Bloch-Messiah reduction on a two source HOM Dip

Generated by Doxygen 1.8.11

# **Contents**

1	Tod	o List			1
2	Mod	lules Ind	dex		3
	2.1	Module	es List		3
3	File	Index			5
	3.1	File Lis	st		5
4	Mod	lule Doc	cumentati	on	7
	4.1	makeo	pticalelem	nents Module Reference	7
		4.1.1	Detailed	Description	7
		4.1.2	Function	/Subroutine Documentation	7
			4.1.2.1	abt(i, j, ft, nspec)	7
			4.1.2.2	alloc_temparrays(nspace, nspec)	8
			4.1.2.3	amp(a)	8
			4.1.2.4	bbd(i, j, ft, nspec)	8
			4.1.2.5	dealloc_temparrays	9
			4.1.2.6	g4(ft, nspec)	9
			4.1.2.7	make_bs(nspace, nspec, symp_mat, m1, m2, theta)	9
			4.1.2.8	make_sq(nspace, nspec, symp_mat, m1, m2, alpha, beta)	9
		4.1.3	Variable	Documentation	10
			4.1.3.1	ident	10
	4.2	olis_f9	Ostdlib Mo	odule Reference	10
		4.2.1	Function	/Subroutine Documentation	10
			4.2.1.1	alloc_complex_eigenvects(matrix, eigenvals, u, v)	10

iv CONTENTS

			4.2.1.2	alloc_complex_svd(matrix, sigma, u, vt)	11
			4.2.1.3	c_identity(n)	11
			4.2.1.4	c_inv2(m_in)	11
			4.2.1.5	complex_eigenvects(a, w, vl, vr)	11
			4.2.1.6	complex_svd(a, sigma, u, vt)	12
			4.2.1.7	complextrace(a)	12
			4.2.1.8	expmatrix(matrix, n)	12
			4.2.1.9	factorial(n)	13
			4.2.1.10	$matrixmul(x,n)\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$	13
			4.2.1.11	matrixnorm(c)	13
			4.2.1.12	outerproduct(a, b)	13
			4.2.1.13	printvectors(vect, desc, f)	13
			4.2.1.14	randseed(seed)	13
			4.2.1.15	tprod(a, b)	14
		4.2.2	Variable l	Documentation	14
			4.2.2.1	imaginary	14
			4.2.2.2	pi	14
5	File	Docume	entation		15
	5.1	makeo	pticalelem	ents.f90 File Reference	15
	5.2	num_h	om.f90 File	e Reference	16
		5.2.1	Function/	Subroutine Documentation	16
			5.2.1.1	f(w1, w2, sig)	16
			5.2.1.2	gen_jsa(w1_start, w1_end, w1_incr, w2_start, w2_end, w2_incr, sigma, outfile) .	16
			5.2.1.3	make_squeezer(mode1, mode2, jsa)	16
			5.2.1.4	num_hom	17
	5.3	olis_f9	0stdlib.f90	File Reference	17
Ind	ex				19

# **Todo List**

```
Subprogram makeopticalelements::abt (i, j, ft, nspec)
check this

Subprogram makeopticalelements::bbd (i, j, ft, nspec)
check this
```

2 Todo List

# **Modules Index**

# 2.1 Modules List

Here is a list of all modules with brief descriptions:

makeopticalelements	
Module for building symplectic matrices for optical elements	7
olis f90stdlib	10

