.
out	tau	A MIN(M, N)-element array used to store the scalar factors of the elementary reflectors.
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.
out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for work, and returns without performing any actual calculations.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if tau or work are not sized appropriately.  • LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.

#### Remarks

QR factorization without pivoting is best suited to solving an overdetermined system in least-squares terms, or to solve a normally defined system. To solve an underdetermined system, it is recommended to use either LQ factorization, or a column-pivoting based QR factorization.

### Usage

To solve a system of M equations of N unknowns using QR factorization, the following code will suffice assuming M >= N.

```
! Solve the system: A*X = B in a least-squares sense, where A is an
! M-by-N matrix, B is an M-by-NRHS matrix, and X is an N-by-NRHS matrix.
real(dp), dimension(m, n) :: a
\text{real}(\text{dp}), \text{dimension}(\text{m, nrhs}) :: b, \text{qtb}
real(dp), dimension(n, nrhs) :: x
real(dp), dimension(n) :: tau
real(dp), dimension(m, m) :: q
! Initialize A and B...
! Compute the QR factorization. We're intentionally not forming the full
! Q matrix, but instead storing it in terms of its elementary reflector
! components in the sub-diagonal portions of A, and the corresponding
! scalar factors in TAU. Additionally, we'll let the algorithm allocate
 it's own workspace array; therefore, the call to factor A is:
call qr_factor(a, tau)
! Solve A \star X = B for X. The first N rows of B are used to store X.
call solve_qr(a, tau, b)
! Also note, we could form {\tt Q} and {\tt R} explicitly. Then solution of the
! system of equations can be found. First we form {\tt Q} and {\tt R}\text{.}
call form_qr(a, tau, q) ! Forms Q, and R is stored in A
 Since we now have Q and R, we seek a solution to the equation:
 Q*R*X = B, but Q is an orthogonal matrix (i.e. Q**T = inv(Q)). Then: R*X = Q**T * B, and R is upper triangular; therefore, back
  substitution will suffice for a solution procedure.
! Next, compute Q**T*B, and store in QTB.
call mtx_mult(.true., .false., 1.0d0, q, b, 0.0d0, qtb)
! Copy the first N rows of Q**T*B into X for the solution process.
```

```
! Notice, only the first N rows are needed as rows N+1:M are all zero in ! matrix R. x = \text{qtb}(1:n, nrhs) ! Compute the solution and store in X call solve_triangular_system(.true., .true., .false., .true., 1.0d0, & a(1:n,1:n), x)
```

#### Notes

This routine utilizes the LAPACK routine DGEQRF.

Definition at line 425 of file linalg\_factor.f90.

4.6.2.14 subroutine linalg\_factor::qr\_factor\_pivot ( real(dp), dimension(:,:), intent(inout) a, real(dp), dimension(:), intent(out) tau, integer(i32), dimension(:), intent(inout) jpvt, real(dp), dimension(:), intent(out), optional, target work, integer(i32), intent(out), optional olwork, class(errors), intent(inout), optional, target err ) [private]

Computes the QR factorization of an M-by-N matrix with column pivoting such that A \* P = Q \* R.

#### **Parameters**

in,out	а	On input, the M-by-N matrix to factor. On output, the elements on and above the diagonal contain the MIN(M, N)-by-N upper trapezoidal matrix R (R is upper triangular if $M \ge N$ ). The elements below the diagonal, along with the array tau, represent the orthogonal matrix Q as a product of elementary reflectors.
out	tau	A MIN(M, N)-element array used to store the scalar factors of the elementary reflectors.
in,out	jpvt	On input, an N-element array that if $JPVT(I)$ .ne. 0, the I-th column of A is permuted to the front of A $*$ P; if $JPVT(I) = 0$ , the I-th column of A is a free column. On output, if $JPVT(I) = K$ , then the I-th column of A $*$ P was the K-th column of A.
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.
out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for work, and returns without performing any actual calculations.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized
		<ul> <li>appropriately.</li> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>

#### Usage

To solve a system of M equations of N unknowns using QR factorization, the following code will suffice for any M and N.

```
! Solve the least-squares (M >= N), or the underdetermined (M < N) ! problem A*X = B, where A is an M-by-N matrix, B is an M-by-NRHS matrix, ! and X is an N-by-NRHS matrix. In the underdetermined case, or the ! case where the rank of matrix A is less than N, the solution obtained ! contains the fewest possible non-zero entries.
```

```
real(dp), dimension(m, n) :: a
real(dp), dimension(n, nrhs) :: b
real(dp), dimension(k) :: tau ! k = min(m, n)
\texttt{real(dp), dimension(m, m) :: } \texttt{q}
real(dp), dimension(n, n) :: p
integer(i32), dimension(n) :: pvt
! Initialize A and B...
! Allow all columns to be free.
! Compute the QR factorization. We're intentionally not forming the full
! Q matrix, but instead storing it in terms of its elementary reflector
! components in the sub-diagonal portions of {\tt A}, and the corresponding
! scalar factors in TAU. Additionally, we'll let the algorithm allocate
! it's own workspace array; therefore, the call to factor \mbox{A} is:
call qr_factor(a, tau, pvt)
! Solve A\star X = B for X. If M > N, the first N rows of B are used to store
! X. If M < N, the input matrix B must be N-by-NRHS, and only the first
! M rows are used for the actual matrix B. The remaining N\!-\!M rows
!\ \mbox{can} contain whatever as they are not referenced until they are
! overwritten by the N-by-NRHS solution matrix {\tt X.}
call solve_qr(a, tau, pvt, b)
! Notice, if the explicit {\tt Q} matrix from the factorization is desired,
! the form_qr routine works similarly as in the no-pivot case;
! however, the permutation matrix P is also constructed. The call would
! be as follows. Also, as with the no-pivot algorithm, the matrix \ensuremath{\mathbf{R}} is
! stored in matrix A.
call form_qr(a, tau, pvt, q, p)
! Solution can proceed as per typical, but with a full {\tt Q} matrix. Also
! note, the problem is of the form: A \star P = Q \star R. Solution is straight! forward, as with the no-pivot case; however, if M < N, then R is upper
 trapezoidal, and must be appropriately partitioned to solve. The rank
! of matrix r should be considered when applying the partition.
```

#### Notes

This routine utilizes the LAPACK routine DGEQP3.

Definition at line 579 of file linalg\_factor.f90.

4.6.2.15 subroutine, public linalg\_factor::qr\_rank1\_update ( real(dp), dimension(:,:), intent(inout) q, real(dp), dimension(:,:), intent(inout) r, real(dp), dimension(:), intent(inout) u, real(dp), dimension(:), intent(inout) v, real(dp), dimension(:), intent(out), optional, target work, class(errors), intent(inout), optional, target err )

Computes the rank 1 update to an M-by-N QR factored matrix A (M >= N) where A = Q \* R, and A1 = A + U \* V\*\*T such that A1 = Q1 \* R1.

in,out	q	On input, the original M-by-K orthogonal matrix Q. On output, the updated matrix Q1.
in,out	r	On input, the M-by-N matrix R. On output, the updated matrix R1.
in, out	и	On input, the M-element U update vector. On output, the original content of the array is
		overwritten.
in,out	V	On input, the N-element V update vector. On output, the original content of the array is
		overwritten.
out	work	An optional argument that if supplied prevents local memory allocation. If provided, the
		array must have at least 2*K elements.

#### **Parameters**

out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.</li> </ul>
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>

#### Remarks

```
Notice, K must either be equal to M, or to N. In the event that K = N, only the submatrix Qa is updated. This is appropriate as the QR factorization for an overdetermined system can be written as follows: A = Q * R = [Qa, Qb] * [Ra]
```

Note: Ra is upper triangular of dimension N-by-N.

#### **Notes**

This routine utilizes the QRUPDATE routine DQR1UP.

#### See Also

Source

Definition at line 1180 of file linalg\_factor.f90.

4.6.2.16 subroutine, public linalg\_factor::rz\_factor ( real(dp), dimension(:,:), intent(inout) *a,* real(dp), dimension(:), intent(out) *tau,* real(dp), dimension(:), intent(out), optional, target *work,* integer(i32), intent(out), optional *olwork,* class(errors), intent(inout), optional, target *err* )

Factors an upper trapezoidal matrix by means of orthogonal transformations such that  $A = R * Z = (R \ 0) * Z$ . Z is an orthogonal matrix of dimension N-by-N, and R is an M-by-M upper triangular matrix.

in,out	а	On input, the M-by-N upper trapezoidal matrix to factor. On output, the leading M-by-M upper triangular part of the matrix contains the upper triangular matrix R, and elements N-L+1 to N of the first M rows of A, with the array tau, represent the orthogonal matrix Z as a product of M elementary reflectors.
out	tau	An M-element array used to store the scalar factors of the elementary reflectors.
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.
out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for $\mathtt{work}$ , and returns without performing any actual calculations.

out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.</li> </ul>
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>

#### **Further Details**

```
The factorization is obtained by Householder's method. The kth transformation matrix, Z(k), which is used to introduce zeros into the (m-k+1)th row of A, is given in the form Z(k) = (I \quad 0 \quad ), \\  \quad (0 \quad T(k)) where T(k) = I - tau*u(k)*u(k)**T, \quad u(k) = (I \quad ), \\  \quad (0 \quad ) \quad (z(k)) tau is a scalar and z(k) is an I element vector. tau and z(k) are chosen to annihilate the elements of the kth row of A2. The scalar tau is returned in the kth element of TAU and the vector u(k) in the kth row of A2, such that the elements of z(k) are in a(k, l+1), \ldots, a(k, n). The elements of R are returned in the upper triangular part of A1. Z is given by Z = Z(1)*Z(2)*\ldots*Z(m).
```

# Notes

This routine is based upon the LAPACK routine DTZRZF.

#### See Also

• LAPACK Users Manual

Definition at line 1633 of file linalg\_factor.f90.

4.6.2.17 subroutine, public linalg\_factor::svd ( real(dp), dimension(:,:), intent(inout) a, real(dp), dimension(:), intent(out) s, real(dp), dimension(:,:), intent(out), optional u, real(dp), dimension(:,:), intent(out), optional vt, real(dp), dimension(:), intent(out), optional, target work, integer(i32), intent(out), optional olwork, class(errors), intent(inout), optional, target err)

Computes the singular value decomposition of a matrix A. The SVD is defined as: A = U \* S \* V \* \* T, where U is an M-by-M orthogonal matrix, S is an M-by-N diagonal matrix, and V is an N-by-N orthogonal matrix.

#### **Parameters**

in, out	а	On input, the M-by-N matrix to factor. The matrix is overwritten on output.
out	s	A MIN(M, N)-element array containing the singular values of a sorted in descending order.
out	и	An optional argument, that if supplied, is used to contain the orthogonal matrix U from the decomposition. The matrix U contains the left singular vectors, and can be either M-by-M (all left singular vectors are computed), or M-by-MIN(M,N) (only the first MIN(M, N) left singular vectors are computed).
out	vt	An optional argument, that if supplied, is used to contain the transpose of the N-by-N orthogonal matrix V. The matrix V contains the right singular vectors.
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.
out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for work, and returns without performing any actual calculations.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.</li> </ul>
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>
		<ul> <li>LA_CONVERGENCE_ERROR: Occurs as a warning if the QR iteration process could not converge to a zero value.</li> </ul>

# Usage

```
! Decompose matrix the M-by-N matrix A such that A = U * S * V**T with
! M >= N.

! Variables
real(dp), dimension(m, n) :: a
real(dp), dimension(m, m) :: u
real(dp), dimension(n) :: s

! Initialize A...

! Compute the SVD of A. On output, S contains the MIN(M,N) singular
! values of A in descending order, U contains the left singular vectors
! (one per column), and VT contains the right singular vectors (one per ! row).
call svd(a, s, u, vt)

! Note: If M > N, then we can make U M-by-N, and compute the N
! left singular vectors of A, as there are at most N singular values
! of A. Also, if M < N, then there are at most M singular values of A,
! and as such, the length of the array s should be m.</pre>
```

### Notes

This routine utilizes the LAPACK routine DGESVD.

# See Also

- Wikipedia
- Wolfram MathWorld

Definition at line 2046 of file linalg\_factor.f90.

# 4.7 linalg\_solve Module Reference

# linalg\_solve

### **Data Types**

· interface solve cholesky

Solves a system of Cholesky factored equations.

• interface solve\_least\_squares

Solves the overdetermined or underdetermined system (A\*X = B) of M equations of N unknowns.

· interface solve least squares full

Solves the overdetermined or underdetermined system (A\*X = B) of M equations of N unknowns, but uses a full orthogonal factorization of the system.

· interface solve least squares svd

Solves the overdetermined or underdetermined system (A\*X = B) of M equations of N unknowns using a singular value decomposition of matrix A.

· interface solve lu

Solves a system of LU-factored equations.

· interface solve gr

Solves a system of M QR-factored equations of N unknowns.

• interface solve\_triangular\_system

Solves a triangular system of equations.

# **Functions/Subroutines**

• subroutine solve tri mtx (Iside, upper, trans, nounit, alpha, a, b, err)

Solves one of the matrix equations: op(A) \* X = alpha \* B, or X \* op(A) = alpha \* B, where A is a triangular matrix.

• subroutine solve\_tri\_vec (upper, trans, nounit, a, x, err)

Solves the system of equations: op(A) \* X = B, where A is a triangular matrix.

• subroutine solve\_lu\_mtx (a, ipvt, b, err)

Solves a system of LU-factored equations.

• subroutine solve\_lu\_vec (a, ipvt, b, err)

Solves a system of LU-factored equations.

• subroutine solve\_qr\_no\_pivot\_mtx (a, tau, b, work, olwork, err)

Solves a system of M QR-factored equations of N unknowns where M >= N.

• subroutine solve\_qr\_no\_pivot\_vec (a, tau, b, work, olwork, err)

Solves a system of M QR-factored equations of N unknowns where M >= N.

subroutine solve\_qr\_pivot\_mtx (a, tau, jpvt, b, work, olwork, err)

Solves a system of M QR-factored equations of N unknowns where the QR factorization made use of column pivoting.

subroutine solve\_qr\_pivot\_vec (a, tau, jpvt, b, work, olwork, err)

Solves a system of M QR-factored equations of N unknowns where the QR factorization made use of column pivoting.

subroutine solve\_cholesky\_mtx (upper, a, b, err)

Solves a system of Cholesky factored equations.

subroutine solve\_cholesky\_vec (upper, a, b, err)

Solves a system of Cholesky factored equations.

subroutine, public mtx\_inverse (a, iwork, work, olwork, err)

Computes the inverse of a square matrix.

• subroutine, public mtx\_pinverse (a, ainv, tol, work, olwork, err)

Computes the Moore-Penrose pseudo-inverse of a M-by-N matrix using the singular value decomposition of the matrix

subroutine solve\_least\_squares\_mtx (a, b, work, olwork, err)

Solves the overdetermined or underdetermined system (A\*X = B) of M equations of N unknowns using a QR or LQ factorization of the matrix A. Notice, it is assumed that matrix A has full rank.

• subroutine solve least squares vec (a, b, work, olwork, err)

Solves the overdetermined or underdetermined system (A\*X = B) of M equations of N unknowns using a QR or LQ factorization of the matrix A. Notice, it is assumed that matrix A has full rank.

• subroutine solve least squares mtx pvt (a, b, ipvt, arnk, work, olwork, err)

Solves the overdetermined or underdetermined system (A\*X = B) of M equations of N unknowns using a complete orthogonal factorization of matrix A.

• subroutine solve\_least\_squares\_vec\_pvt (a, b, ipvt, arnk, work, olwork, err)

Solves the overdetermined or underdetermined system (A\*X = B) of M equations of N unknowns using a complete orthogonal factorization of matrix A.

• subroutine solve\_least\_squares\_mtx\_svd (a, b, arnk, s, work, olwork, err)

Solves the overdetermined or underdetermined system (A\*X = B) of M equations of N unknowns using a singular value decomposition of matrix A.

• subroutine solve\_least\_squares\_vec\_svd (a, b, arnk, s, work, olwork, err)

Solves the overdetermined or underdetermined system (A\*X = B) of M equations of N unknowns using a singular value decomposition of matrix A.

# 4.7.1 Detailed Description

#### linalg solve

#### **Purpose**

Provides a set of routines for solving systems of linear equations.

### 4.7.2 Function/Subroutine Documentation

4.7.2.1 subroutine, public linalg\_solve::mtx\_inverse ( real(dp), dimension(:,:), intent(inout) *a*, integer(i32), dimension(:), intent(out), optional, target *iwork*, real(dp), dimension(:), intent(out), optional, target *work*, integer(i32), intent(out), optional *olwork*, class(errors), intent(inout), optional, target *err* )

Computes the inverse of a square matrix.

in,out	а	On input, the N-by-N matrix to invert. On output, the inverted matrix.
out	iwork	An optional N-element integer workspace array.
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.
out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for $work$ , and returns without performing any actual calculations.

out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if a is not square. Will also occur if incorrectly sized workspace arrays are provided.</li> </ul>
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>
		LA_SINGULAR_MATRIX_ERROR: Occurs if the input matrix is singular.

### Usage

```
! The following example illustrates how to solve a system of linear
! equations by matrix inversion. Notice, this is not a preferred
! solution technique (use LU factorization instead), but is merely a
! means of illustrating how to compute the inverse of a square matrix.
! Variables
real(dp), dimension(n, n) :: a
real(dp), dimension(n, nrhs) :: b, x
! Initialize A and B...
! Compute the inverse of A. The inverse will overwrite the original
! matrix.
call mtx_inverse(a)
! Solve A*X = B as X = inv(A) * B.
x = matmul(a, b)
```

# Notes

This routine utilizes the LAPACK routines DGETRF to perform an LU factorization of the matrix, and DGETRI to invert the LU factored matrix.

#### See Also

- Wikipedia
- Wolfram MathWorld

Definition at line 1240 of file linalg\_solve.f90.

4.7.2.2 subroutine, public linalg\_solve::mtx\_pinverse ( real(dp), dimension(:,:), intent(inout) *a,* real(dp), dimension(:,:), intent(out) *ainv,* real(dp), intent(in), optional *tol,* real(dp), dimension(:), intent(out), optional, target *work,* integer(i32), intent(out), optional *olwork,* class(errors), intent(inout), optional, target *err* )

Computes the Moore-Penrose pseudo-inverse of a M-by-N matrix using the singular value decomposition of the matrix.

in,out	а	On input, the M-by-N matrix to invert. The matrix is overwritten on output.
out	ainv	The N-by-M matrix where the pseudo-inverse of a will be written.

#### **Parameters**

in	tol	An optional input, that if supplied, overrides the default tolerance on singular values such that singular values less than this tolerance are forced to have a reciprocal of zero, as opposed to $1/S(I)$ . The default tolerance is: $MAX(M, N) * EPS * MAX(S)$ . If the supplied value is less than a value that causes an overflow, the tolerance reverts back to its default value, and the operation continues; however, a warning message is issued.
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.
out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for work, and returns without performing any actual calculations.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.</li> </ul>
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>
		<ul> <li>LA_CONVERGENCE_ERROR: Occurs as a warning if the QR iteration process could not converge to a zero value.</li> </ul>

# Usage

```
! Use the pseudo-inverse to obtain a least-squares solution to the
! overdetermined problem A*X = B, where A is an M-by-N matrix (M >= N),
! B is an M-by-NRHS matrix, and X is an N-by-NRHS matrix.
! Variables
real(dp), dimension(m, n) :: a
real(dp), dimension(m, m) :: ainv
real(dp), dimension(m, nrhs) :: b
real(dp), dimension(n, nrhs) :: x
! Initialize A, and B...
! Compute the pseudo-inverse of A. Let the subroutine allocate its
! own workspace array.
call mtx_pinverse(a, ainv)
! Compute X = AINV * B to obtain the solution.
x = matmul(ainv, b)
```

#### See Also

- Wikipedia
- Wolfram MathWorld
- MathWorks

Definition at line 1400 of file linalg\_solve.f90.

4.7.2.3 subroutine linalg\_solve::solve\_cholesky\_mtx ( logical, intent(in) *upper*, real(dp), dimension(:,:), intent(in) *a*, real(dp), dimension(:,:), intent(inout) *b*, class(errors), intent(inout), optional, target *err* ) [private]

Solves a system of Cholesky factored equations.

in	upper	Set to true if the original matrix A was factored such that $A = U**T*U$ ; else, set to false if
		the factorization of A was $A = L**T*L$ .
in	а	The N-by-N Cholesky factored matrix.
in,out	b	On input, the N-by-NRHS right-hand-side matrix B. On output, the solution matrix X.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		LA_ARRAY_SIZE_ERROR: Occurs if any of the input array sizes are incorrect.

#### Notes

This routine utilizes the LAPACK routine DPOTRS.

Definition at line 1073 of file linalg\_solve.f90.

4.7.2.4 subroutine linalg\_solve::solve\_cholesky\_vec ( logical, intent(in) *upper*, real(dp), dimension(:,:), intent(in) *a*, real(dp), dimension(:), intent(inout) *b*, class(errors), intent(inout), optional, target *err* ) [private]

Solves a system of Cholesky factored equations.

#### **Parameters**

in	upper	Set to true if the original matrix A was factored such that $A = U**T*U$ ; else, set to false if
		the factorization of A was $A = L**T*L$ .
in	а	The N-by-N Cholesky factored matrix.
in,out	b	On input, the N-element right-hand-side vector B. On output, the solution vector X.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		LA_ARRAY_SIZE_ERROR: Occurs if any of the input array sizes are incorrect.

#### Notes

This routine utilizes the LAPACK routine DPOTRS.

Definition at line 1139 of file linalg\_solve.f90.

4.7.2.5 subroutine linalg\_solve::solve\_least\_squares\_mtx ( real(dp), dimension(:,:), intent(inout) a, real(dp), dimension(:,:), intent(inout) b, real(dp), dimension(:), intent(out), optional, target work, integer(i32), intent(out), optional olwork, class(errors), intent(inout), optional, target err ) [private]

Solves the overdetermined or underdetermined system (A\*X = B) of M equations of N unknowns using a QR or LQ factorization of the matrix A. Notice, it is assumed that matrix A has full rank.

#### **Parameters**

in,out	а	On input, the M-by-N matrix A. On output, if $M \ge N$ , the QR factorization of A in the form as output by qr_factor; else, if $M < N$ , the LQ factorization of A.
in,out	b	If $M >= N$ , the M-by-NRHS matrix B. On output, the first N rows contain the N-by-NRHS solution matrix X. If $M < N$ , an N-by-NRHS matrix with the first M rows containing the matrix B. On output, the N-by-NRHS solution matrix X.
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.
out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for $work$ , and returns without performing any actual calculations.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.</li> </ul>
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>
		LA_INVALID_OPERATION_ERROR: Occurs if a is not of full rank.

# Notes

This routine utilizes the LAPACK routine DGELS.

Definition at line 1568 of file linalg\_solve.f90.

4.7.2.6 subroutine linalg\_solve::solve\_least\_squares\_mtx\_pvt ( real(dp), dimension(:,:), intent(inout) a, real(dp), dimension(:,:), intent(inout) b, integer(i32), dimension(:), intent(inout), optional, target ipvt, integer(i32), intent(out), optional arnk, real(dp), dimension(:), intent(out), optional, target work, integer(i32), intent(out), optional olwork, class(errors), intent(inout), optional, target err ) [private]

Solves the overdetermined or underdetermined system (A\*X = B) of M equations of N unknowns using a complete orthogonal factorization of matrix A.

in,out	а	On input, the M-by-N matrix A. On output, the matrix is overwritten by the details of its complete orthogonal factorization.
in,out	b	If $M >= N$ , the M-by-NRHS matrix B. On output, the first N rows contain the N-by-NRHS solution matrix X. If $M < N$ , an N-by-NRHS matrix with the first M rows containing the matrix B. On output, the N-by-NRHS solution matrix X.
out	ipvt	An optional input that on input, an N-element array that if $IPVT(I)$ .ne. 0, the I-th column of A is permuted to the front of A $*$ P; if $IPVT(I) = 0$ , the I-th column of A is a free column. On output, if $IPVT(I) = K$ , then the I-th column of A $*$ P was the K-th column of A. If not supplied, memory is allocated internally, and $IPVT$ is set to all zeros such that all columns are treated as free.
out	arnk	An optional output, that if provided, will return the rank of a.
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.

out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for work, and returns without performing any actual calculations.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>

# Notes

This routine utilizes the LAPACK routine DGELSY.

Definition at line 1786 of file linalg\_solve.f90.

4.7.2.7 subroutine linalg\_solve::solve\_least\_squares\_mtx\_svd ( real(dp), dimension(:,:), intent(inout) a, real(dp), dimension(:,:), intent(inout) b, integer(i32), intent(out), optional arnk, real(dp), dimension(:), intent(out) s, real(dp), dimension(:), intent(out), optional, target work, integer(i32), intent(out), optional olwork, class(errors), intent(inout), optional, target err ) [private]

Solves the overdetermined or underdetermined system (A\*X = B) of M equations of N unknowns using a singular value decomposition of matrix A.

in,out	а	On input, the M-by-N matrix A. On output, the matrix is overwritten by the details of its complete orthogonal factorization.
in,out	b	If $M >= N$ , the M-by-NRHS matrix B. On output, the first N rows contain the N-by-NRHS solution matrix X. If $M < N$ , an N-by-NRHS matrix with the first M rows containing the matrix B. On output, the N-by-NRHS solution matrix X.
out	arnk	An optional output, that if provided, will return the rank of a.
out	s	A MIN(M, N)-element array that on output contains the singular values of $a$ in descending order. Notice, the condition number of $a$ can be determined by S(1) / S(MIN(M, N)).
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.
out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for work, and returns without performing any actual calculations.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.</li> </ul>
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>
Generated by Do	oxygen	LA_CONVERGENCE_ERROR: Occurs as a warning if the QR iteration process could not converge to a zero value.

#### Notes

This routine utilizes the LAPACK routine DGELSS.

Definition at line 2076 of file linalg solve.f90.

4.7.2.8 subroutine linalg\_solve::solve\_least\_squares\_vec ( real(dp), dimension(:,:), intent(inout) a, real(dp), dimension(:), intent(inout) b, real(dp), dimension(:), intent(out), optional, target work, integer(i32), intent(out), optional olwork, class(errors), intent(inout), optional, target err ) [private]

Solves the overdetermined or underdetermined system (A\*X = B) of M equations of N unknowns using a QR or LQ factorization of the matrix A. Notice, it is assumed that matrix A has full rank.

#### **Parameters**

in,out	а	On input, the M-by-N matrix A. On output, if $M \ge N$ , the QR factorization of A in the form as output by qr_factor; else, if $M < N$ , the LQ factorization of A.
in,out	b	If $M >= N$ , the M-element array B. On output, the first N elements contain the N-element solution array X. If $M < N$ , an N-element array with the first M elements containing the array B. On output, the N-element solution array X.
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.
out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for work, and returns without performing any actual calculations.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.</li> </ul>
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>
		LA_INVALID_OPERATION_ERROR: Occurs if a is not of full rank.

#### Notes

This routine utilizes the LAPACK routine DGELS.

Definition at line 1674 of file linalg\_solve.f90.

4.7.2.9 subroutine linalg\_solve::solve\_least\_squares\_vec\_pvt ( real(dp), dimension(:,:), intent(inout) a, real(dp), dimension(:), intent(inout) b, integer(i32), dimension(:), intent(inout), optional, target ipvt, integer(i32), intent(out), optional arnk, real(dp), dimension(:), intent(out), optional, target work, integer(i32), intent(out), optional olwork, class(errors), intent(inout), optional, target err ) [private]

Solves the overdetermined or underdetermined system (A\*X = B) of M equations of N unknowns using a complete orthogonal factorization of matrix A.

in,out	а	On input, the M-by-N matrix A. On output, the matrix is overwritten by the details of its complete orthogonal factorization.
in,out	b	If $M >= N$ , the M-element array B. On output, the first N elements contain the N-element solution array X. If $M < N$ , an N-element array with the first M elements containing the array B. On output, the N-element solution array X.
out	ipvt	An optional input that on input, an N-element array that if $IPVT(I)$ .ne. 0, the I-th column of A is permuted to the front of A $*$ P; if $IPVT(I) = 0$ , the I-th column of A is a free column. On output, if $IPVT(I) = K$ , then the I-th column of A $*$ P was the K-th column of A. If not supplied, memory is allocated internally, and $IPVT$ is set to all zeros such that all columns are treated as free.
out	arnk	An optional output, that if provided, will return the rank of a.
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.
out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for work, and returns without performing any actual calculations.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.</li> </ul>
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>

#### Notes

This routine utilizes the LAPACK routine DGELSY.

Definition at line 1932 of file linalg\_solve.f90.

4.7.2.10 subroutine linalg\_solve::solve\_least\_squares\_vec\_svd ( real(dp), dimension(:,:), intent(inout) a, real(dp), dimension(:), intent(inout) b, integer(i32), intent(out), optional arnk, real(dp), dimension(:), intent(out) s, real(dp), dimension(:), intent(out), optional, target work, integer(i32), intent(out), optional olwork, class(errors), intent(inout), optional, target err ) [private]

Solves the overdetermined or underdetermined system (A\*X = B) of M equations of N unknowns using a singular value decomposition of matrix A.

in,out	а	On input, the M-by-N matrix A. On output, the matrix is overwritten by the details of its complete orthogonal factorization.
in,out	b	If $M >= N$ , the M-by-NRHS matrix B. On output, the first N rows contain the N-by-NRHS solution matrix X. If $M < N$ , an N-by-NRHS matrix with the first M rows containing the matrix B. On output, the N-by-NRHS solution matrix X.
out	arnk	An optional output, that if provided, will return the rank of a.
out	s	A MIN(M, N)-element array that on output contains the singular values of $a$ in descending order. Notice, the condition number of $a$ can be determined by S(1) / S(MIN(M, N)).

# **Parameters**

out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.
out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for work, and returns without performing any actual calculations.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.  • LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.  • LA_CONVERGENCE_ERROR: Occurs as a warning if the QR iteration process could not converge to a zero value.

#### Notes

This routine utilizes the LAPACK routine DGELSS.

Definition at line 2204 of file linalg\_solve.f90.

4.7.2.11 subroutine linalg\_solve::solve\_lu\_mtx ( real(dp), dimension(:,:), intent(in) *a,* integer(i32), dimension(:), intent(in) *ipvt,* real(dp), dimension(:,:), intent(inout) *b,* class(errors), intent(inout), optional, target *err* ) [private]

Solves a system of LU-factored equations.

# **Parameters**

in	а	The N-by-N LU factored matrix as output by lu_factor.
in	ipvt	The N-element pivot array as output by lu_factor.
in,out	b	On input, the N-by-NRHS right-hand-side matrix. On output, the N-by-NRHS solution matrix.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if any of the input array sizes are incorrect.

# Notes

The routine is based upon the LAPACK routine DGETRS.

Definition at line 351 of file linalg\_solve.f90.

4.7.2.12 subroutine linalg\_solve::solve\_lu\_vec ( real(dp), dimension(:,:), intent(in) *a*, integer(i32), dimension(:), intent(in) *ipvt*, real(dp), dimension(:), intent(inout) *b*, class(errors), intent(inout), optional, target *err* ) [private]

Solves a system of LU-factored equations.

#### **Parameters**

in	а	The N-by-N LU factored matrix as output by lu_factor.
in	ipvt	The N-element pivot array as output by lu_factor.
in,out	b	On input, the N-element right-hand-side array. On output, the N-element solution array.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if any of the input array sizes are incorrect.

#### Notes

The routine is based upon the LAPACK routine DGETRS.

Definition at line 412 of file linalg\_solve.f90.

4.7.2.13 subroutine linalg\_solve::solve\_qr\_no\_pivot\_mtx ( real(dp), dimension(:,:), intent(inout) a, real(dp), dimension(:), intent(in) tau, real(dp), dimension(:,:), intent(inout) b, real(dp), dimension(:), intent(out), optional, target work, integer(i32), intent(out), optional olwork, class(errors), intent(inout), optional, target err ) [private]

Solves a system of M QR-factored equations of N unknowns where  $M \ge N$ .

in	а	On input, the M-by-N QR factored matrix as returned by qr_factor. On output, the contents of this matrix are restored. Notice, M must be greater than or equal to N.
in	tau	A MIN(M, N)-element array containing the scalar factors of the elementary reflectors as returned by qr_factor.
in	Ь	On input, the M-by-NRHS right-hand-side matrix. On output, the first N columns are overwritten by the solution matrix X.
		An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.
out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for work, and returns without performing any actual calculations.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.</li> </ul>
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>

#### Notes

This routine is based upon a subset of the LAPACK routine DGELS.

Definition at line 487 of file linalg solve.f90.

4.7.2.14 subroutine linalg\_solve::solve\_qr\_no\_pivot\_vec ( real(dp), dimension(:,:), intent(inout) a, real(dp), dimension(:), intent(in) tau, real(dp), dimension(:), intent(inout) b, real(dp), dimension(:), intent(out), optional, target work, integer(i32), intent(out), optional olwork, class(errors), intent(inout), optional, target err ) [private]

Solves a system of M QR-factored equations of N unknowns where  $M \ge N$ .

#### **Parameters**

in	а	On input, the M-by-N QR factored matrix as returned by qr_factor. On output, the contents of this matrix are restored. Notice, M must be greater than or equal to N.	
in	tau	A MIN(M, N)-element array containing the scalar factors of the elementary reflectors as returned by qr_factor.	
in	b	On input, the M-element right-hand-side vector. On output, the first N elements are overwritten by the solution vector X.	
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.	
out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for work, and returns without performing any actual calculations.	
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.	
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.</li> </ul>	
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>	

#### Notes

This routine is based upon a subset of the LAPACK routine DGELS.

Definition at line 602 of file linalg solve.f90.

4.7.2.15 subroutine linalg\_solve::solve\_qr\_pivot\_mtx ( real(dp), dimension(:,:), intent(inout) a, real(dp), dimension(:), intent(in) tau, integer(i32), dimension(:), intent(in) jpvt, real(dp), dimension(:,:), intent(inout) b, real(dp), dimension(:), intent(out), optional, target work, integer(i32), intent(out), optional olwork, class(errors), intent(inout), optional, target err )

[private]

Solves a system of M QR-factored equations of N unknowns where the QR factorization made use of column pivoting.

in	а	On input, the M-by-N QR factored matrix as returned by qr_factor. On output, the contents of this matrix are altered.
in	tau	A MIN(M, N)-element array containing the scalar factors of the elementary reflectors as returned by qr_factor.
in	jpvt	An N-element array, as output by qr_factor, used to track the column pivots.
in	b	On input, the MAX(M, N)-by-NRHS matrix where the first M rows contain the right-hand-side matrix B. On output, the first N rows are overwritten by the solution matrix X.
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.
out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for $work$ , and returns without performing any actual calculations.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.</li> </ul>
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>

# Notes

This routine is based upon a subset of the LAPACK routine DGELSY.

Definition at line 715 of file linalg\_solve.f90.

4.7.2.16 subroutine linalg\_solve::solve\_qr\_pivot\_vec ( real(dp), dimension(:,:), intent(inout) a, real(dp), dimension(:), intent(in) tau, integer(i32), dimension(:), intent(in) jpvt, real(dp), dimension(:), intent(inout) b, real(dp), dimension(:), intent(out), optional, target work, integer(i32), intent(out), optional olwork, class(errors), intent(inout), optional, target err )

[private]

Solves a system of M QR-factored equations of N unknowns where the QR factorization made use of column pivoting.

in	а	On input, the M-by-N QR factored matrix as returned by qr_factor. On output, the contents of this matrix are altered.
in tau A MIN(M, N)-element array containing t returned by qr_factor.		A MIN(M, N)-element array containing the scalar factors of the elementary reflectors as returned by qr_factor.
in	An N-element array, as output by qr_factor, used to track the column pivots.	
in	b	On input, the MAX(M, N)-element array where the first M elements contain the right-hand-side vector B. On output, the first N elements are overwritten by the solution vector X.
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.
out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for work, and returns without performing any actual calculations.