4 Modules Index

# File Index

# 3.1 File List

Here is a list of all files with brief descriptions:

makeopticalelements.f90	15
num_hom.f90	16
olis f90stdlib.f90	17

6 File Index

# **Module Documentation**

# 4.1 makeoptical elements Module Reference

module for building symplectic matrices for optical elements

#### **Functions/Subroutines**

- subroutine make\_bs (nspace, nspec, symp\_mat, m1, m2, theta)
   makes beamsplitter symplectic matrix takes in an allocated matrix for the beamsplitter matrix to be written to uses the private ident\_spec, spatial\_work, n\_work arrays
- subroutine make\_sq (nspace, nspec, symp\_mat, m1, m2, alpha, beta)
  - make symplectic squeezing matrix from exponetiated JSA a lot is broken...
- real(kind=dp) function g4 (ft, nspec)
  - calculates g4 using matrix elements sum
- real(kind=dp) function amp (a)
  - returns the absolute value squared |a|\*\*2
- complex(kind=dp) function abt (i, j, ft, nspec)
  - calculates matrix elements Alpha-Beta\*\*T for M = (A B ) (B\* A\*) computes AB\*\*T and returns the i,j-th element
- complex(kind=dp) function bbd (i, j, ft, nspec)
  - calculates the matrix elements Beta\*Beta\*\*H for  $M = (A\ B\ )$   $(B*\ A*)$  computes B\*B\*\*H (Hermitian conjg) and returns the i,j-th element
- subroutine alloc\_temparrays (nspace, nspec)
  - allocates temp arrays for matrices
- subroutine dealloc\_temparrays

#### **Variables**

real(kind=dp), public ident

# 4.1.1 Detailed Description

module for building symplectic matrices for optical elements

### 4.1.2 Function/Subroutine Documentation

4.1.2.1 complex(kind=dp) function makeopticalelements::abt ( integer *i*, integer *j*, complex(kind=dp), dimension(:,:), intent(in), allocatable *ft*, integer *nspec* )

calculates matrix elements Alpha-Beta\*\*T for M = (A B) (B\*A\*) computes AB\*\*T and returns the i,j-th element

8 Module Documentation

#### **Parameters**

i	input index 1
j	input index 2
ft	input symplectic transform matrix for the optical circuit
nspec	input number of spectral DOF

#### Todo check this

4.1.2.2 subroutine makeopticalelements::alloc\_temparrays ( integer, intent(in) nspace, integer, intent(in) nspec )

allocates temp arrays for matrices

#### **Parameters**

nspace	input
nspec	input allocates memory for ident_spec a spectral size matrix for tensor producting.

allocates mem for spatial\_work, array size of spatial modes

allocates mem for n\_work, work array size of alpha or beta in sympectic matrix

4.1.2.3 real(kind=dp) function makeopticalelements::amp ( complex(kind=dp) a )

returns the absolute value squared |a|\*\*2

### **Parameters**

ĺ	а	input complex number to be  a **2

4.1.2.4 complex(kind=dp) function makeopticalelements::bbd ( integer, intent(in) *i*, integer, intent(in) *j*, complex(kind=dp), dimension(:,:), intent(in), allocatable *ft*, integer, intent(in) *nspec* )

calculates the matrix elements Beta\*Beta\*\*H for  $M = (A\ B\ )\ (B*\ A*)$  computes B\*B\*\*H (Hermitian conjg) and returns the i,j-th element

#### **Parameters**

i	input index 1
j	input index 2
ft	input symplectic transform matrix for the optical circuit
nspec	input number of spectral DOF

# Todo check this

- 4.1.2.5 subroutine makeopticalelements::dealloc\_temparrays ( )
- 4.1.2.6 real(kind=dp) function makeopticalelements::g4 ( complex(kind=dp), dimension(:,:), intent(in), allocatable ft, integer, intent(in) nspec )

calculates g4 using matrix elements sum

#### **Parameters**

ft	input is the full symplectic transfor	
nspec	input spectral DOF	

4.1.2.7 subroutine makeopticalelements::make\_bs ( integer *nspace*, integer *nspec*, complex(kind=dp), dimension(:,:), allocatable *symp\_mat*, integer *m1*, integer *m2*, real(kind=dp) *theta* )

makes beamsplitter symplectic matrix takes in an allocated matrix for the beamsplitter matrix to be written to uses the private ident\_spec, spatial\_work, n\_work arrays

#### **Parameters**

nspace	is number of total spatial modes
nspec	is number of total spectral modes
m_bs	allocated n*n matrix for beamsplitter
m1	is spatial mode 1 for beam splitter
m2	is spatial mode 2 for beam splitter

4.1.2.8 subroutine makeopticalelements::make\_sq ( integer *nspace*, integer *nspec*, complex(kind=dp), dimension(:,:), allocatable *symp\_mat*, integer *m1*, integer *m2*, complex(kind=dp), dimension(:,:), intent(inout) *alpha*, complex(kind=dp), dimension(:,:), intent(inout) *beta* )

make symplectic squeezing matrix from exponetiated JSA a lot is broken...

### Note

only works if modes are consectutive

#### Note

alpha & beta are 2 spatial modes and all spectral modes dim 2\*nspace\*nspec

#### loop for alpha

check this is legal... full diag sq symp\_mat(m1s:m1s+nspec, m1s+n:m1s+nspec+n)=beta(1:nspec, 1+nspec :2\*nspec)

probably not legal symp\_mat(m2s:m2s+nspec, m2s+n:m2s+nspec+n)=beta(nspec+1:2\*nspec, 1:nspec)

loop for beta, offset to col+n

10 Module Documentation

#### 4.1.3 Variable Documentation

4.1.3.1 real(kind=dp), public makeopticalelements::ident

# 4.2 olis\_f90stdlib Module Reference

#### **Functions/Subroutines**

• subroutine alloc complex eigenvects (matrix, eigenvals, u, v)

allocates eigenvals, u & v arrays for eigenvals & eigenvects

• subroutine alloc\_complex\_svd (matrix, sigma, u, vt)

allocates sigma (singular vals), u and vt for complexSVD allocates temp work arrays too

• subroutine randseed (seed)

generates random seed

subroutine printvectors (vect, desc, f)

print formatted matrices can take optional args for labels or write directly to a file

• complex(kind=dp) function, dimension(2, 2) outerproduct (a, b)

outerproduct of two complex vectors, returns a complex matrix

• complex(kind=dp) function, dimension(n, n) c\_identity (n)

makes complex identity matrix dim (nxn)

complex(kind=dp) function, dimension(:,:), allocatable tprod (a, b)

tensor product for complex matrices aXb

complex(kind=dp) function complextrace (a)

computes the trace of a complex matrix

subroutine complex\_eigenvects (a, w, vl, vr)

computes the complex eigenvalues and eigenvectors overwrites matrix in, input eigenvalue array and eigenvector arrays uses the zgeev subroutine from lapack

• subroutine complex\_svd (a, sigma, u, vt)

computes the complex eigenvalues and eigenvectors overwrites matrix in, input eigenvalue array and eigenvector arrays uses the zgeev subroutine from lapack

• complex(kind=dp) function, dimension(2, 2) c inv2 (m in)

inverse for a complex 2x2 matrix

• real(kind=dp) function matrixnorm (c)

computed Frobenieus matrix norm of complex matrix using lapack zlange

- complex(kind=dp) function, dimension(size(matrix, 1), size(matrix, 2)) expmatrix (matrix, n)
- recursive complex(kind=dp) function, dimension(size(x, 1), size(x, 2)) matrixmul (x, n)
- recursive real(kind=dp) function factorial (n)

#### **Variables**

- real(kind=dp), parameter pi =4.0\_dp\*atan(1.0)
- complex(kind=dp), parameter imaginary =(0.0\_dp, 1.0\_dp)

### 4.2.1 Function/Subroutine Documentation

4.2.1.1 subroutine olis\_f90stdlib::alloc\_complex\_eigenvects ( complex(kind=dp), dimension(:,:), intent(in) *matrix*, complex(kind=dp), dimension(:,:), intent(inout), allocatable *eigenvals*, complex(kind=dp), dimension(:,:), intent(inout), allocatable *u*, complex(kind=dp), dimension(:,:), intent(inout), allocatable *v* )

allocates eigenvals, u & v arrays for eigenvals & eigenvects allocated temp work arrays also