#### **Parameters**

out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.</li> </ul>
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>

#### Notes

This routine is based upon a subset of the LAPACK routine DGELSY.

Definition at line 902 of file linalg\_solve.f90.

4.7.2.17 subroutine linalg\_solve::solve\_tri\_mtx ( logical, intent(in) *lside,* logical, intent(in) *upper,* logical, intent(in) *trans,* logical, intent(in) *nounit,* real(dp), intent(in) *alpha,* real(dp), dimension(:,:), intent(in) *a,* real(dp), dimension(:,:), intent(inout) *b,* class(errors), intent(inout), optional, target *err* ) [private]

Solves one of the matrix equations: op(A) \* X = alpha \* B, or X \* op(A) = alpha \* B, where A is a triangular matrix.

# **Parameters**

in	Iside	Set to true to solve $op(A) * X = alpha * B$ ; else, set to false to solve $X * op(A) = alpha * B$ .
in	upper	Set to true if A is an upper triangular matrix; else, set to false if A is a lower triangular matrix.
in	trans	Set to true if $op(A) = A**T$ ; else, set to false if $op(A) = A$ .
in	nounit	Set to true if A is not a unit-diagonal matrix (ones on every diagonal element); else, set to false if A is a unit-diagonal matrix.
in	alpha	The scalar multiplier to B.
in	а	If lside is true, the M-by-M triangular matrix on which to operate; else, if lside is false, the N-by-N triangular matrix on which to operate.
in,out	b	On input, the M-by-N right-hand-side. On output, the M-by-N solution.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if a is not square, or if the sizes of a and b are not compatible.</li> </ul>

# Usage

To solve a triangular system of N equations of N unknowns A\*X = B, where A is an N-by-N upper triangular matrix, and B and X are N-by-NRHS matrices, the following code will suffice.

! Solve the system: A\*X = B, where A is an upper triangular N-by-N

#### Notes

This routine is based upon the BLAS routine DTRSM.

Definition at line 169 of file linalg solve.f90.

4.7.2.18 subroutine linalg\_solve::solve\_tri\_vec ( logical, intent(in) *upper*, logical, intent(in) *trans*, logical, intent(in) *nounit*, real(dp), dimension(:,:), intent(in) *a*, real(dp), dimension(:), intent(inout) *x*, class(errors), intent(inout), optional, target err ) [private]

Solves the system of equations: op(A) \* X = B, where A is a triangular matrix.

#### **Parameters**

in	upper	Set to true if A is an upper triangular matrix; else, set to false if A is a lower triangular matrix.
in	trans	Set to true if $op(A) = A**T$ ; else, set to false if $op(A) = A$ .
in	nounit	Set to true if A is not a unit-diagonal matrix (ones on every diagonal element); else, set to false if A is a unit-diagonal matrix.
in	а	The N-by-N triangular matrix.
in,out	X	On input, the N-element right-hand-side array. On output, the N-element solution array.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if a is not square, or if the sizes of a and b are not compatible.</li> </ul>

#### Usage

To solve a triangular system of N equations of N unknowns A\*X = B, where A is an N-by-N upper triangular matrix, and B and X are N-element arrays, the following code will suffice.

```
! Solve the system: A*X = B, where A is an upper triangular N-by-N
! matrix, and B and X are N-elements in size.
! Variables
integer(i32) :: info
real(dp), dimension(n, n) :: a
real(dp), dimension(n) :: b
! Initialize A and B...
! Solve A*X = B for X - Note: X overwrites B.
call solve_triangular_system(.true., .false., a, b)
```

#### Notes

This routine is based upon the BLAS routine DTRSV.

Definition at line 275 of file linalg solve.f90.

# 4.8 linalg\_sorting Module Reference

#### linalg sorting

# **Data Types**

· interface sort

Sorts an array.

#### **Functions/Subroutines**

• subroutine sort dbl array (x, ascend)

Sorts an array.

• subroutine sort\_dbl\_array\_ind (x, ind, ascend, err)

Sorts an array.

• subroutine sort cmplx array (x, ascend)

Sorts an array.

subroutine sort\_cmplx\_array\_ind (x, ind, ascend, err)

Sorts an array.

• subroutine sort\_eigen\_cmplx (vals, vecs, ascend, err)

A sorting routine specifically tailored for sorting of eigenvalues and their associated eigenvectors using a quick-sort approach.

subroutine sort\_eigen\_dbl (vals, vecs, ascend, err)

A sorting routine specifically tailored for sorting of eigenvalues and their associated eigenvectors using a quick-sort approach.

recursive subroutine qsort\_dbl\_ind (ascend, x, ind)

A recursive quick sort algorithm.

subroutine dbl\_partition\_ind (ascend, x, ind, marker)

A routine to perform the partioning necessary for the quick sort algorithm.

• recursive subroutine qsort\_cmplx (ascend, x)

A recursive quick sort algorithm.

• subroutine cmplx\_partition (ascend, x, marker)

A routine to perform the partioning necessary for the quick sort algorithm.

recursive subroutine qsort\_cmplx\_ind (ascend, x, ind)

A recursive quick sort algorithm.

• subroutine cmplx\_partition\_ind (ascend, x, ind, marker)

A routine to perform the partioning necessary for the quick sort algorithm.

# 4.8.1 Detailed Description

# linalg\_sorting

# **Purpose**

Provides sorting routines.

# 4.8.2 Function/Subroutine Documentation

4.8.2.1 subroutine linalg\_sorting::cmplx\_partition ( logical, intent(in) ascend, complex(dp), dimension(:), intent(inout) x, integer(i32), intent(out) marker ) [private]

A routine to perform the partioning necessary for the quick sort algorithm.

#### **Parameters**

in	ascend	Set to true to sort in ascending order; else, false to sort in descending order.
in,out	x	On input, the array to sort. On output, the sorted array.
out	marker	The partioning marker.

#### Remarks

As this routine operates on complex valued items, the complex values are sorted based upon the real component of the number.

#### Notes

This implementation is a slight modification of the code presented at  $http://www.fortran. \leftarrow com/qsort_c.f95$ .

Definition at line 578 of file linalg\_sorting.f90.

4.8.2.2 subroutine linalg\_sorting::cmplx\_partition\_ind ( logical, intent(in) ascend, complex(dp), dimension(:), intent(inout) x, integer(i32), dimension(:), intent(inout) ind, integer(i32), intent(out) marker ) [private]

A routine to perform the partioning necessary for the quick sort algorithm.

#### **Parameters**

in	ascend	Set to true to sort in ascending order; else, false to sort in descending order.
in, out	X	On input, the array to sort. On output, the sorted array.
in,out	ind	On input, a tracking array of the same length as $x$ . On output, the same array, but shuffled to match the sorting order of $x$ .
out	marker	The partioning marker.

#### Remarks

As this routine operates on complex valued items, the complex values are sorted based upon the real component of the number.

#### Notes

This implementation is a slight modification of the code presented at  $http://www.fortran. \leftarrow com/qsort_c.f95$ .

Definition at line 703 of file linalg\_sorting.f90.

4.8.2.3 subroutine linalg\_sorting::dbl\_partition\_ind ( logical, intent(in) ascend, real(dp), dimension(:), intent(inout) x, integer(i32), dimension(:), intent(inout) ind, integer(i32), intent(out) marker ) [private]

A routine to perform the partioning necessary for the quick sort algorithm.

#### **Parameters**

in	ascend	Set to true to sort in ascending order; else, false to sort in descending order.
in,out	X	On input, the array to sort. On output, the sorted array.
in,out	ind	On input, a tracking array of the same length as $x$ . On output, the same array, but shuffled to match the sorting order of $x$ .
out	marker	The partioning marker.

#### Notes

This implementation is a slight modification of the code presented at  $http://www.fortran. \leftarrow com/qsort\_c.f95$ 

Definition at line 452 of file linalg\_sorting.f90.

4.8.2.4 recursive subroutine linalg\_sorting::qsort\_cmplx ( logical, intent(in) ascend, complex(dp), dimension(:), intent(inout) x
) [private]

A recursive quick sort algorithm.

# **Parameters**

in	ascend	Set to true to sort in ascending order; else, false to sort in descending order.
in,out	X	On input, the array to sort. On output, the sorted array.

#### Remarks

As this routine operates on complex valued items, the complex values are sorted based upon the real component of the number.

#### Notes

This implementation is a slight modification of the code presented at  $http://www.fortran. \leftarrow com/qsort\_c.f95$ 

Definition at line 545 of file linalg\_sorting.f90.

4.8.2.5 recursive subroutine linalg\_sorting::qsort\_cmplx\_ind ( logical, intent(in) ascend, complex(dp), dimension(:), intent(inout) x, integer(i32), dimension(:), intent(inout) ind ) [private]

A recursive quick sort algorithm.

in	ascend	Set to true to sort in ascending order; else, false to sort in descending order.
in,out	х	On input, the array to sort. On output, the sorted array.
in,out	ind	On input, a tracking array of the same length as $x$ . On output, the same array, but shuffled to match the sorting order of $x$ .

#### Remarks

As this routine operates on complex valued items, the complex values are sorted based upon the real component of the number.

#### Notes

This implementation is a slight modification of the code presented at  $http://www.fortran. \leftarrow com/gsort_c.f95$ 

Definition at line 666 of file linalg\_sorting.f90.

4.8.2.6 recursive subroutine linalg\_sorting::qsort\_dbl\_ind ( logical, intent(in) ascend, real(dp), dimension(:), intent(inout) x, integer(i32), dimension(:), intent(inout) ind ) [private]

A recursive quick sort algorithm.

#### **Parameters**

in	ascend	Set to true to sort in ascending order; else, false to sort in descending order.
in,out	X	On input, the array to sort. On output, the sorted array.
in,out	ind	On input, a tracking array of the same length as $x$ . On output, the same array, but shuffled to match the sorting order of $x$ .

#### Notes

This implementation is a slight modification of the code presented at  $http://www.fortran. \leftarrow com/qsort\_c.f95$ .

Definition at line 419 of file linalg\_sorting.f90.

4.8.2.7 subroutine linalg\_sorting::sort\_cmplx\_array ( complex(dp), dimension(:), intent(inout) x, logical, intent(in), optional ascend ) [private]

Sorts an array.

in,out	х	On input, the array to sort. On output, the sorted array.
in	ascend	An optional input that, if specified, controls if the the array is sorted in an ascending order
		(default), or a descending order.

#### Remarks

This routine utilizes a quick sort algorithm. As this routine operates on complex valued items, the complex values are sorted based upon the real component of the number.

#### Notes

This implementation is a slight modification of the code presented at  $http://www.fortran. \leftarrow com/qsort_c.f95$ .

Definition at line 156 of file linalg\_sorting.f90.

4.8.2.8 subroutine linalg\_sorting::sort\_cmplx\_array\_ind ( complex(dp), dimension(:), intent(inout) x, integer(i32), dimension(:), intent(inout) ind, logical, intent(in), optional ascend, class(errors), intent(inout), optional, target err ) [private]

Sorts an array.

#### **Parameters**

in,out	X	On input, the array to sort. On output, the sorted array.
in,out	ind	On input, an integer array. On output, the contents of this array are shifted in the same order as that of $x$ as a means of tracking the sorting operation. It is often useful to set this array to an ascending group of values $(1, 2, n)$ such that this array tracks the original positions of the sorted array. Such an array can then be used to align other arrays. This array must be the same size as $x$ .
in	ascend	An optional input that, if specified, controls if the the array is sorted in an ascending order (default), or a descending order.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if ind is not sized to match x.

#### Remarks

This routine utilizes a quick sort algorithm. As this routine operates on complex valued items, the complex values are sorted based upon the real component of the number.

#### Notes

This implementation is a slight modification of the code presented at  $http://www.fortran. \leftarrow com/qsort_c.f95$ .

Definition at line 205 of file linalg\_sorting.f90.

4.8.2.9 subroutine linalg\_sorting::sort\_dbl\_array ( real(dp), dimension(:), intent(inout) x, logical, intent(in), optional ascend ) [private]

Sorts an array.

in,out	x	On input, the array to sort. On output, the sorted array.
in	ascend	An optional input that, if specified, controls if the the array is sorted in an ascending order
		(default), or a descending order.

#### Remarks

The routine utilizes a quick sort algorithm unless the size of the array is less than or equal to 20. For such small arrays an insertion sort algorithm is utilized.

#### Notes

This routine utilizes the LAPACK routine DLASRT.

Definition at line 47 of file linalg\_sorting.f90.

4.8.2.10 subroutine linalg\_sorting::sort\_dbl\_array\_ind ( real(dp), dimension(:), intent(inout) x, integer(i32), dimension(:), intent(inout) ind, logical, intent(in), optional ascend, class(errors), intent(inout), optional, target err ) [private]

Sorts an array.

#### **Parameters**

in,out	X	On input, the array to sort. On output, the sorted array.
in,out	ind	On input, an integer array. On output, the contents of this array are shifted in the same order as that of $x$ as a means of tracking the sorting operation. It is often useful to set this array to an ascending group of values $(1, 2, n)$ such that this array tracks the original positions of the sorted array. Such an array can then be used to align other arrays. This array must be the same size as $x$ .
in	ascend	An optional input that, if specified, controls if the the array is sorted in an ascending order (default), or a descending order.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if ind is not sized to match x.

# Remarks

This routine utilizes a quick sort algorithm explained at http://www.fortran.com/qsort\_c.f95.

Definition at line 97 of file linalg\_sorting.f90.

4.8.2.11 subroutine linalg\_sorting::sort\_eigen\_cmplx ( complex(dp), dimension(:), intent(inout) vals, complex(dp), dimension(:,:), intent(inout) vecs, logical, intent(in), optional ascend, class(errors), intent(inout), optional, target err )

[private]

A sorting routine specifically tailored for sorting of eigenvalues and their associated eigenvectors using a quick-sort approach.

# **Parameters**

in,out	vals	On input, an N-element array containing the eigenvalues. On output, the sorted eigenvalues.
in,out	vecs	On input, an N-by-N matrix containing the eigenvectors associated with $vals$ (one vector per column). On output, the sorted eigenvector matrix.
in	ascend	An optional input that, if specified, controls if the the array is sorted in an ascending order (default), or a descending order.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if vecs is not sized to match vals.  • LA_OUT_OF_MEMORY_ERROR: Occurs if there is insufficient memory available to comoplete this operation.

Definition at line 267 of file linalg\_sorting.f90.

4.8.2.12 subroutine linalg\_sorting::sort\_eigen\_dbl ( real(dp), dimension(:), intent(inout) vals, real(dp), dimension(:,:), intent(inout) vecs, logical, intent(in), optional ascend, class(errors), intent(inout), optional, target err )

[private]

A sorting routine specifically tailored for sorting of eigenvalues and their associated eigenvectors using a quick-sort approach.

#### **Parameters**

in,out	vals	On input, an N-element array containing the eigenvalues. On output, the sorted eigenvalues.
in,out	vecs	On input, an N-by-N matrix containing the eigenvectors associated with $vals$ (one vector per column). On output, the sorted eigenvector matrix.
in	ascend	An optional input that, if specified, controls if the the array is sorted in an ascending order (default), or a descending order.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if vecs is not sized to match vals.  • LA_OUT_OF_MEMORY_ERROR: Occurs if there is insufficient memory available to comoplete this operation.

Definition at line 345 of file linalg\_sorting.f90.

# **Chapter 5**

# **Data Type Documentation**

# 5.1 linalg\_core::diag\_mtx\_mult Interface Reference

Multiplies a diagonal matrix with another matrix or array.

#### **Private Member Functions**

- subroutine diag mtx mult mtx (Iside, trans, alpha, a, b, beta, c, err)
  - Computes the matrix operation: C = alpha \* A \* op(B) + beta \* C, or C = alpha \* op(B) \* A + beta \* C.
- subroutine diag\_mtx\_mult\_mtx2 (Iside, alpha, a, b, err)
  - Computes the matrix operation: B = alpha \* A \* op(B), or B = alpha \* op(B) \* A.
- subroutine diag\_mtx\_mult\_mtx3 (Iside, trans, alpha, a, b, beta, c, err)
  - Computes the matrix operation: C = alpha \* A \* op(B) + beta \* C, or C = alpha \* op(B) \* A + beta \* C, where A and C are complex-valued.
- subroutine diag\_mtx\_mult\_mtx4 (Iside, trans, alpha, a, b, beta, c, err)
  - Computes the matrix operation: C = alpha \* A \* op(B) + beta \* C, or C = alpha \* op(B) \* A + beta \* C, where A, B, and C are complex-valued.

# 5.1.1 Detailed Description

Multiplies a diagonal matrix with another matrix or array.

Definition at line 46 of file linalg core.f90.

# 5.1.2 Member Function/Subroutine Documentation

subroutine linalg\_core::diag\_mtx\_mult::diag\_mtx\_mult\_mtx ( logical, intent(in) *lside,* logical, intent(in) *trans,* real(dp) *alpha,* real(dp), dimension(:), intent(in) *a,* real(dp), dimension(:), intent(in) *b,* real(dp) *beta,* real(dp), dimension(:), intent(inout) *c,* class(errors), intent(inout), optional, target *err* ) [private]

Computes the matrix operation: C = alpha \* A \* op(B) + beta \* C, or C = alpha \* op(B) \* A + beta \* C.

in	Iside	Set to true to apply matrix A from the left; else, set to false to apply matrix A from the left.
in	trans	Set to true if $op(B) == B**T$ ; else, set to false if $op(B) == B$ .
in	alpha	A scalar multiplier.
in	а	A K-element array containing the diagonal elements of A where $K = MIN(M,P)$ if lside is true; else, if lside is false, $K = MIN(N,P)$ .
in	b	The LDB-by-TDB matrix B where (LDB = leading dimension of B, and TDB = trailing dimension of B):  • lside == true & trans == true: LDB = N, TDB = P  • lside == true & trans == false: LDB = P, TDB = N  • lside == false & trans == true: LDB = P, TDB = M  • lside == false & trans == false: LDB = M, TDB = P
in	beta	A scalar multiplier.
in,out	С	On input, the M-by-N matrix C. On output, the resulting M-by-N matrix.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if any of the input array sizes are incorrect.

Definition at line 329 of file linalg\_core.f90.

5.1.2.2 subroutine linalg\_core::diag\_mtx\_mult::diag\_mtx\_mult\_mtx2 ( logical, intent(in) *lside*, real(dp), intent(in) *alpha*, real(dp), dimension(:), intent(in) *a*, real(dp), dimension(:,:), intent(inout) *b*, class(errors), intent(inout), optional, target *err* ) [private]

Computes the matrix operation: B = alpha \* A \* op(B), or B = alpha \* op(B) \* A.

#### **Parameters**

in	Iside	Set to true to apply matrix A from the left; else, set to false to apply matrix A from the left.
in	alpha	A scalar multiplier.
in	а	A K-element array containing the diagonal elements of A where $K = MIN(M,P)$ if lside is true; else, if lside is false, $K = MIN(N,P)$ .
in	b	On input, the M-by-N matrix B. On output, the resulting M-by-N matrix.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if any of the input array sizes are incorrect.

Definition at line 496 of file linalg\_core.f90.

5.1.2.3 subroutine linalg\_core::diag\_mtx\_mult::diag\_mtx\_mult\_mtx3 ( logical, intent(in) *lside*, logical, intent(in) *trans*, real(dp) *alpha*, complex(dp), dimension(:,:), intent(in) *a*, real(dp), dimension(:,:), intent(in) *b*, real(dp) *beta*, complex(dp), dimension(:,:), intent(inout) *c*, class(errors), intent(inout), optional, target *err* ) [private]

Computes the matrix operation: C = alpha \* A \* op(B) + beta \* C, or C = alpha \* op(B) \* A + beta \* C, where A and C are complex-valued.

#### **Parameters**

in	Iside	Set to true to apply matrix A from the left; else, set to false to apply matrix A from the left.
in	trans	Set to true if $op(B) == B**T$ ; else, set to false if $op(B) == B$ .
in	alpha	A scalar multiplier.
in	а	A K-element array containing the diagonal elements of A where $K = MIN(M,P)$ if lside is true; else, if lside is false, $K = MIN(N,P)$ .
in	b	The LDB-by-TDB matrix B where (LDB = leading dimension of B, and TDB = trailing dimension of B):  • lside == true & trans == true: LDB = N, TDB = P  • lside == true & trans == false: LDB = P, TDB = N  • lside == false & trans == true: LDB = P, TDB = M  • lside == false & trans == false: LDB = M, TDB = P
in	beta	A scalar multiplier.
in,out	С	On input, the M-by-N matrix C. On output, the resulting M-by-N matrix.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if any of the input array sizes are incorrect.

Definition at line 579 of file linalg\_core.f90.

5.1.2.4 subroutine linalg\_core::diag\_mtx\_mult::diag\_mtx\_mult\_mtx4 ( logical, intent(in) *lside*, logical, intent(in) *trans*, real(dp) *alpha*, complex(dp), dimension(:,:), intent(in) *a*, complex(dp), dimension(:,:), intent(in) *b*, real(dp) *beta*, complex(dp), dimension(:,:), intent(inout) *c*, class(errors), intent(inout), optional, target *err* ) [private]

Computes the matrix operation: C = alpha \* A \* op(B) + beta \* C, or C = alpha \* op(B) \* A + beta \* C, where A, B, and C are complex-valued.

in	Iside	Set to true to apply matrix A from the left; else, set to false to apply matrix A from the left.
in	trans	Set to true if $op(B) == B**T$ ; else, set to false if $op(B) == B$ .
in	alpha	A scalar multiplier.
in	а	A K-element array containing the diagonal elements of A where K = MIN(M,P) if lside is
		true; else, if lside is false, K = MIN(N,P).

in	b	The LDB-by-TDB matrix B where:
		• lside == true & trans == true: LDA = N, TDB = P
		• lside == true & trans == false: LDA = P, TDB = N
		• lside == false & trans == true: LDA = P, TDB = M
		• lside == false & trans == false: LDA = M, TDB = P
in	beta	A scalar multiplier.
in,out	С	On input, the M-by-N matrix C. On output, the resulting M-by-N matrix.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if any of the input array sizes are incorrect.

Definition at line 755 of file linalg\_core.f90.

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

• /home/jason/Documents/Code/linalg/src/linalg\_core.f90

# 5.2 lapack::DLAMCH Interface Reference

**Public Member Functions** 

• real(dp) function **dlamch** (cmach)

# 5.2.1 Detailed Description

Definition at line 14 of file lapack.f90.

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

• /home/jason/Documents/Code/linalg/src/lapack.f90

# 5.3 linalg\_eigen::eigen Interface Reference

Computes the eigenvalues, and optionally the eigenvectors, of a matrix.

#### **Private Member Functions**

- subroutine eigen\_symm (vecs, a, vals, work, olwork, err)
  - Computes the eigenvalues, and optionally the eigenvectors of a real, symmetric matrix.
- subroutine eigen\_asymm (a, vals, vecs, work, olwork, err)
  - Computes the eigenvalues, and optionally the right eigenvectors of a square matrix.
- subroutine eigen\_gen (a, b, alpha, beta, vecs, work, olwork, err)

Computes the eigenvalues, and optionally the right eigenvectors of a square matrix assuming the structure of the eigenvalue problem is A\*X = lambda\*B\*X.

# 5.3.1 Detailed Description

Computes the eigenvalues, and optionally the eigenvectors, of a matrix.

#### See Also

- Wikipedia
- Wolfram MathWorld
- LAPACK Users Manual

Definition at line 24 of file linalg\_eigen.f90.

#### 5.3.2 Member Function/Subroutine Documentation

5.3.2.1 subroutine linalg\_eigen::eigen\_asymm ( real(dp), dimension(:,:), intent(inout) a, complex(dp), dimension(:), intent(out) vals, complex(dp), dimension(:,:), intent(out), optional vecs, real(dp), dimension(:), intent(out), optional, pointer work, integer(i32), intent(out), optional olwork, class(errors), intent(inout), optional, target err ) [private]

Computes the eigenvalues, and optionally the right eigenvectors of a square matrix.

in,out	а	On input, the N-by-N matrix on which to operate. On output, the contents of this matrix are overwritten.
out	vals	An N-element array containing the eigenvalues of the matrix. The eigenvalues are not sorted.
out	vecs	An optional N-by-N matrix, that if supplied, signals to compute the right eigenvectors (one per column). If not provided, only the eigenvalues will be computed.
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.
out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for work, and returns without performing any actual calculations.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.</li> </ul>
		LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.
enerated by De	xygen	LA_CONVERGENCE_ERROR: Occurs if the algorithm failed to converge.
	1	

#### Notes

This routine utilizes the LAPACK routine DGEEV.

Definition at line 185 of file linalg\_eigen.f90.

5.3.2.2 subroutine linalg\_eigen::eigen\_gen ( real(dp), dimension(:,:), intent(inout) a, real(dp), dimension(:,:), intent(inout) b, complex(dp), dimension(:), intent(out) alpha, real(dp), dimension(:), intent(out), optional beta, complex(dp), dimension(:,:), intent(out), optional vecs, real(dp), dimension(:), intent(out), optional, pointer work, integer(i32), intent(out), optional olwork, class(errors), intent(inout), optional, target err ) [private]

Computes the eigenvalues, and optionally the right eigenvectors of a square matrix assuming the structure of the eigenvalue problem is A\*X = lambda\*B\*X.

#### **Parameters**

in,out	а	On input, the N-by-N matrix A. On output, the contents of this matrix are overwritten.
in,out	b	On input, the N-by-N matrix B. On output, the contents of this matrix are overwritten.
out	alpha	An N-element array that, if beta is not supplied, contains the eigenvalues. If beta is supplied however, the eigenvalues must be computed as ALPHA / BETA. This however, is not as trivial as it seems as it is entirely possible, and likely, that ALPHA / BETA can overflow or underflow. With that said, the values in ALPHA will always be less than and usually comparable with the NORM(A).
out	beta	An optional N-element array that if provided forces alpha to return the numerator, and this array contains the denominator used to determine the eigenvalues as ALPHA / BETA. If used, the values in this array will always be less than and usually comparable with the NORM(B).
out	vecs	An optional N-by-N matrix, that if supplied, signals to compute the right eigenvectors (one per column). If not provided, only the eigenvalues will be computed.
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.
out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for work, and returns without performing any actual calculations.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.</li> </ul>
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>
		LA_CONVERGENCE_ERROR: Occurs if the algorithm failed to converge.

#### Usage

As an example, consider the eigenvalue problem arising from a mechanical system of masses and springs such that the masses are described by a mass matrix M, and the arrangement of springs are described by a stiffness matrix K.

- ! This is an example illustrating the use of the eigenvalue and eigenvector
- ! routines to solve a free vibration problem of 3 masses connected by springs.

```
As illustrated above, the system consists of 3 masses connected by springs. Spring k1 and spring k4 connect the end masses to ground. The equations of
        motion for this system are as follows.
 ! | m1 0 0 | |x1" | | k1+k2 -k2
! | 0 m2 0 | |x2" | + | -k2 k2+k3
! | 0 0 m3 | |x3" | 0 -k3
                                                                                               | k1+k2 -k2
                                                                                                                                                                          0 | |x1|
                                                                                                                                                                        -k3 \mid |x2| = |0|
                                                                                                                                                                   k3+k4| |x3|
       Notice: x1" = the second time derivative of x1.
program example
                use linalg_constants, only : dp, i32
                 use linalg_eigen
                implicit none
                ! Define the model parameters
                 real(dp), parameter :: pi = 3.14159265359d0
                 real(dp), parameter :: m1 = 0.5d0
                 real(dp), parameter :: m2 = 2.5d0
                 real(dp), parameter :: m3 = 0.75d0
                 real(dp), parameter :: k1 = 5.0d6
                 real(dp), parameter :: k2 = 10.0d6
                 real(dp), parameter :: k3 = 10.0d6
                 real(dp), parameter :: k4 = 5.0d6
                 ! Local Variables
                integer(i32) :: i, j
real(dp) :: m(3,3), k(3,3), natfreq(3)
             complex(dp) :: vals(3), modeshapes(3,3)
                 ! Define the mass matrix % \left( 1\right) =\left( 1\right) \left( 1\right) 
               m = reshape([m1, 0.0d0, 0.0d0, 0.0d0, m2, 0.0d0, 0.0d0, 0.0d0, m3], [3, 3])
                 ! Define the stiffness matrix
                 k = reshape([k1 + k2, -k2, 0.0d0, -k2, k2 + k3, -k3, 0.0d0, -k3, k3 + k4], &
                 ! Compute the eigenvalues and eigenvectors.
                call eigen(k, m, vals, vecs = modeshapes)
                 ! Compute the natural frequency values, and return them with units of Hz.
                  ! Notice, all eigenvalues and eigenvectors are real for this example.
                 natfreq = sqrt(real(vals)) / (2.0d0 * pi)
                 ! Display the natural frequency and mode shape values. Notice, the eigen
               ! routine does not necessarily sort the values. print ' (A)', "Modal Information (Not Sorted):"
                do i = 1, size(natfreq)
print '(AIOAF8.4A)', "Mode ", i, ": (", natfreq(i), " Hz)"
                                 print '(F10.3)', (real(modeShapes(j,i)), j = 1, size(natfreq))
                 end do
end program
```

#### **Notes**

This routine utilizes the LAPACK routine DGGEV.

Definition at line 465 of file linalg\_eigen.f90.

5.3.2.3 subroutine linalg\_eigen::eigen\_symm ( logical, intent(in) *vecs*, real(dp), dimension(:,:), intent(inout) *a*, real(dp), dimension(:), intent(out), optional, pointer *work*, integer(i32), intent(out), optional *olwork*, class(errors), intent(inout), optional, target *err* ) [private]

Computes the eigenvalues, and optionally the eigenvectors of a real, symmetric matrix.

in	vecs	Set to true to compute the eigenvectors as well as the eigenvalues; else, set to false to
		just compute the eigenvalues.

in,out	а	On input, the N-by-N symmetric matrix on which to operate. On output, and if vecs is set to true, the matrix will contain the eigenvectors (one per column) corresponding to each eigenvalue in vals. If vecs is set to false, the lower triangular portion of the matrix is overwritten.
out	vals	An N-element array that will contain the eigenvalues sorted into ascending order.
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.
out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for work, and returns without performing any actual calculations.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.</li> </ul>
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>
		LA_CONVERGENCE_ERROR: Occurs if the algorithm failed to converge.

#### Notes

This routine utilizes the LAPACK routine DSYEV.

Definition at line 68 of file linalg\_eigen.f90.

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

/home/jason/Documents/Code/linalg/src/linalg\_eigen.f90

# 5.4 linalg\_factor::form\_lu Interface Reference

Extracts the L and U matrices from the condensed [L\U] storage format used by the lu\_factor.

#### **Private Member Functions**

• subroutine form\_lu\_all (lu, ipvt, u, p, err)

Extracts the L, U, and P matrices from the output of the lu\_factor routine.

• subroutine form\_lu\_only (lu, u, err)

Extracts the L, and U matrices from the output of the lu\_factor routine.

# 5.4.1 Detailed Description

Extracts the L and U matrices from the condensed [L\U] storage format used by the lu\_factor.

Definition at line 31 of file linalg\_factor.f90.

# 5.4.2 Member Function/Subroutine Documentation

5.4.2.1 subroutine linalg\_factor::form\_lu::form\_lu\_all ( real(dp), dimension(:,:), intent(inout) *lu*, integer(i32), dimension(:), intent(in) *ipvt*, real(dp), dimension(:,:), intent(out) *u*, real(dp), dimension(:,:), intent(out) *p*, class(errors), intent(inout), optional, target *err* ) [private]

Extracts the L, U, and P matrices from the output of the lu factor routine.

#### **Parameters**

in, out	lu	On input, the N-by-N matrix as output by lu_factor. On output, the N-by-N lower triangular
		matrix L.
in	ipvt	The N-element pivot array as output by lu_factor.
out	и	An N-by-N matrix where the U matrix will be written.
out	р	An N-by-N matrix where the row permutation matrix will be written.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if any of the input array sizes are incorrect.

#### Remarks

This routine allows extraction of the actual "L", "U", and "P" matrices of the decomposition. To use these matrices to solve the system A\*X = B, the following approach is used.

- 1. First, solve the linear system: L\*Y = P\*B for Y.
- 2. Second, solve the linear system: U\*X = Y for X.

Notice, as both L and U are triangular in structure, the above equations can be solved by forward and backward substitution.

# See Also

- Wikipedia
- Wolfram MathWorld

Definition at line 211 of file linalg\_factor.f90.

5.4.2.2 subroutine linalg\_factor::form\_lu::form\_lu\_only ( real(dp), dimension(:,:), intent(inout) *lu*, real(dp), dimension(:,:), intent(out) *u*, class(errors), intent(inout), optional, target *err* ) [private]

Extracts the L, and U matrices from the output of the lu\_factor routine.

in,out	lu	On input, the N-by-N matrix as output by lu_factor. On output, the N-by-N lower triangular
		matrix L.
out	и	An N-by-N matrix where the U matrix will be written.
out Generated by D	<i>err</i> oxygen	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
Generated by D	oxygen	errors class is used internally to provide error handling. Possible errors and warning

Definition at line 288 of file linalg\_factor.f90.

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

· /home/jason/Documents/Code/linalg/src/linalg factor.f90

# 5.5 linalg\_factor::form\_gr Interface Reference

Forms the full M-by-M orthogonal matrix Q from the elementary reflectors returned by the base QR factorization algorithm.

#### **Private Member Functions**

- subroutine form\_qr\_no\_pivot (r, tau, q, work, olwork, err)

  Forms the full M-by-M orthogonal matrix Q from the elementary reflectors returned by the base QR factorization algorithm.
- subroutine form\_qr\_pivot (r, tau, pvt, q, p, work, olwork, err)

Forms the full M-by-M orthogonal matrix Q from the elementary reflectors returned by the base QR factorization algorithm.

# 5.5.1 Detailed Description

Forms the full M-by-M orthogonal matrix Q from the elementary reflectors returned by the base QR factorization algorithm.

#### See Also

- Wikipedia
- LAPACK Users Manual

Definition at line 55 of file linalg\_factor.f90.

#### 5.5.2 Member Function/Subroutine Documentation

5.5.2.1 subroutine linalg\_factor::form\_qr::form\_qr\_no\_pivot ( real(dp), dimension(:,:), intent(inout) r, real(dp), dimension(:), intent(in) tau, real(dp), dimension(:,:), intent(out) q, real(dp), dimension(:), intent(out), optional, target work, integer(i32), intent(out), optional olwork, class(errors), intent(inout), optional, target err ) [private]

Forms the full M-by-M orthogonal matrix Q from the elementary reflectors returned by the base QR factorization algorithm.

in,out	r	On input, an M-by-N matrix where the elements below the diagonal contain the
		elementary reflectors generated from the QR factorization. On and above the diagonal,
		the matrix contains the matrix R. On output, the elements below the diagonal are zeroed
		such that the remaining matrix is simply the M-by-N matrix R.

in	tau	A MIN(M, N)-element array containing the scalar factors of each elementary reflector defined in r.
out	q	An M-by-M matrix where the full orthogonal matrix Q will be written. In the event that $M > N$ , Q may be supplied as M-by-N, and therefore only return the useful submatrix Q1 (Q = [Q1, Q2]) as the factorization can be written as Q * R = [Q1, Q2] * [R1; 0].
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.
out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for work, and returns without performing any actual calculations.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.</li> </ul>
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>

#### Notes

This routine utilizes the LAPACK routine DORGQR.

Definition at line 694 of file linalg\_factor.f90.