**Author** 

Oliver Thomas August 2018

#### **Parameters**

matrix	input complex matrix
eigenvals	1d array for eigenvalues, is overwriten on exit
и	2d array of left eigenvectors
V	3d array of right eigenvectors

4.2.1.2 subroutine olis\_f90stdlib::alloc\_complex\_svd ( complex(kind=dp), dimension(:,:), intent(in) *matrix,* real(kind=dp), dimension(:), intent(inout), allocatable *sigma,* complex(kind=dp), dimension(:,:), intent(inout), allocatable *u,* complex(kind=dp), dimension(:,:), intent(inout), allocatable *vt* )

allocates sigma (singular vals), u and vt for complexSVD allocates temp work arrays too

#### **Parameters**

matrix	input complex matrix	
sigma	real vector of singular values sorted in descending order	
и	unitary matrix	
vt	unitary matrix returns V**H NOT v	

4.2.1.3 complex(kind=dp) function, dimension(n,n) olis\_f90stdlib::c\_identity ( integer, intent(in) n )

makes complex identity matrix dim (nxn)

# Parameters

n	input dimension

 $4.2.1.4 \quad \text{complex(kind=dp) function, dimension(2,2) olis\_f90stdlib::c\_inv2 ( \ \text{complex(kind=dp), dimension(2,2), intent(in)} \ \textit{m\_in} \ )$ 

inverse for a complex 2x2 matrix

#### **Parameters**

m⊷	is input complex 2x2 matrix
_in	

4.2.1.5 subroutine olis\_f90stdlib::complex\_eigenvects ( complex(kind=dp), dimension(:,:), allocatable *a,* complex(kind=dp), dimension(:,:), allocatable *vI,* complex(kind=dp), dimension(:,:), allocatable *vr* )

computes the complex eigenvalues and eigenvectors overwrites matrix in, input eigenvalue array and eigenvector arrays uses the zgeev subroutine from lapack

12 Module Documentation

#### **Parameters**

а	input allocatable complex matrix to be diagonalised
W	output allocatable complex 1d array containing eigenvals
vI	output allocatable complex 2d array containing left eigenvectors
vr	output allocatable complex 2d array containing right eigenvectors

#### Note

need to check this is optimised

4.2.1.6 subroutine olis\_f90stdlib::complex\_svd ( complex(kind=dp), dimension(:,:), intent(inout), allocatable *a*, real(kind=dp), dimension(:,:), allocatable *u*, complex(kind=dp), dimension(:,:), allocatable *vt* )

computes the complex eigenvalues and eigenvectors overwrites matrix in, input eigenvalue array and eigenvector arrays uses the zgeev subroutine from lapack

#### **Parameters**

а	input allocatable complex matrix to be SVD'd	
sigma	output allocatable complex 1d array containing ordered singular values	
и	output allocatable complex 2d array containing u	
vt	output allocatable complex 2d array containing v**H	

#### Note

need to check this is optimised

4.2.1.7 complex(kind=dp) function olis\_f90stdlib::complextrace ( complex(kind=dp), dimension(:,:) a )

computes the trace of a complex matrix

#### **Parameters**

a is the complex matrix in

4.2.1.8 complex(kind=dp) function, dimension(size(matrix,1),size(matrix,2)) olis\_f90stdlib::expmatrix ( complex(kind=dp), dimension(:,:) matrix, integer n )

# **Parameters**

n is the number of terms in taylor expansion to consider

- 4.2.1.9 recursive real(kind=dp) function olis\_f90stdlib::factorial ( integer n )
- 4.2.1.10 recursive complex(kind=dp) function, dimension(size(x,1),size(x,2)) olis\_f90stdlib::matrixmul ( complex(kind=dp), dimension(:,:) x, integer n )
- 4.2.1.11 real(kind=dp) function olis\_f90stdlib::matrixnorm ( complex(kind=dp), dimension(:,:) c )

computed Frobenieus matrix norm of complex matrix using lapack zlange

#### **Parameters**

c input complex matrix

4.2.1.12 complex(kind=dp) function, dimension(2,2) olis\_f90stdlib::outerproduct ( complex(kind=dp), dimension(:), intent(in) a, complex(kind=dp), dimension(:), intent(in) b)

outerproduct of two complex vectors, returns a complex matrix

#### **Parameters**

а	is input vector 1,  ket>
b	is input vector 2, <bra< td=""></bra<>

4.2.1.13 subroutine olis\_f90stdlib::printvectors ( complex(kind=dp), dimension(:,:), intent(in) *vect*, character(len=\*), intent(in), optional *desc*, integer, intent(in), optional *f* )

print formatted matrices can take optional args for labels or write directly to a file

### **Parameters**

vect	is the input complex matrix	
desc	is the optional string to be written above the matrix	
f	is the optional file output unit to write to, default is console	

4.2.1.14 subroutine olis\_f90stdlib::randseed ( integer, dimension(:), allocatable seed )

generates random seed

#### **Parameters**

seed	is input allocatable 1d array

14 Module Documentation

4.2.1.15 complex(kind=dp) function, dimension(:,:), allocatable olis\_f90stdlib::tprod ( complex(kind=dp), dimension (:,:), intent(in) a, complex(kind=dp), dimension (:,:), intent(in) b)

tensor product for complex matrices aXb

# **Parameters**

а	complex matrix in
b	complex matrix in

### 4.2.2 Variable Documentation

- 4.2.2.1 complex(kind=dp), parameter olis\_f90stdlib::imaginary =(0.0\_dp, 1.0\_dp)
- 4.2.2.2 real(kind=dp), parameter olis\_f90stdlib::pi =4.0\_dp\*atan(1.0)

# **File Documentation**

# 5.1 makeopticalelements.f90 File Reference

#### **Modules**

· module makeopticalelements

module for building symplectic matrices for optical elements

#### **Functions/Subroutines**

- subroutine makeopticalelements::make\_bs (nspace, nspec, symp\_mat, m1, m2, theta)

  makes beamsplitter symplectic matrix takes in an allocated matrix for the beamsplitter matrix to be written to uses the private ident\_spec, spatial\_work, n\_work arrays
- subroutine makeopticalelements::make\_sq (nspace, nspec, symp\_mat, m1, m2, alpha, beta) make symplectic squeezing matrix from exponetiated JSA a lot is broken...
- real(kind=dp) function makeopticalelements::g4 (ft, nspec)
  - calculates g4 using matrix elements sum
- real(kind=dp) function makeopticalelements::amp (a)
  - returns the absolute value squared |a|\*\*2
- complex(kind=dp) function makeopticalelements::abt (i, j, ft, nspec)
  - calculates matrix elements Alpha-Beta\*\*T for M = (A B ) (B\* A\*) computes AB\*\*T and returns the i,j-th element
- complex(kind=dp) function makeopticalelements::bbd (i, j, ft, nspec)
  - calculates the matrix elements Beta\*Beta\*\*H for M = (A B) (B\*A\*) computes B\*B\*\*H (Hermitian conjg) and returns the i.j-th element
- subroutine makeopticalelements::alloc\_temparrays (nspace, nspec)
  - allocates temp arrays for matrices
- subroutine makeopticalelements::dealloc\_temparrays

## Variables

• real(kind=dp), public makeopticalelements::ident

16 File Documentation

# 5.2 num hom.f90 File Reference

# **Functions/Subroutines**

• program num\_hom

program to compute matrix of a JSA

- complex(kind=dp) function, dimension(:,:), allocatable make\_squeezer (mode1, mode2, jsa)

  make sqq matrix from jsa function
- real(kind=dp) function, dimension(:,:), allocatable gen\_jsa (w1\_start, w1\_end, w1\_incr, w2\_start, w2\_end, w2\_incr, sigma, outfile)

samples the given jsa for frequency ranges w1, w2

• complex(kind=dp) function f (w1, w2, sig)

JSA function taking two freq.