5.5.2.2 subroutine linalg\_factor::form\_qr::form\_qr\_pivot ( real(dp), dimension(:,:), intent(inout) r, real(dp), dimension(:), intent(in) tau, integer(i32), dimension(:), intent(in) pvt, real(dp), dimension(:,:), intent(out) q, real(dp), dimension(:,:), intent(out) p, real(dp), dimension(:), intent(out), optional, target work, integer(i32), intent(out), optional olwork, class(errors), intent(inout), optional, target err ) [private]

Forms the full M-by-M orthogonal matrix Q from the elementary reflectors returned by the base QR factorization algorithm.

in,out	r	On input, an M-by-N matrix where the elements below the diagonal contain the elementary reflectors generated from the QR factorization. On and above the diagonal, the matrix contains the matrix R. On output, the elements below the diagonal are zeroed such that the remaining matrix is simply the M-by-N matrix R.
in	tau	A MIN(M, N)-element array containing the scalar factors of each elementary reflector defined in r.
in	pvt	An N-element column pivot array as returned by the QR factorization.
out	q	An M-by-M matrix where the full orthogonal matrix Q will be written. In the event that $M > N$ , Q may be supplied as M-by-N, and therefore only return the useful submatrix Q1 (Q = [Q1, Q2]) as the factorization can be written as Q * R = [Q1, Q2] * [R1; 0].
out	р	An N-by-N matrix where the pivot matrix will be written.
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.

out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for work, and returns without performing any actual calculations.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>

#### **Notes**

This routine utilizes the LAPACK routine DORGQR.

Definition at line 825 of file linalg\_factor.f90.

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

/home/jason/Documents/Code/linalg/src/linalg\_factor.f90

# 5.6 linalg\_core::mtx\_mult Interface Reference

Performs the matrix operation: C = alpha \* op(A) \* op(B) + beta \* C.

# **Private Member Functions**

- subroutine mtx\_mult\_mtx (transa, transb, alpha, a, b, beta, c, err)
   Performs the matrix operation: C = alpha \* op(A) \* op(B) + beta \* C.
- subroutine mtx\_mult\_vec (trans, alpha, a, b, beta, c, err)

  Performs the matrix-vector operation: c = alpha \* op(A) \* b + beta \* c.

#### 5.6.1 Detailed Description

Performs the matrix operation: C = alpha \* op(A) \* op(B) + beta \* C.

Definition at line 39 of file linalg\_core.f90.

# 5.6.2 Member Function/Subroutine Documentation

5.6.2.1 subroutine linalg\_core::mtx\_mult::mtx ( logical, intent(in) transa, logical, intent(in) transb, real(dp), intent(in) alpha, real(dp), dimension(:,:), intent(in) a, real(dp), dimension(:,:), intent(in) b, real(dp), intent(in) beta, real(dp), dimension(:,:), intent(inout) c, class(errors), intent(inout), optional, target err ) [private]

Performs the matrix operation: C = alpha \* op(A) \* op(B) + beta \* C.

in	transa	Set to true if $op(A) = A**T$ ; else, set to false for $op(A) = A$ .
in	transb	Set to true if $op(B) = B**T$ ; else, set to false for $op(B) = B$ .
in	alpha	A scalar multiplier.
in	а	If transa is set to true, an K-by-M matrix; else, if transa is set to false, an M-by-K matrix.
in	b	If transb is set to true, an N-by-K matrix; else, if transb is set to false, a K-by-N matrix.
in	beta	A scalar multiplier.
in,out	С	On input, the M-by-N matrix C. On output, the M-by-N result.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if any of the input array sizes are incorrect.

# Notes

This routine utilizes the BLAS routine DGEMM.

Definition at line 85 of file linalg\_core.f90.

5.6.2.2 subroutine linalg\_core::mtx\_mult::mtx\_mult\_vec ( logical, intent(in) trans, real(dp), intent(in) alpha, real(dp), dimension(:), intent(in) b, real(dp), intent(in) beta, real(dp), dimension(:), intent(inout) c, class(errors), intent(inout), optional, target err ) [private]

Performs the matrix-vector operation: c = alpha \* op(A) \* b + beta \* c.

# **Parameters**

in	trans	Set to true if $op(A) = A**T$ ; else, set to false for $op(A) = A$ .
in	alpha	A scalar multiplier.
in	а	The M-by-N matrix A.
in	b	If trans is set to true, an M-element array; else, if trans is set to false, an N-element array.
in	beta	A scalar multiplier.
in,out	С	On input, if trans is set to true, an N-element array; else, if trans is set to false, an M-element array. On output, the results of the operation.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if any of the input array sizes are incorrect.

# Notes

This routine utilizes the BLAS routine DGEMV.

Definition at line 179 of file linalg\_core.f90.

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

/home/jason/Documents/Code/linalg/src/linalg core.f90

# 5.7 linalg\_factor::mult\_gr Interface Reference

Multiplies a general matrix by the orthogonal matrix Q from a QR factorization.

#### **Private Member Functions**

- subroutine mult\_qr\_mtx (Iside, trans, a, tau, c, work, olwork, err)
   Multiplies a general matrix by the orthogonal matrix Q from a QR factorization such that: C = op(Q) \* C, or C = C \* op(Q)
- subroutine mult\_qr\_vec (trans, a, tau, c, work, olwork, err)
   Multiplies a vector by the orthogonal matrix Q from a QR factorization such that: C = op(Q) \* C.

# 5.7.1 Detailed Description

Multiplies a general matrix by the orthogonal matrix Q from a QR factorization.

Definition at line 63 of file linalg factor.f90.

# 5.7.2 Member Function/Subroutine Documentation

5.7.2.1 subroutine linalg\_factor::mult\_qr::mult\_qr\_mtx ( logical, intent(in) *lside*, logical, intent(in) *trans*, real(dp), dimension(:,:), intent(inout) *a*, real(dp), dimension(:), intent(inout) *a*, real(dp), dimension(:), intent(out), optional, target *work*, integer(i32), intent(out), optional *olwork*, class(errors), intent(inout), optional, target *err* ) [private]

Multiplies a general matrix by the orthogonal matrix Q from a QR factorization such that: C = op(Q) \* C, or C = C \* op(Q).

in	Iside	Set to true to apply Q or Q**T from the left; else, set to false to apply Q or Q**T from the right.
in	trans	Set to true to apply Q**T; else, set to false.
in	а	On input, an LDA-by-K matrix containing the elementary reflectors output from the QR factorization. If lside is set to true, LDA = M, and $M >= K >= 0$ ; else, if lside is set to false, LDA = N, and $N >= K >= 0$ . Notice, the contents of this matrix are restored on exit.
in	tau	A K-element array containing the scalar factors of each elementary reflector defined in a.
in,out	С	On input, the M-by-N matrix C. On output, the product of the orthogonal matrix Q and the original matrix C.
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.

out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for work, and returns without performing any actual calculations.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>

#### Notes

This routine utilizes the LAPACK routine DORMQR.

Definition at line 927 of file linalg\_factor.f90.

5.7.2.2 subroutine linalg\_factor::mult\_qr::mult\_qr\_vec ( logical, intent(in) *trans*, real(dp), dimension(:,:), intent(inout) *a*, real(dp), dimension(:), intent(in) *tau*, real(dp), dimension(:), intent(inout) *c*, real(dp), dimension(:), intent(out), optional, target *work*, integer(i32), intent(out), optional *olwork*, class(errors), intent(inout), optional, target *err* ) [private]

Multiplies a vector by the orthogonal matrix Q from a QR factorization such that: C = op(Q) \* C.

in	trans	Set to true to apply Q**T; else, set to false.
in	а	On input, an M-by-K matrix containing the elementary reflectors output from the QR factorization. Notice, the contents of this matrix are restored on exit.
in	tau	A K-element array containing the scalar factors of each elementary reflector defined in a.
in,out	С	On input, the M-element vector C. On output, the product of the orthogonal matrix Q and the original vector C.
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.
out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for $work$ , and returns without performing any actual calculations.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.</li> </ul>
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>

#### Notes

This routine is based upon the LAPACK routine DORM2R.

Definition at line 1054 of file linalg factor.f90.

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

· /home/jason/Documents/Code/linalg/src/linalg factor.f90

# 5.8 linalg\_factor::mult\_rz Interface Reference

Multiplies a general matrix by the orthogonal matrix Z from an RZ factorization.

#### **Private Member Functions**

- subroutine mult\_rz\_mtx (Iside, trans, I, a, tau, c, work, olwork, err)
   Multiplies a general matrix by the orthogonal matrix Z from an RZ factorization such that: C = op(Z) \* C, or C = C \* op(Z).
- subroutine mult\_rz\_vec (trans, I, a, tau, c, work, olwork, err)
   Multiplies a vector by the orthogonal matrix Z from an RZ factorization such that: C = op(Z) \* C.

# 5.8.1 Detailed Description

Multiplies a general matrix by the orthogonal matrix Z from an RZ factorization.

Definition at line 71 of file linalg\_factor.f90.

# 5.8.2 Member Function/Subroutine Documentation

5.8.2.1 subroutine linalg\_factor::mult\_rz::mult\_rz\_mtx ( logical, intent(in) *lside*, logical, intent(in) *trans*, integer(i32), intent(in) *l*, real(dp), dimension(:,:), intent(inout) *a*, real(dp), dimension(:), intent(inout) *a*, real(dp), dimension(:), intent(inout), optional, target *work*, integer(i32), intent(out), optional *olwork*, class(errors), intent(inout), optional, target *err* ) [private]

Multiplies a general matrix by the orthogonal matrix Z from an RZ factorization such that: C = op(Z) \* C, or C = C \* op(Z).

in	Iside	Set to true to apply Z or Z**T from the left; else, set to false to apply Z or Z**T from the right.
	4	Oaks Awa to cook 7. To dee and to false
in	trans	Set to true to apply Z**T; else, set to false.
in	1	The number of columns in matrix a containing the meaningful part of the Householder
		vectors. If lside is true, $M >= L >= 0$ ; else, if lside is false, $N >= L >= 0$ .
in,out	а	On input the K-by-LTA matrix Z, where LTA = M if lside is true; else, LTA = N if lside
		is false. The I-th row must contain the Householder vector in the last k rows. Notice, the
		contents of this matrix are restored on exit.

in	tau	A K-element array containing the scalar factors of the elementary reflectors, where $M >= K >= 0$ if lside is true; else, $N >= K >= 0$ if lside is false.
in,out	С	On input, the M-by-N matrix C. On output, the product of the orthogonal matrix Z and the original matrix C.
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.
out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for work, and returns without performing any actual calculations.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.</li> </ul>
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>

#### Notes

This routine utilizes the LAPACK routine DORMRZ.

Definition at line 1745 of file linalg\_factor.f90.

5.8.2.2 subroutine linalg\_factor::mult\_rz::mult\_rz\_vec ( logical, intent(in) trans, integer(i32), intent(in) l, real(dp), dimension(:,:), intent(inout) a, real(dp), dimension(:), intent(in) tau, real(dp), dimension(:), intent(inout) c, real(dp), dimension(:), intent(out), optional, target work, integer(i32), intent(out), optional olwork, class(errors), intent(inout), optional, target err ) [private]

Multiplies a vector by the orthogonal matrix Z from an RZ factorization such that: C = op(Z) \* C.

in	trans	Set to true to apply Z**T; else, set to false.
in	1	The number of columns in matrix a containing the meaningful part of the Householder vectors. If lside is true, $M >= L >= 0$ ; else, if lside is false, $N >= L >= 0$ .
in,out	а	On input the K-by-LTA matrix Z, where LTA = M if lside is true; else, LTA = N if lside is false. The I-th row must contain the Householder vector in the last k rows. Notice, the contents of this matrix are restored on exit.
in	tau	A K-element array containing the scalar factors of the elementary reflectors, where $M \ge K \ge 0$ if lside is true; else, $N \ge K \ge 0$ if lside is false.
in,out	С	On input, the M-element array C. On output, the product of the orthogonal matrix Z and the original array C.
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.
out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for work, and returns without performing any actual calculations.

out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.</li> </ul>
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>

#### Notes

This routine utilizes the LAPACK routine DORMRZ.

Definition at line 1887 of file linalg\_factor.f90.

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

/home/jason/Documents/Code/linalg/src/linalg\_factor.f90

# 5.9 linalg\_factor::qr\_factor Interface Reference

Computes the QR factorization of an M-by-N matrix.

# **Private Member Functions**

- subroutine qr\_factor\_no\_pivot (a, tau, work, olwork, err)
   Computes the QR factorization of an M-by-N matrix without pivoting.
- subroutine qr\_factor\_pivot (a, tau, jpvt, work, olwork, err)
   Computes the QR factorization of an M-by-N matrix with column pivoting such that A \* P = Q \* R.

# 5.9.1 Detailed Description

Computes the QR factorization of an M-by-N matrix.

# See Also

- Wikipedia
- Wolfram MathWorld
- LAPACK Users Manual

Definition at line 43 of file linalg\_factor.f90.

# 5.9.2 Member Function/Subroutine Documentation

5.9.2.1 subroutine linalg\_factor::qr\_factor::qr\_factor\_no\_pivot ( real(dp), dimension(:,:), intent(inout) *a,* real(dp), dimension(:), intent(out) *tau,* real(dp), dimension(:), intent(out), optional, target *work,* integer(i32), intent(out), optional *olwork,* class(errors), intent(inout), optional, target *err* ) [private]

Computes the QR factorization of an M-by-N matrix without pivoting.

in,out	а	On input, the M-by-N matrix to factor. On output, the elements on and above the diagonal contain the MIN(M, N)-by-N upper trapezoidal matrix R (R is upper triangular if $M \ge N$ ). The elements below the diagonal, along with the array tau, represent the orthogonal matrix Q as a product of elementary reflectors.
out	tau	A MIN(M, N)-element array used to store the scalar factors of the elementary reflectors.
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.
out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for $\mathtt{work}$ , and returns without performing any actual calculations.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if tau or work are not sized appropriately.  • LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.

#### Remarks

QR factorization without pivoting is best suited to solving an overdetermined system in least-squares terms, or to solve a normally defined system. To solve an underdetermined system, it is recommended to use either LQ factorization, or a column-pivoting based QR factorization.

# Usage

To solve a system of M equations of N unknowns using QR factorization, the following code will suffice assuming  $M \ge N$ .

```
! Solve the system: A*X = B in a least-squares sense, where A is an
! M-by-N matrix, B is an M-by-NRHS matrix, and X is an N-by-NRHS matrix.
real(dp), dimension(m, n) :: a
\text{real}(\text{dp}), \text{dimension}(\text{m, nrhs}) :: b, \text{qtb}
real(dp), dimension(n, nrhs) :: x
real(dp), dimension(n) :: tau
real(dp), dimension(m, m) :: q
! Initialize A and B...
! Compute the QR factorization. We're intentionally not forming the full
! Q matrix, but instead storing it in terms of its elementary reflector
! components in the sub-diagonal portions of A, and the corresponding
! scalar factors in TAU. Additionally, we'll let the algorithm allocate
 it's own workspace array; therefore, the call to factor A is:
call qr_factor(a, tau)
! Solve A \star X = B for X. The first N rows of B are used to store X.
call solve_qr(a, tau, b)
! Also note, we could form {\tt Q} and {\tt R} explicitly. Then solution of the
! system of equations can be found. First we form {\tt Q} and {\tt R}\text{.}
call form_qr(a, tau, q) ! Forms Q, and R is stored in A
 Since we now have Q and R, we seek a solution to the equation:
! Q*R*X = B, but Q is an orthogonal matrix (i.e. Q**T = inv(Q)). ! Then: R*X = Q**T * B, and R is upper triangular; therefore, back
 substitution will suffice for a solution procedure.
! Next, compute Q**T*B, and store in QTB.
call mtx_mult(.true., .false., 1.0d0, q, b, 0.0d0, qtb)
! Copy the first N rows of Q**T * B into X for the solution process.
```

```
! Notice, only the first N rows are needed as rows N+1:M are all zero in ! matrix R. x = \text{qtb}(1:n, nrhs) ! Compute the solution and store in X call solve_triangular_system(.true., .true., .false., .true., 1.0d0, & a(1:n,1:n), x)
```

This routine utilizes the LAPACK routine DGEQRF.

Definition at line 425 of file linalg\_factor.f90.

5.9.2.2 subroutine linalg\_factor::qr\_factor::qr\_factor\_pivot ( real(dp), dimension(:,:), intent(inout) a, real(dp), dimension(:), intent(out) tau, integer(i32), dimension(:), intent(inout) jpvt, real(dp), dimension(:), intent(out), optional, target work, integer(i32), intent(out), optional olwork, class(errors), intent(inout), optional, target err ) [private]

Computes the QR factorization of an M-by-N matrix with column pivoting such that A \* P = Q \* R.

### **Parameters**

in,out	а	On input, the M-by-N matrix to factor. On output, the elements on and above the diagonal contain the MIN(M, N)-by-N upper trapezoidal matrix R (R is upper triangular if $M \ge N$ ). The elements below the diagonal, along with the array tau, represent the orthogonal matrix Q as a product of elementary reflectors.
out	tau	A MIN(M, N)-element array used to store the scalar factors of the elementary reflectors.
in,out	jpvt	On input, an N-element array that if $JPVT(I)$ .ne. 0, the I-th column of A is permuted to the front of A $*$ P; if $JPVT(I) = 0$ , the I-th column of A is a free column. On output, if $JPVT(I) = K$ , then the I-th column of A $*$ P was the K-th column of A.
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.
out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for work, and returns without performing any actual calculations.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.</li> </ul>
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>

# Usage

To solve a system of M equations of N unknowns using QR factorization, the following code will suffice for any M and N.

```
! Solve the least-squares (M >= N), or the underdetermined (M < N) ! problem A*X = B, where A is an M-by-N matrix, B is an M-by-NRHS matrix, ! and X is an N-by-NRHS matrix. In the underdetermined case, or the ! case where the rank of matrix A is less than N, the solution obtained ! contains the fewest possible non-zero entries.
```

```
real(dp), dimension(m, n) :: a
real(dp), dimension(n, nrhs) :: b
real(dp), dimension(k) :: tau ! k = min(m, n)
\texttt{real(dp), dimension(m, m) :: } \texttt{q}
real(dp), dimension(n, n) :: p
integer(i32), dimension(n) :: pvt
! Initialize A and B...
! Allow all columns to be free.
pvt = 0
! Compute the QR factorization. We're intentionally not forming the full
! Q matrix, but instead storing it in terms of its elementary reflector
! components in the sub-diagonal portions of {\tt A}, and the corresponding
! scalar factors in TAU. Additionally, we'll let the algorithm allocate
! it's own workspace array; therefore, the call to factor {\tt A} is:
call qr_factor(a, tau, pvt)
! Solve A\star X = B for X. If M > N, the first N rows of B are used to store
 X. If M < N, the input matrix B must be N-by-NRHS, and only the first
! M rows are used for the actual matrix B. The remaining N\!-\!M rows
!\ \mbox{can} contain whatever as they are not referenced until they are
! overwritten by the N-by-NRHS solution matrix X.
call solve_gr(a, tau, pvt, b)
! Notice, if the explicit {\tt Q} matrix from the factorization is desired,
! the form_qr routine works similarly as in the no-pivot case;
! however, the permutation matrix P is also constructed. The call would
! be as follows. Also, as with the no-pivot algorithm, the matrix \ensuremath{\mathbf{R}} is
! stored in matrix A.
call form gr(a, tau, pvt, g, p)
! Solution can proceed as per typical, but with a full {\tt Q} matrix. Also
! note, the problem is of the form: A*P = Q*R. Solution is straight! forward, as with the no-pivot case; however, if M < N, then R is upper
 trapezoidal, and must be appropriately partitioned to solve. The rank
! of matrix r should be considered when applying the partition.
```

This routine utilizes the LAPACK routine DGEQP3.