#### 5.2.1 Function/Subroutine Documentation

5.2.1.1 complex(kind=dp) function num\_hom::f ( real(kind=dp), intent(in) w1, real(kind=dp), intent(in) w2, real(kind=dp), intent(in) sig )

JSA function taking two freq.

#### **Parameters**

w1	input signal freq
w2	input idler freq
sig	input variance

5.2.1.2 real(kind=dp) function, dimension (:,:), allocatable num\_hom::gen\_jsa ( real(kind=dp), intent(in) w1\_start, real(kind=dp), intent(in) w1\_end, real(kind=dp), intent(in) w1\_incr, real(kind=dp), intent(in) w2\_start, real(kind=dp), intent(in) w2\_end, real(kind=dp), intent(in) w2\_incr, real(kind=dp), intent(in) sigma, integer outfile)

samples the given jsa for frequency ranges w1, w2

### **Parameters**

f_mat	allocatable Jsa matrix values out	
w_start		

5.2.1.3 complex(kind=dp) function, dimension(:,:), allocatable num\_hom::make\_squeezer ( integer, intent(in) *mode1*, integer, intent(in) *mode2*, complex(kind=dp), dimension(:,:), intent(in), allocatable *jsa* )

make sqq matrix from jsa function

Note

files to write to

Note

```
to make off diagonal for fmatrix m_sq=0.0_dp ! top right m_sq(1:1*f_size, 3*f_size+1:4*f_size)=1 ! mid right m_sq(1*f_size+1:2*f_size, 2*f_size+1:3*f_size)=2 ! mid left m_sq(2*f_size+1:3*f_size, 1*f_size+1:2*f_ \leftrightarrow size)=3 ! bot left m_sq(3*f_size+1:4*f_size, 1:1*f_size)=4 !h= 0.0 F_JSA F_JSA*T 0.0
```

```
f jsa = f mat
```

M sq =  $\exp(i (0 H) (-H*0)$ 

 $M_sq = exp(i (0 0 0 F_JSA) (0 0 F_JSA**T 0) (0 -conjg(F_JSA) 0 0) (-F_JSA**H 0 0 0)$ 

Note

alpha beta are top left and top right of M M = (A B) (B\*A\*)

#### **Parameters**

```
alpha size is 2*f size as all spectral modes for 2 spatial
```

#### Note

allocate for sq on modes 1&2

5.2.1.4 program num\_hom ( )

program to compute matrix of a JSA

# 5.3 olis\_f90stdlib.f90 File Reference

## Modules

· module olis f90stdlib

#### **Functions/Subroutines**

- subroutine olis\_f90stdlib::alloc\_complex\_eigenvects (matrix, eigenvals, u, v)
   allocates eigenvals, u & v arrays for eigenvals & eigenvects
- subroutine olis\_f90stdlib::alloc\_complex\_svd (matrix, sigma, u, vt)

allocates sigma (singular vals), u and vt for complexSVD allocates temp work arrays too

• subroutine olis\_f90stdlib::randseed (seed)

generates random seed

- subroutine olis\_f90stdlib::printvectors (vect, desc, f)
  - print formatted matrices can take optional args for labels or write directly to a file
- complex(kind=dp) function, dimension(2, 2) olis\_f90stdlib::outerproduct (a, b)
  - outerproduct of two complex vectors, returns a complex matrix
- complex(kind=dp) function, dimension(n, n) olis\_f90stdlib::c\_identity (n)

18 File Documentation

makes complex identity matrix dim (nxn)