Definition at line 579 of file linalg\_factor.f90.

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

/home/jason/Documents/Code/linalg/src/linalg\_factor.f90

# 5.10 linalg\_solve::solve\_cholesky Interface Reference

Solves a system of Cholesky factored equations.

# **Private Member Functions**

- subroutine solve\_cholesky\_mtx (upper, a, b, err)
   Solves a system of Cholesky factored equations.
- subroutine solve\_cholesky\_vec (upper, a, b, err)
   Solves a system of Cholesky factored equations.

# 5.10.1 Detailed Description

Solves a system of Cholesky factored equations.

Definition at line 83 of file linalg\_solve.f90.

# 5.10.2 Member Function/Subroutine Documentation

5.10.2.1 subroutine linalg\_solve::solve\_cholesky::solve\_cholesky\_mtx ( logical, intent(in) *upper*, real(dp), dimension(:,:), intent(in) *a*, real(dp), dimension(:,:), intent(inout) *b*, class(errors), intent(inout), optional, target *err* ) [private]

Solves a system of Cholesky factored equations.

### **Parameters**

in	upper	Set to true if the original matrix A was factored such that $A = U**T*U$ ; else, set to false if
		the factorization of A was $A = L**T*L$ .
in	а	The N-by-N Cholesky factored matrix.
in,out	b	On input, the N-by-NRHS right-hand-side matrix B. On output, the solution matrix X.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if any of the input array sizes are incorrect.</li> </ul>

### **Notes**

This routine utilizes the LAPACK routine DPOTRS.

Definition at line 1073 of file linalg\_solve.f90.

5.10.2.2 subroutine linalg\_solve::solve\_cholesky::solve\_cholesky\_vec ( logical, intent(in) *upper*, real(dp), dimension(:,:), intent(in) *a*, real(dp), dimension(:), intent(inout) *b*, class(errors), intent(inout), optional, target *err* ) [private]

Solves a system of Cholesky factored equations.

# **Parameters**

in	upper	Set to true if the original matrix A was factored such that $A = U**T*U$ ; else, set to false if
		the factorization of A was $A = L**T*L$ .
in	а	The N-by-N Cholesky factored matrix.
in,out	b	On input, the N-element right-hand-side vector B. On output, the solution vector X.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		LA_ARRAY_SIZE_ERROR: Occurs if any of the input array sizes are incorrect.

# Notes

This routine utilizes the LAPACK routine DPOTRS.

Definition at line 1139 of file linalg\_solve.f90.

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

/home/jason/Documents/Code/linalg/src/linalg\_solve.f90

# 5.11 linalg\_solve::solve\_least\_squares Interface Reference

Solves the overdetermined or underdetermined system (A\*X = B) of M equations of N unknowns.

# **Private Member Functions**

- subroutine solve\_least\_squares\_mtx (a, b, work, olwork, err)
  - Solves the overdetermined or underdetermined system (A\*X = B) of M equations of N unknowns using a QR or LQ factorization of the matrix A. Notice, it is assumed that matrix A has full rank.
- subroutine solve\_least\_squares\_vec (a, b, work, olwork, err)

Solves the overdetermined or underdetermined system (A\*X = B) of M equations of N unknowns using a QR or LQ factorization of the matrix A. Notice, it is assumed that matrix A has full rank.

# 5.11.1 Detailed Description

Solves the overdetermined or underdetermined system (A\*X = B) of M equations of N unknowns.

Definition at line 91 of file linalg\_solve.f90.

# 5.11.2 Member Function/Subroutine Documentation

5.11.2.1 subroutine linalg\_solve::solve\_least\_squares::solve\_least\_squares\_mtx ( real(dp), dimension(:,:), intent(inout) a, real(dp), dimension(:,:), intent(inout) b, real(dp), dimension(:), intent(out), optional, target work, integer(i32), intent(out), optional olwork, class(errors), intent(inout), optional, target err ) [private]

Solves the overdetermined or underdetermined system (A\*X = B) of M equations of N unknowns using a QR or LQ factorization of the matrix A. Notice, it is assumed that matrix A has full rank.

in,out	а	On input, the M-by-N matrix A. On output, if $M \ge N$ , the QR factorization of A in the form as output by qr_factor; else, if $M < N$ , the LQ factorization of A.
in,out	b	If $M>=N$ , the M-by-NRHS matrix B. On output, the first N rows contain the N-by-NRHS solution matrix X. If $M< N$ , an N-by-NRHS matrix with the first M rows containing the matrix B. On output, the N-by-NRHS solution matrix X.
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.
out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for work, and returns without performing any actual calculations.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.
Generated by Do	xygen	<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>
		LA_INVALID_OPERATION_ERROR: Occurs if a is not of full rank.

This routine utilizes the LAPACK routine DGELS.

Definition at line 1568 of file linalg solve.f90.

5.11.2.2 subroutine linalg\_solve::solve\_least\_squares::solve\_least\_squares\_vec ( real(dp), dimension(:,:), intent(inout) a, real(dp), dimension(:), intent(inout) b, real(dp), dimension(:), intent(out), optional, target work, integer(i32), intent(out), optional olwork, class(errors), intent(inout), optional, target err ) [private]

Solves the overdetermined or underdetermined system (A\*X = B) of M equations of N unknowns using a QR or LQ factorization of the matrix A. Notice, it is assumed that matrix A has full rank.

# **Parameters**

in,out	а	On input, the M-by-N matrix A. On output, if $M \ge N$ , the QR factorization of A in the form as output by qr_factor; else, if $M < N$ , the LQ factorization of A.
in,out	b	If $M >= N$ , the M-element array B. On output, the first N elements contain the N-element solution array X. If $M < N$ , an N-element array with the first M elements containing the array B. On output, the N-element solution array X.
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.
out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for work, and returns without performing any actual calculations.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.</li> </ul>
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>
		LA_INVALID_OPERATION_ERROR: Occurs if a is not of full rank.

# Notes

This routine utilizes the LAPACK routine DGELS.

Definition at line 1674 of file linalg\_solve.f90.

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

• /home/jason/Documents/Code/linalg/src/linalg\_solve.f90

# 5.12 linalg\_solve::solve\_least\_squares\_full Interface Reference

Solves the overdetermined or underdetermined system (A\*X = B) of M equations of N unknowns, but uses a full orthogonal factorization of the system.

# **Private Member Functions**

- subroutine solve\_least\_squares\_mtx\_pvt (a, b, ipvt, arnk, work, olwork, err)
  - Solves the overdetermined or underdetermined system (A\*X = B) of M equations of N unknowns using a complete orthogonal factorization of matrix A.
- subroutine solve\_least\_squares\_vec\_pvt (a, b, ipvt, arnk, work, olwork, err)
  - Solves the overdetermined or underdetermined system (A\*X = B) of M equations of N unknowns using a complete orthogonal factorization of matrix A.

# 5.12.1 Detailed Description

Solves the overdetermined or underdetermined system (A\*X = B) of M equations of N unknowns, but uses a full orthogonal factorization of the system.

Definition at line 100 of file linalg\_solve.f90.

# 5.12.2 Member Function/Subroutine Documentation

5.12.2.1 subroutine linalg\_solve::solve\_least\_squares\_full::solve\_least\_squares\_mtx\_pvt ( real(dp), dimension(:,:), intent(inout) a, real(dp), dimension(:,:), intent(inout) b, integer(i32), dimension(:), intent(inout), optional, target ipvt, integer(i32), intent(out), optional arnk, real(dp), dimension(:), intent(out), optional, target work, integer(i32), intent(out), optional olwork, class(errors), intent(inout), optional, target err ) [private]

Solves the overdetermined or underdetermined system (A\*X = B) of M equations of N unknowns using a complete orthogonal factorization of matrix A.

# **Parameters**

in,out	а	On input, the M-by-N matrix A. On output, the matrix is overwritten by the details of its complete orthogonal factorization.
in,out	b	If $M >= N$ , the M-by-NRHS matrix B. On output, the first N rows contain the N-by-NRHS solution matrix X. If $M < N$ , an N-by-NRHS matrix with the first M rows containing the matrix B. On output, the N-by-NRHS solution matrix X.
out	ipvt	An optional input that on input, an N-element array that if $IPVT(I)$ .ne. 0, the I-th column of A is permuted to the front of A $*$ P; if $IPVT(I) = 0$ , the I-th column of A is a free column. On output, if $IPVT(I) = K$ , then the I-th column of A $*$ P was the K-th column of A. If not supplied, memory is allocated internally, and $IPVT$ is set to all zeros such that all columns are treated as free.
out	arnk	An optional output, that if provided, will return the rank of a.
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.
out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for work, and returns without performing any actual calculations.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.</li> </ul>
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>

Generated by Doxygen

This routine utilizes the LAPACK routine DGELSY.

Definition at line 1786 of file linalg\_solve.f90.

5.12.2.2 subroutine linalg\_solve::solve\_least\_squares\_full::solve\_least\_squares\_vec\_pvt ( real(dp), dimension(:,:), intent(inout) a, real(dp), dimension(:), intent(inout) b, integer(i32), dimension(:), intent(inout), optional, target ipvt, integer(i32), intent(out), optional arnk, real(dp), dimension(:), intent(out), optional, target work, integer(i32), intent(out), optional olwork, class(errors), intent(inout), optional, target err ) [private]

Solves the overdetermined or underdetermined system (A\*X = B) of M equations of N unknowns using a complete orthogonal factorization of matrix A.

### **Parameters**

in,out	а	On input, the M-by-N matrix A. On output, the matrix is overwritten by the details of its complete orthogonal factorization.
in,out	b	If $M >= N$ , the M-element array B. On output, the first N elements contain the N-element solution array X. If $M < N$ , an N-element array with the first M elements containing the array B. On output, the N-element solution array X.
out	ipvt	An optional input that on input, an N-element array that if $IPVT(I)$ .ne. 0, the I-th column of A is permuted to the front of A $*$ P; if $IPVT(I) = 0$ , the I-th column of A is a free column. On output, if $IPVT(I) = K$ , then the I-th column of A $*$ P was the K-th column of A. If not supplied, memory is allocated internally, and $IPVT$ is set to all zeros such that all columns are treated as free.
out	arnk	An optional output, that if provided, will return the rank of a.
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.
out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for work, and returns without performing any actual calculations.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.</li> </ul>
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>

# Notes

This routine utilizes the LAPACK routine DGELSY.

Definition at line 1932 of file linalg\_solve.f90.

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

• /home/jason/Documents/Code/linalg/src/linalg\_solve.f90

# 5.13 linalg\_solve::solve\_least\_squares\_svd Interface Reference

Solves the overdetermined or underdetermined system (A\*X = B) of M equations of N unknowns using a singular value decomposition of matrix A.

### **Private Member Functions**

- subroutine solve\_least\_squares\_mtx\_svd (a, b, arnk, s, work, olwork, err)

  Solves the overdetermined or underdetermined system (A\*X = B) of M equations of N unknowns using a singular value decomposition of matrix A.
- subroutine solve\_least\_squares\_vec\_svd (a, b, arnk, s, work, olwork, err)

  Solves the overdetermined or underdetermined system (A\*X = B) of M equations of N unknowns using a singular value decomposition of matrix A.

# 5.13.1 Detailed Description

Solves the overdetermined or underdetermined system (A\*X = B) of M equations of N unknowns using a singular value decomposition of matrix A.

Definition at line 109 of file linalg\_solve.f90.

# 5.13.2 Member Function/Subroutine Documentation

5.13.2.1 subroutine linalg\_solve::solve\_least\_squares\_svd::solve\_least\_squares\_mtx\_svd ( real(dp), dimension(:,:), intent(inout) a, real(dp), dimension(:,:), intent(inout) b, integer(i32), intent(out), optional arnk, real(dp), dimension(:), intent(out) s, real(dp), dimension(:), intent(out), optional, target work, integer(i32), intent(out), optional olwork, class(errors), intent(inout), optional, target err ) [private]

Solves the overdetermined or underdetermined system (A\*X = B) of M equations of N unknowns using a singular value decomposition of matrix A.

in,out	а	On input, the M-by-N matrix A. On output, the matrix is overwritten by the details of its complete orthogonal factorization.
in,out	b	If $M >= N$ , the M-by-NRHS matrix B. On output, the first N rows contain the N-by-NRHS solution matrix X. If $M < N$ , an N-by-NRHS matrix with the first M rows containing the matrix B. On output, the N-by-NRHS solution matrix X.
out	arnk	An optional output, that if provided, will return the rank of a.
out	s	A MIN(M, N)-element array that on output contains the singular values of $a$ in descending order. Notice, the condition number of $a$ can be determined by S(1) / S(MIN(M, N)).
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.
out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for work, and returns without performing any actual calculations.

out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.</li> </ul>
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>
		<ul> <li>LA_CONVERGENCE_ERROR: Occurs as a warning if the QR iteration process could not converge to a zero value.</li> </ul>

# Notes

This routine utilizes the LAPACK routine DGELSS.

Definition at line 2076 of file linalg\_solve.f90.

5.13.2.2 subroutine linalg\_solve::solve\_least\_squares\_svd::solve\_least\_squares\_vec\_svd ( real(dp), dimension(:,:), intent(inout) a, real(dp), dimension(:), intent(inout) b, integer(i32), intent(out), optional arnk, real(dp), dimension(:), intent(out) s, real(dp), dimension(:), intent(out), optional, target work, integer(i32), intent(out), optional olwork, class(errors), intent(inout), optional, target err ) [private]

Solves the overdetermined or underdetermined system (A\*X = B) of M equations of N unknowns using a singular value decomposition of matrix A.

in,out	а	On input, the M-by-N matrix A. On output, the matrix is overwritten by the details of its complete orthogonal factorization.
in,out	b	If $M >= N$ , the M-by-NRHS matrix B. On output, the first N rows contain the N-by-NRHS solution matrix X. If $M < N$ , an N-by-NRHS matrix with the first M rows containing the matrix B. On output, the N-by-NRHS solution matrix X.
out	arnk	An optional output, that if provided, will return the rank of a.
out	s	A MIN(M, N)-element array that on output contains the singular values of $a$ in descending order. Notice, the condition number of $a$ can be determined by S(1) / S(MIN(M, N)).
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.
out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for work, and returns without performing any actual calculations.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.</li> </ul>
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>
		<ul> <li>LA_CONVERGENCE_ERROR: Occurs as a warning if the QR iteration process could not converge to a zero value.</li> </ul>

This routine utilizes the LAPACK routine DGELSS.

Definition at line 2204 of file linalg solve.f90.

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

/home/jason/Documents/Code/linalg/src/linalg\_solve.f90

# 5.14 linalg\_solve::solve\_lu Interface Reference

Solves a system of LU-factored equations.

### **Private Member Functions**

```
    subroutine solve_lu_mtx (a, ipvt, b, err)
    Solves a system of LU-factored equations.
```

subroutine solve\_lu\_vec (a, ipvt, b, err)
 Solves a system of LU-factored equations.

# 5.14.1 Detailed Description

Solves a system of LU-factored equations.

### Usage

To solve a system of N equations of N unknowns using LU factorization, the following code will suffice.

```
! Solve the system: A*X = B, where A is an N-by-N matrix, and B and X are
! N-by-NRHS in size.
! Variables
real(dp), dimension(n, n) :: a
real(dp), dimension(n, nrhs) :: b
! Define the array used to track row pivots.
integer(i32), dimension(n) :: pvt
! Initialize A and B...
! Compute the LU factorization of A. On output, A contains [L\U].
call lu_factor(a, pvt)
! Solve A*X = B for X - Note: X overwrites B.
call solve_lu(a, pvt, b)
```

### See Also

- Wikipedia
- Wolfram MathWorld

Definition at line 63 of file linalg\_solve.f90.

# 5.14.2 Member Function/Subroutine Documentation

5.14.2.1 subroutine linalg\_solve::solve\_lu::solve\_lu\_mtx ( real(dp), dimension(:,:), intent(in) a, integer(i32), dimension(:), intent(in) ipvt, real(dp), dimension(:,:), intent(inout) b, class(errors), intent(inout), optional, target err )

[private]

Solves a system of LU-factored equations.

in	а	The N-by-N LU factored matrix as output by lu_factor.
in	ipvt	The N-element pivot array as output by lu_factor.
in,out	b	On input, the N-by-NRHS right-hand-side matrix. On output, the N-by-NRHS solution matrix.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		LA_ARRAY_SIZE_ERROR: Occurs if any of the input array sizes are incorrect.

# Notes

The routine is based upon the LAPACK routine DGETRS.

Definition at line 351 of file linalg\_solve.f90.