• complex(kind=dp) function, dimension(:,:), allocatable olis\_f90stdlib::tprod (a, b)

tensor product for complex matrices aXb

• complex(kind=dp) function olis\_f90stdlib::complextrace (a)

computes the trace of a complex matrix

• subroutine olis\_f90stdlib::complex\_eigenvects (a, w, vl, vr)

computes the complex eigenvalues and eigenvectors overwrites matrix in, input eigenvalue array and eigenvector arrays uses the zgeev subroutine from lapack

• subroutine olis f90stdlib::complex svd (a, sigma, u, vt)

computes the complex eigenvalues and eigenvectors overwrites matrix in, input eigenvalue array and eigenvector arrays uses the zgeev subroutine from lapack

• complex(kind=dp) function, dimension(2, 2) olis\_f90stdlib::c\_inv2 (m\_in)

inverse for a complex 2x2 matrix

real(kind=dp) function olis\_f90stdlib::matrixnorm (c)

computed Frobenieus matrix norm of complex matrix using lapack zlange

- complex(kind=dp) function, dimension(size(matrix, 1), size(matrix, 2)) olis f90stdlib::expmatrix (matrix, n)
- recursive complex(kind=dp) function, dimension(size(x, 1), size(x, 2)) olis\_f90stdlib::matrixmul (x, n)
- recursive real(kind=dp) function olis f90stdlib::factorial (n)

### **Variables**

- real(kind=dp), parameter olis\_f90stdlib::pi =4.0\_dp\*atan(1.0)
- complex(kind=dp), parameter olis\_f90stdlib::imaginary =(0.0\_dp, 1.0\_dp)

# Index

abt	num hom.f90, 16
makeopticalelements, 7	makeopticalelements, 7
alloc_complex_eigenvects	abt, 7
olis_f90stdlib, 10	alloc_temparrays, 8
alloc_complex_svd	amp, 8
olis_f90stdlib, 11	bbd, 8
alloc_temparrays	dealloc_temparrays, 9
makeopticalelements, 8	g4, 9
amp	ident, 10
makeopticalelements, 8	make_bs, 9
•	make_sq, 9
bbd	makeopticalelements.f90, 15
makeopticalelements, 8	matrixmul
	olis_f90stdlib, 13
c_identity	matrixnorm
olis_f90stdlib, 11	olis_f90stdlib, 13
c_inv2	_ ,
olis_f90stdlib, 11	num_hom
complex_eigenvects	num_hom.f90, 17
olis_f90stdlib, 11	num_hom.f90, 16
complex_svd	f, 16
olis_f90stdlib, 12	gen_jsa, 16
complextrace	make_squeezer, 16
olis_f90stdlib, 12	num_hom, 17
dealloc_temparrays	II. (00 t III. t 0
makeopticalelements, 9	olis_f90stdlib, 10
manoophoaloiomorno, o	alloc_complex_eigenvects, 10
expmatrix	alloc_complex_svd, 11
olis_f90stdlib, 12	c_identity, 11
	c_inv2, 11
f	complex_eigenvects, 11
num_hom.f90, 16	complex_svd, 12
factorial	complextrace, 12
olis_f90stdlib, 12	expmatrix, 12
	factorial, 12
g4	imaginary, 14
makeopticalelements, 9	matrixmul, 13 matrixnorm, 13
gen_jsa	
num_hom.f90, 16	outerproduct, 13 pi, 14
ident	printvectors, 13
	randseed, 13
makeopticalelements, 10	tprod, 13
imaginary	olis_f90stdlib.f90, 17
olis_f90stdlib, 14	
make bs	outerproduct
makeopticalelements, 9	olis_f90stdlib, 13
make_sq	pi
makeopticalelements, 9	olis f90stdlib, 14
make_squeezer	printvectors
a.to_0qu00201	printvootora

20 INDEX

```
olis_f90stdlib, 13

randseed
olis_f90stdlib, 13

tprod
olis_f90stdlib, 13
```