5.14.2.2 subroutine linalg\_solve::solve\_lu::solve\_lu\_vec ( real(dp), dimension(:,:), intent(in) a, integer(i32), dimension(:), intent(in) ipvt, real(dp), dimension(:), intent(inout) b, class(errors), intent(inout), optional, target err ) [private]

Solves a system of LU-factored equations.

# **Parameters**

in	а	The N-by-N LU factored matrix as output by lu_factor.
in	ipvt	The N-element pivot array as output by lu_factor.
in,out	b	On input, the N-element right-hand-side array. On output, the N-element solution array.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if any of the input array sizes are incorrect.</li> </ul>

# Notes

The routine is based upon the LAPACK routine DGETRS.

Definition at line 412 of file linalg\_solve.f90.

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

• /home/jason/Documents/Code/linalg/src/linalg\_solve.f90

# 5.15 linalg\_solve::solve\_qr Interface Reference

Solves a system of M QR-factored equations of N unknowns.

### **Private Member Functions**

- subroutine solve\_qr\_no\_pivot\_mtx (a, tau, b, work, olwork, err)
  - Solves a system of M QR-factored equations of N unknowns where M >= N.
- subroutine solve\_qr\_no\_pivot\_vec (a, tau, b, work, olwork, err)
  - Solves a system of M QR-factored equations of N unknowns where M >= N.
- subroutine solve\_qr\_pivot\_mtx (a, tau, jpvt, b, work, olwork, err)
  - Solves a system of M QR-factored equations of N unknowns where the QR factorization made use of column pivoting.
- subroutine solve\_qr\_pivot\_vec (a, tau, jpvt, b, work, olwork, err)
  - Solves a system of M QR-factored equations of N unknowns where the QR factorization made use of column pivoting.

# 5.15.1 Detailed Description

Solves a system of M QR-factored equations of N unknowns.

### See Also

- Wikipedia
- LAPACK Users Manual

Definition at line 74 of file linalg\_solve.f90.

# 5.15.2 Member Function/Subroutine Documentation

5.15.2.1 subroutine linalg\_solve::solve\_qr\_no\_pivot\_mtx ( real(dp), dimension(:,:), intent(inout) *a*, real(dp), dimension(:), intent(inout) *b*, real(dp), dimension(:), intent(out), optional, target *work*, integer(i32), intent(out), optional *olwork*, class(errors), intent(inout), optional, target *err* ) [private]

Solves a system of M QR-factored equations of N unknowns where  $M \ge N$ .

in	а	On input, the M-by-N QR factored matrix as returned by qr_factor. On output, the contents of this matrix are restored. Notice, M must be greater than or equal to N.		
in	tau	A MIN(M, N)-element array containing the scalar factors of the elementary reflectors as returned by qr_factor.		
in	b	On input, the M-by-NRHS right-hand-side matrix. On output, the first N columns are overwritten by the solution matrix X.		
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.		
out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for work, and returns without performing any actual calculations.		
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.		
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.</li> </ul>		
Generated	by Doxygen	LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.		
	1			

This routine is based upon a subset of the LAPACK routine DGELS.

Definition at line 487 of file linalg solve.f90.

5.15.2.2 subroutine linalg\_solve::solve\_qr::solve\_qr\_no\_pivot\_vec ( real(dp), dimension(:,:), intent(inout) *a,* real(dp), dimension(:), intent(inout) *b,* real(dp), dimension(:), intent(out), optional, target work, integer(i32), intent(out), optional olwork, class(errors), intent(inout), optional, target err ) [private]

Solves a system of M QR-factored equations of N unknowns where  $M \ge N$ .

### **Parameters**

in	а	On input, the M-by-N QR factored matrix as returned by qr_factor. On output, the contents of this matrix are restored. Notice, M must be greater than or equal to N.
in	tau	A MIN(M, N)-element array containing the scalar factors of the elementary reflectors as returned by qr_factor.
in	b	On input, the M-element right-hand-side vector. On output, the first N elements are overwritten by the solution vector X.
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.
out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for work, and returns without performing any actual calculations.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.</li> </ul>
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>

### **Notes**

This routine is based upon a subset of the LAPACK routine DGELS.

Definition at line 602 of file linalg solve.f90.

5.15.2.3 subroutine linalg\_solve::solve\_qr::solve\_qr\_pivot\_mtx ( real(dp), dimension(:,:), intent(inout) a, real(dp), dimension(:), intent(in) tau, integer(i32), dimension(:), intent(in) jpvt, real(dp), dimension(:,:), intent(inout) b, real(dp), dimension(:), intent(out), optional, target work, integer(i32), intent(out), optional olwork, class(errors), intent(inout), optional, target err ) [private]

Solves a system of M QR-factored equations of N unknowns where the QR factorization made use of column pivoting.

	this matrix are altered.
	this matrix are altered.
tau	A MIN(M, N)-element array containing the scalar factors of the elementary reflectors as
	returned by qr_factor.
jpvt	An N-element array, as output by qr_factor, used to track the column pivots.
b	On input, the MAX(M, N)-by-NRHS matrix where the first M rows contain the right-hand-side
	matrix B. On output, the first N rows are overwritten by the solution matrix X.
work	An optional input, that if provided, prevents any local memory allocation. If not provided, the
	memory required is allocated within. If provided, the length of the array must be at least
	olwork.
olwork	An optional output used to determine workspace size. If supplied, the routine determines the
	optimal size for $\mathtt{work},$ and returns without performing any actual calculations.
err	An optional errors-based object that if provided can be used to retrieve information relating to
	any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning
	messages that may be encountered are as follows.
	<ul> <li>LA ARRAY SIZE ERROR: Occurs if any of the input arrays are not sized</li> </ul>
	appropriately.
	<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>
	jpvt b work

# Notes

This routine is based upon a subset of the LAPACK routine DGELSY.

Definition at line 715 of file linalg\_solve.f90.

5.15.2.4 subroutine linalg\_solve::solve\_qr::solve\_qr\_pivot\_vec ( real(dp), dimension(:,:), intent(inout) a, real(dp), dimension(:), intent(in) tau, integer(i32), dimension(:), intent(in) jpvt, real(dp), dimension(:), intent(inout) b, real(dp), dimension(:), intent(out), optional, target work, integer(i32), intent(out), optional olwork, class(errors), intent(inout), optional, target err ) [private]

Solves a system of M QR-factored equations of N unknowns where the QR factorization made use of column pivoting.

in	а	On input, the M-by-N QR factored matrix as returned by qr_factor. On output, the contents of this matrix are altered.
in	tau	A MIN(M, N)-element array containing the scalar factors of the elementary reflectors as returned by qr_factor.
in	jpvt	An N-element array, as output by qr_factor, used to track the column pivots.
in	b	On input, the MAX(M, N)-element array where the first M elements contain the right-hand-side vector B. On output, the first N elements are overwritten by the solution vector X.
out	work	An optional input, that if provided, prevents any local memory allocation. If not provided, the memory required is allocated within. If provided, the length of the array must be at least olwork.
out	olwork	An optional output used to determine workspace size. If supplied, the routine determines the optimal size for work, and returns without performing any actual calculations.

out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if any of the input arrays are not sized appropriately.</li> </ul>
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if local memory must be allocated, and there is insufficient memory available.</li> </ul>

### Notes

This routine is based upon a subset of the LAPACK routine DGELSY.

Definition at line 902 of file linalg\_solve.f90.

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

• /home/jason/Documents/Code/linalg/src/linalg\_solve.f90

# 5.16 linalg\_solve::solve\_triangular\_system Interface Reference

Solves a triangular system of equations.

# **Private Member Functions**

- subroutine solve\_tri\_mtx (Iside, upper, trans, nounit, alpha, a, b, err)

  Solves one of the matrix equations: op(A) \* X = alpha \* B, or X \* op(A) = alpha \* B, where A is a triangular matrix.
- subroutine solve\_tri\_vec (upper, trans, nounit, a, x, err)
   Solves the system of equations: op(A) \* X = B, where A is a triangular matrix.

# 5.16.1 Detailed Description

Solves a triangular system of equations.

Definition at line 29 of file linalg solve.f90.

# 5.16.2 Member Function/Subroutine Documentation

5.16.2.1 subroutine linalg\_solve::solve\_triangular\_system::solve\_tri\_mtx ( logical, intent(in) *lside*, logical, intent(in) *upper*, logical, intent(in) *trans*, logical, intent(in) *nounit*, real(dp), intent(in) *alpha*, real(dp), dimension(:,:), intent(in) *a*, real(dp), dimension(:,:), intent(inout) *b*, class(errors), intent(inout), optional, target *err* ) [private]

Solves one of the matrix equations: op(A) \* X = alpha \* B, or X \* op(A) = alpha \* B, where A is a triangular matrix.

in	Iside	Set to true to solve $op(A) * X = alpha * B$ ; else, set to false to solve $X * op(A) = alpha * B$ .
in	upper	Set to true if A is an upper triangular matrix; else, set to false if A is a lower triangular matrix.
in	trans	Set to true if $op(A) = A**T$ ; else, set to false if $op(A) = A$ .
in	nounit	Set to true if A is not a unit-diagonal matrix (ones on every diagonal element); else, set to false if A is a unit-diagonal matrix.
in	alpha	The scalar multiplier to B.
in	а	If lside is true, the M-by-M triangular matrix on which to operate; else, if lside is false, the N-by-N triangular matrix on which to operate.
in,out	b	On input, the M-by-N right-hand-side. On output, the M-by-N solution.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if a is not square, or if the sizes of a and b are not compatible.

# Usage

To solve a triangular system of N equations of N unknowns A\*X = B, where A is an N-by-N upper triangular matrix, and B and X are N-by-NRHS matrices, the following code will suffice.

# Notes

This routine is based upon the BLAS routine DTRSM.

Definition at line 169 of file linalg\_solve.f90.

5.16.2.2 subroutine linalg\_solve::solve\_triangular\_system::solve\_tri\_vec ( logical, intent(in) upper, logical, intent(in) trans, logical, intent(in) nounit, real(dp), dimension(:,:), intent(in) a, real(dp), dimension(:), intent(inout) x, class(errors), intent(inout), optional, target err ) [private]

Solves the system of equations: op(A) \* X = B, where A is a triangular matrix.

in	upper	Set to true if A is an upper triangular matrix; else, set to false if A is a lower triangular matrix.
in	trans	Set to true if $op(A) = A**T$ ; else, set to false if $op(A) = A$ .

in	nounit	Set to true if A is not a unit-diagonal matrix (ones on every diagonal element); else, set to false if A is a unit-diagonal matrix.
in	а	The N-by-N triangular matrix.
in,out	X	On input, the N-element right-hand-side array. On output, the N-element solution array.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if a is not square, or if the sizes of a and b are
		not compatible.

# Usage

To solve a triangular system of N equations of N unknowns A\*X = B, where A is an N-by-N upper triangular matrix, and B and X are N-element arrays, the following code will suffice.

```
! Solve the system: A*X = B, where A is an upper triangular N-by-N
! matrix, and B and X are N-elements in size.
! Variables
integer(i32) :: info
real(dp), dimension(n, n) :: a
real(dp), dimension(n) :: b
! Initialize A and B...
! Solve A*X = B for X - Note: X overwrites B.
call solve_triangular_system(.true., .false., a, b)
```

# Notes

This routine is based upon the BLAS routine DTRSV.

Definition at line 275 of file linalg\_solve.f90.

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

/home/jason/Documents/Code/linalg/src/linalg\_solve.f90

# 5.17 linalg\_sorting::sort Interface Reference

Sorts an array.

# **Private Member Functions**

• subroutine sort\_dbl\_array (x, ascend)

Sorts an array.

• subroutine sort\_dbl\_array\_ind (x, ind, ascend, err)

Sorts an array.

subroutine sort\_cmplx\_array (x, ascend)

Sorts an array.

subroutine sort\_cmplx\_array\_ind (x, ind, ascend, err)

Sorts an array.

subroutine sort\_eigen\_cmplx (vals, vecs, ascend, err)

A sorting routine specifically tailored for sorting of eigenvalues and their associated eigenvectors using a quick-sort approach.

subroutine sort\_eigen\_dbl (vals, vecs, ascend, err)

A sorting routine specifically tailored for sorting of eigenvalues and their associated eigenvectors using a quick-sort approach.

# 5.17.1 Detailed Description

Sorts an array.

Definition at line 18 of file linalg\_sorting.f90.

# 5.17.2 Member Function/Subroutine Documentation

5.17.2.1 subroutine linalg\_sorting::sort::sort\_cmplx\_array ( complex(dp), dimension(:), intent(inout) x, logical, intent(in), optional ascend ) [private]

Sorts an array.

# **Parameters**

in,out	Х	On input, the array to sort. On output, the sorted array.
in	ascend	An optional input that, if specified, controls if the the array is sorted in an ascending order
		(default), or a descending order.

### Remarks

This routine utilizes a quick sort algorithm. As this routine operates on complex valued items, the complex values are sorted based upon the real component of the number.

# Notes

This implementation is a slight modification of the code presented at  $http://www.fortran. \leftarrow com/qsort_c.f95$ .

Definition at line 156 of file linalg\_sorting.f90.

5.17.2.2 subroutine linalg\_sorting::sort::sort\_cmplx\_array\_ind ( complex(dp), dimension(:), intent(inout) x, integer(i32), dimension(:), intent(inout) ind, logical, intent(in), optional ascend, class(errors), intent(inout), optional, target err )

[private]

# Sorts an array.

### **Parameters**

in,out	X	On input, the array to sort. On output, the sorted array.
in,out	ind	On input, an integer array. On output, the contents of this array are shifted in the same order as that of $x$ as a means of tracking the sorting operation. It is often useful to set this array to an ascending group of values $(1, 2, n)$ such that this array tracks the original positions of the sorted array. Such an array can then be used to align other arrays. This array must be the same size as $x$ .
in	ascend	An optional input that, if specified, controls if the the array is sorted in an ascending order (default), or a descending order.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if ind is not sized to match x.

# Remarks

This routine utilizes a quick sort algorithm. As this routine operates on complex valued items, the complex values are sorted based upon the real component of the number.

# Notes

This implementation is a slight modification of the code presented at  $http://www.fortran. \leftarrow com/qsort_c.f95$ .

Definition at line 205 of file linalg\_sorting.f90.

5.17.2.3 subroutine linalg\_sorting::sort\_dbl\_array ( real(dp), dimension(:), intent(inout) x, logical, intent(in), optional ascend ) [private]

# Sorts an array.

# **Parameters**

in,out	X	On input, the array to sort. On output, the sorted array.
in	ascend	An optional input that, if specified, controls if the the array is sorted in an ascending order
		(default), or a descending order.

# Remarks

The routine utilizes a quick sort algorithm unless the size of the array is less than or equal to 20. For such small arrays an insertion sort algorithm is utilized.

This routine utilizes the LAPACK routine DLASRT.

Definition at line 47 of file linalg\_sorting.f90.

5.17.2.4 subroutine linalg\_sorting::sort::sort\_dbl\_array\_ind ( real(dp), dimension(:), intent(inout) x, integer(i32), dimension(:), intent(inout) ind, logical, intent(in), optional ascend, class(errors), intent(inout), optional, target err ) [private]

Sorts an array.

#### **Parameters**

in,out	x	On input, the array to sort. On output, the sorted array.
in,out	ind	On input, an integer array. On output, the contents of this array are shifted in the same order as that of $x$ as a means of tracking the sorting operation. It is often useful to set this array to an ascending group of values $(1, 2, n)$ such that this array tracks the original positions of the sorted array. Such an array can then be used to align other arrays. This array must be the same size as $x$ .
in	ascend	An optional input that, if specified, controls if the the array is sorted in an ascending order (default), or a descending order.
out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.  • LA_ARRAY_SIZE_ERROR: Occurs if ind is not sized to match x.

### Remarks

This routine utilizes a quick sort algorithm explained at http://www.fortran.com/qsort\_c.f95.

Definition at line 97 of file linalg\_sorting.f90.

5.17.2.5 subroutine linalg\_sorting::sort::sort\_eigen\_cmplx ( complex(dp), dimension(:), intent(inout) vals, complex(dp), dimension(:,:), intent(inout) vecs, logical, intent(in), optional ascend, class(errors), intent(inout), optional, target err )

[private]

A sorting routine specifically tailored for sorting of eigenvalues and their associated eigenvectors using a quick-sort approach.

in,out	vals	On input, an N-element array containing the eigenvalues. On output, the sorted eigenvalues.
in,out	vecs	On input, an N-by-N matrix containing the eigenvectors associated with $vals$ (one vector per column). On output, the sorted eigenvector matrix.
in	ascend	An optional input that, if specified, controls if the the array is sorted in an ascending order (default), or a descending order.

out	err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
		LA_ARRAY_SIZE_ERROR: Occurs if vecs is not sized to match vals.
		<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if there is insufficient memory available to comoplete this operation.</li> </ul>

Definition at line 267 of file linalg\_sorting.f90.

5.17.2.6 subroutine linalg\_sorting::sort::sort\_eigen\_dbl ( real(dp), dimension(:), intent(inout) vals, real(dp), dimension(:,:), intent(inout) vecs, logical, intent(in), optional ascend, class(errors), intent(inout), optional, target err )

[private]

A sorting routine specifically tailored for sorting of eigenvalues and their associated eigenvectors using a quick-sort approach.

# **Parameters**

vals	On input, an N-element array containing the eigenvalues. On output, the sorted eigenvalues.
vecs	On input, an N-by-N matrix containing the eigenvectors associated with $vals$ (one vector per column). On output, the sorted eigenvector matrix.
ascend	An optional input that, if specified, controls if the the array is sorted in an ascending order (default), or a descending order.
err	An optional errors-based object that if provided can be used to retrieve information relating to any errors encountered during execution. If not provided, a default implementation of the errors class is used internally to provide error handling. Possible errors and warning messages that may be encountered are as follows.
	<ul> <li>LA_ARRAY_SIZE_ERROR: Occurs if vecs is not sized to match vals.</li> </ul>
	<ul> <li>LA_OUT_OF_MEMORY_ERROR: Occurs if there is insufficient memory available to comoplete this operation.</li> </ul>
	vecs

Definition at line 345 of file linalg\_sorting.f90.

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

• /home/jason/Documents/Code/linalg/src/linalg\_sorting.f90

# Index

cholesky_factor	eigen_asymm
linalg_factor, 44	linalg_eigen, 39
cholesky_factor_c	linalg_eigen::eigen, 85
linalg_c_binding, 9	eigen_asymm_c
cholesky_rank1_downdate	linalg_c_binding, 14
linalg_factor, 44	eigen_gen
cholesky_rank1_downdate_c	linalg_eigen, 40
linalg_c_binding, 10	linalg_eigen::eigen, 86
cholesky_rank1_update	eigen_gen_c
linalg_factor, 45	linalg_c_binding, 15
cholesky_rank1_update_c	eigen_symm
linalg_c_binding, 10	linalg_eigen, 41
cmplx_partition	linalg_eigen::eigen, 87
linalg_sorting, 75	eigen_symm_c
cmplx_partition_ind	linalg_c_binding, 15
linalg_sorting, 75	<u> </u>
cmtx_mult_c	form_lu_all
linalg_c_binding, 11	linalg_factor, 46
inalg_c_binding, 11	linalg_factor::form_lu, 89
dbl partition ind	form_lu_c
dbl_partition_ind	linalg_c_binding, 16
linalg_sorting, 75	form_lu_only
det	linalg_factor, 46
linalg_core, 31	linalg_factor::form_lu, 89
det_c	form_qr_c
linalg_c_binding, 11	linalg_c_binding, 16
diag_cmtx_mult_left_c	form_qr_no_pivot
linalg_c_binding, 12	linalg_factor, 47
diag_cmtx_mult_right_c	linalg_factor::form_qr, 90
linalg_c_binding, 12	form_qr_pivot
diag_mtx_mult_cmplx_left_c	linalg_factor, 47
linalg_c_binding, 13	linalg_factor::form_qr, 91
diag_mtx_mult_cmplx_right_c	form_qr_pivot_c
linalg_c_binding, 13	linalg_c_binding, 17
diag_mtx_mult_left_c	
linalg_c_binding, 13	lapack, 7
diag_mtx_mult_mtx	lapack::DLAMCH, 84
linalg_core, 32	linalg_c_binding, 7
linalg_core::diag_mtx_mult, 81	cholesky_factor_c, 9
diag_mtx_mult_mtx2	cholesky_rank1_downdate_c, 10
linalg_core, 33	cholesky_rank1_update_c, 10
linalg_core::diag_mtx_mult, 82	cmtx_mult_c, 11
diag_mtx_mult_mtx3	det_c, 11
linalg_core, 33	diag_cmtx_mult_left_c, 12
linalg_core::diag_mtx_mult, 82	diag_cmtx_mult_right_c, 12
diag_mtx_mult_mtx4	diag_mtx_mult_cmplx_left_c, 13
linalg_core, 34	diag_mtx_mult_cmplx_right_c, 13
linalg_core::diag_mtx_mult, 83	diag_mtx_mult_left_c, 13
disg_mtx_mult_right_c	disg_mtx_mult_right_c, 14
linalg_c_binding, 14	eigen_asymm_c, 14

122 INDEX

eigen_gen_c, 15	eigen_asymm, 85
eigen_symm_c, 15	eigen_gen, 86
form_lu_c, 16	eigen_symm, 87
form_qr_c, 16	linalg_factor, 42
form_qr_pivot_c, 17	cholesky_factor, 44
lu_factor_c, 18	cholesky_rank1_downdate, 44
mtx_inverse_c, 18	cholesky_rank1_update, 45
mtx_mult_c, 19	form_lu_all, 46
mtx_pinverse_c, 19	form_lu_only, 46
mtx_rank_c, 20	form_qr_no_pivot, 47
mult_qr_c, 20	form_qr_pivot, 47
mult_rz_c, 21	lu_factor, 48
qr_factor_c, 21	mult_qr_mtx, 49
qr_factor_pivot_c, 22	mult_qr_vec, 50
qr_rank1_update_c, 23	mult_rz_mtx, 51
rank1_update_c, 23	mult_rz_vec, 52
rz_factor_c, 23	qr_factor_no_pivot, 52
solve_cholesky_c, 24	qr_factor_pivot, 54
solve_least_squares_c, 24	qr_rank1_update, 55
solve_lu_c, 25	rz_factor, 56
solve_qr_c, 25	svd, 57
solve_qr_pivot_c, 26	linalg_factor::form_lu, 88
solve_tri_mtx_c, 26	form_lu_all, 89
sort_cmplx_ind_c, 27	form_lu_only, 89
sort_dbl_ind_c, 27	linalg_factor::form_qr, 90
sort_eigen_cmplx_c, 28	form_qr_no_pivot, 90
sort_eigen_dbl_c, 28	form_qr_pivot, 91
svd_c, 28	linalg_factor::mult_qr, 94
swap_c, 29	mult_qr_mtx, 94
trace_c, 29	mult_qr_vec, 95
linalg_constants, 30	linalg_factor::mult_rz, 96
linalg_core, 30	mult_rz_mtx, 96
det, 31	mult_rz_vec, 97
diag_mtx_mult_mtx, 32	linalg_factor::qr_factor, 98
diag_mtx_mult_mtx2, 33	qr_factor_no_pivot, 98
diag_mtx_mult_mtx3, 33	qr_factor_pivot, 100
diag_mtx_mult_mtx4, 34	linalg_solve, 59
mtx_mult_mtx, 34	mtx_inverse, 60
mtx_mult_vec, 35	mtx_pinverse, 61
mtx_rank, 36	solve_cholesky_mtx, 62
rank1_update, 36	solve_cholesky_vec, 63
recip_mult_array, 37	solve_least_squares_mtx, 63
swap, 37	solve_least_squares_mtx_pvt, 64
trace, 38	solve_least_squares_mtx_svd, 65
tri_mtx_mult, 38	solve_least_squares_vec, 66
linalg_core::diag_mtx_mult, 81	solve_least_squares_vec_pvt, 66
diag_mtx_mult_mtx, 81	solve_least_squares_vec_svd, 67
diag_mtx_mult_mtx2, 82	solve_lu_mtx, 68
diag_mtx_mult_mtx3, 82	solve_lu_vec, 68
diag_mtx_mult_mtx4, 83	solve_qr_no_pivot_mtx, 69
linalg_core::mtx_mult, 92	solve_qr_no_pivot_vec, 70
mtx_mult_mtx, 92	solve_qr_pivot_mtx, 70
mtx_mult_vec, 93	solve_qr_pivot_vec, 71
linalg_eigen, 38	solve_tri_mtx, 72
eigen_asymm, 39	solve_tri_vec, 73
eigen_gen, 40	linalg_solve::solve_cholesky, 101
eigen_symm, 41	solve_cholesky_mtx, 102
linalg_eigen::eigen, 84	solve_cholesky_vec, 102

INDEX 123

linalg_solve::solve_least_squares, 103	mtx_pinverse_c
solve_least_squares_mtx, 103	linalg_c_binding, 19
solve_least_squares_vec, 104	mtx_rank
linalg_solve::solve_least_squares_full, 104	linalg_core, 36
solve_least_squares_mtx_pvt, 105	mtx_rank_c
solve_least_squares_vec_pvt, 106	linalg_c_binding, 20
linalg_solve::solve_least_squares_svd, 107	mult_qr_c
solve_least_squares_mtx_svd, 107	linalg_c_binding, 20
solve_least_squares_vec_svd, 108	mult_qr_mtx
linalg_solve::solve_lu, 109	linalg_factor, 49
solve_lu_mtx, 109	linalg_factor::mult_qr, 94
solve_lu_vec, 110	mult_qr_vec
linalg_solve::solve_qr, 110	linalg_factor, 50
solve_gr_no_pivot_mtx, 111	linalg_factor::mult_qr, 95
solve_qr_no_pivot_vec, 112 solve gr pivot mtx, 112	mult_rz_c
solve_qr_pivot_mix, 112 solve gr pivot vec, 113	linalg_c_binding, 21 mult_rz_mtx
linalg_solve::solve_triangular_system, 114	linalg factor, 51
solve_tri_mtx, 114	linalg_lactor::mult_rz, 96
solve_tri_vec, 115	mult_rz_vec
linalg_sorting, 74	linalg_factor, 52
cmplx_partition, 75	linalg_factor::mult_rz, 97
cmplx_partition_ind, 75	iiiaig_iactoriiiuit_12, 97
dbl_partition_ind, 75	qr_factor_c
qsort_cmplx, 76	linalg_c_binding, 21
qsort_cmplx_ind, 76	qr_factor_no_pivot
qsort_dbl_ind, 77	linalg_factor, 52
sort_cmplx_array, 77	linalg_factor::qr_factor, 98
sort_cmplx_array_ind, 78	qr_factor_pivot
sort_dbl_array, 78	linalg_factor, 54
sort_dbl_array_ind, 79	linalg_factor::qr_factor, 100
sort_eigen_cmplx, 79	qr_factor_pivot_c
sort eigen dbl, 80	linalg_c_binding, 22
linalg_sorting::sort, 116	qr_rank1_update
sort_cmplx_array, 117	linalg_factor, 55
sort_cmplx_array_ind, 117	qr_rank1_update_c
sort_dbl_array, 118	linalg_c_binding, 23
sort_dbl_array_ind, 119	gsort cmplx
sort_eigen_cmplx, 119	linalg_sorting, 76
sort_eigen_dbl, 120	qsort_cmplx_ind
lu factor	linalg_sorting, 76
linalg_factor, 48	qsort_dbl_ind
lu_factor_c	linalg_sorting, 77
linalg_c_binding, 18	
	rank1_update
mtx_inverse	linalg_core, 36
linalg_solve, 60	rank1_update_c
mtx_inverse_c	linalg_c_binding, 23
linalg_c_binding, 18	recip_mult_array
mtx_mult_c	linalg_core, 37
linalg_c_binding, 19	rz_factor
mtx_mult_mtx	linalg_factor, 56
linalg_core, 34	rz_factor_c
linalg_core::mtx_mult, 92	linalg_c_binding, 23
mtx_mult_vec	
linalg_core, 35	solve_cholesky_c
linalg_core::mtx_mult, 93	linalg_c_binding, 24
mtx_pinverse	solve_cholesky_mtx
linalg_solve, 61	linalg_solve, 62

124 INDEX

linalg_solve::solve_cholesky, 102	linalg sorting::sort, 117
solve_cholesky_vec	sort_cmplx_array_ind
linalg solve, 63	linalg sorting, 78
linalg_solve::solve_cholesky, 102	linalg_sorting::sort, 117
solve_least_squares_c	sort_cmplx_ind_c
linalg_c_binding, 24	linalg_c_binding, 27
solve_least_squares_mtx	sort_dbl_array
linalg_solve, 63	linalg_sorting, 78
linalg_solve::solve_least_squares, 103	linalg_sorting::sort, 118
solve_least_squares_mtx_pvt	sort_dbl_array_ind
linalg_solve, 64	linalg_sorting, 79
linalg_solve::solve_least_squares_full, 105	linalg_sorting::sort, 119
solve_least_squares_mtx_svd	sort_dbl_ind_c
linalg_solve, 65	linalg_c_binding, 27
linalg_solve::solve_least_squares_svd, 107	sort_eigen_cmplx
solve_least_squares_vec	linalg_sorting, 79
linalg_solve, 66	linalg_sorting::sort, 119
linalg_solve; solve_least_squares, 104	sort_eigen_cmplx_c
solve_least_squares_vec_pvt	linalg_c_binding, 28
linalg_solve, 66	sort eigen dbl
<del>-</del>	linalg_sorting, 80
linalg_solve::solve_least_squares_full, 106	linalg_sorting::sort, 120
solve_least_squares_vec_svd	sort_eigen_dbl_c
linalg_solve, 67	linalg_c_binding, 28
linalg_solve::solve_least_squares_svd, 108	svd
solve_lu_c	linalg_factor, 57
linalg_c_binding, 25	svd c
solve_lu_mtx	linalg_c_binding, 28
linalg_solve, 68	swap
linalg_solve::solve_lu, 109	linalg_core, 37
solve_lu_vec	swap_c
linalg_solve, 68	linalg_c_binding, 29
linalg_solve::solve_lu, 110	<u>g_</u> g,
solve_qr_c	trace
linalg_c_binding, 25	linalg_core, 38
solve_qr_no_pivot_mtx	trace_c
linalg_solve, 69	linalg_c_binding, 29
linalg_solve::solve_qr, 111	tri_mtx_mult
solve_qr_no_pivot_vec	
_ ·	linalg_core, 38
linalg_solve, 70	
linalg_solve, 70 linalg_solve::solve_qr, 112	
linalg_solve, 70 linalg_solve::solve_qr, 112 solve_qr_pivot_c	
linalg_solve, 70 linalg_solve::solve_qr, 112 solve_qr_pivot_c linalg_c_binding, 26	
linalg_solve, 70 linalg_solve::solve_qr, 112 solve_qr_pivot_c linalg_c_binding, 26 solve_qr_pivot_mtx	
linalg_solve, 70 linalg_solve::solve_qr, 112 solve_qr_pivot_c linalg_c_binding, 26 solve_qr_pivot_mtx linalg_solve, 70	
linalg_solve, 70 linalg_solve::solve_qr, 112 solve_qr_pivot_c linalg_c_binding, 26 solve_qr_pivot_mtx linalg_solve, 70 linalg_solve::solve_qr, 112	
linalg_solve, 70 linalg_solve::solve_qr, 112 solve_qr_pivot_c linalg_c_binding, 26 solve_qr_pivot_mtx linalg_solve, 70 linalg_solve::solve_qr, 112 solve_qr_pivot_vec	
linalg_solve, 70 linalg_solve::solve_qr, 112 solve_qr_pivot_c linalg_c_binding, 26 solve_qr_pivot_mtx linalg_solve, 70 linalg_solve::solve_qr, 112 solve_qr_pivot_vec linalg_solve, 71	
linalg_solve, 70 linalg_solve::solve_qr, 112 solve_qr_pivot_c linalg_c_binding, 26 solve_qr_pivot_mtx linalg_solve, 70 linalg_solve::solve_qr, 112 solve_qr_pivot_vec linalg_solve, 71 linalg_solve::solve_qr, 113	
linalg_solve, 70 linalg_solve::solve_qr, 112 solve_qr_pivot_c linalg_c_binding, 26 solve_qr_pivot_mtx linalg_solve, 70 linalg_solve::solve_qr, 112 solve_qr_pivot_vec linalg_solve, 71 linalg_solve::solve_qr, 113 solve_tri_mtx	
linalg_solve, 70 linalg_solve::solve_qr, 112 solve_qr_pivot_c linalg_c_binding, 26 solve_qr_pivot_mtx linalg_solve, 70 linalg_solve::solve_qr, 112 solve_qr_pivot_vec linalg_solve, 71 linalg_solve::solve_qr, 113 solve_tri_mtx linalg_solve, 72	
linalg_solve, 70 linalg_solve::solve_qr, 112 solve_qr_pivot_c linalg_c_binding, 26 solve_qr_pivot_mtx linalg_solve, 70 linalg_solve::solve_qr, 112 solve_qr_pivot_vec linalg_solve, 71 linalg_solve::solve_qr, 113 solve_tri_mtx linalg_solve, 72 linalg_solve::solve_triangular_system, 114	
linalg_solve, 70 linalg_solve::solve_qr, 112 solve_qr_pivot_c linalg_c_binding, 26 solve_qr_pivot_mtx linalg_solve, 70 linalg_solve::solve_qr, 112 solve_qr_pivot_vec linalg_solve, 71 linalg_solve::solve_qr, 113 solve_tri_mtx linalg_solve, 72 linalg_solve::solve_triangular_system, 114 solve_tri_mtx_c	
linalg_solve, 70 linalg_solve::solve_qr, 112 solve_qr_pivot_c linalg_c_binding, 26 solve_qr_pivot_mtx linalg_solve, 70 linalg_solve::solve_qr, 112 solve_qr_pivot_vec linalg_solve, 71 linalg_solve::solve_qr, 113 solve_tri_mtx linalg_solve, 72 linalg_solve::solve_triangular_system, 114 solve_tri_mtx_c linalg_c_binding, 26	
linalg_solve, 70 linalg_solve::solve_qr, 112 solve_qr_pivot_c linalg_c_binding, 26 solve_qr_pivot_mtx linalg_solve, 70 linalg_solve::solve_qr, 112 solve_qr_pivot_vec linalg_solve, 71 linalg_solve::solve_qr, 113 solve_tri_mtx linalg_solve, 72 linalg_solve::solve_triangular_system, 114 solve_tri_mtx_c linalg_c_binding, 26 solve_tri_vec	
linalg_solve, 70 linalg_solve::solve_qr, 112 solve_qr_pivot_c linalg_c_binding, 26 solve_qr_pivot_mtx linalg_solve, 70 linalg_solve::solve_qr, 112 solve_qr_pivot_vec linalg_solve, 71 linalg_solve::solve_qr, 113 solve_tri_mtx linalg_solve, 72 linalg_solve::solve_triangular_system, 114 solve_tri_mtx_c linalg_c_binding, 26 solve_tri_vec linalg_solve, 73	
linalg_solve, 70 linalg_solve::solve_qr, 112 solve_qr_pivot_c linalg_c_binding, 26 solve_qr_pivot_mtx linalg_solve, 70 linalg_solve::solve_qr, 112 solve_qr_pivot_vec linalg_solve::solve_qr, 113 solve_tri_mtx linalg_solve::solve_triangular_system, 114 solve_tri_mtx_c linalg_c_binding, 26 solve_tri_vec linalg_solve, 73 linalg_solve::solve_triangular_system, 115	
linalg_solve, 70 linalg_solve::solve_qr, 112 solve_qr_pivot_c linalg_c_binding, 26 solve_qr_pivot_mtx linalg_solve, 70 linalg_solve::solve_qr, 112 solve_qr_pivot_vec linalg_solve, 71 linalg_solve::solve_qr, 113 solve_tri_mtx linalg_solve, 72 linalg_solve::solve_triangular_system, 114 solve_tri_mtx_c linalg_c_binding, 26 solve_tri_vec linalg_solve, 73	