CaPS

v0.5

Generated by Doxygen 1.8.13

Contents

1	CaP		1
	1.1	Overview	1
	1.2	Files	1
2	Data	a Structure Index	3
	2.1	Data Structures	3
3	File	Index	5
	3.1	File List	5
4	Data	a Structure Documentation	7
	4.1	argparse Struct Reference	7
		4.1.1 Detailed Description	8
	4.2	argparse_option Struct Reference	8
		4.2.1 Detailed Description	8
	4.3	buf Struct Reference	9
		4.3.1 Field Documentation	9
		4.3.1.1 buffer	9
		4.3.1.2 capacity	9
		4.3.1.3 size	9
	4.4	cache_entry_t Struct Reference	9
	4.5	cache_t Struct Reference	0
	4.6	caps Struct Reference	0
		4.6.1 Detailed Description	11
		4.6.2 Field Documentation	11

ii CONTENTS

4.6.2.1	calL	11
4.6.2.2	detalg	11
4.6.2.3	epsrel	11
4.6.2.4	L	11
4.6.2.5	LbyR	11
4.6.2.6	ldim	11
4.6.2.7	R	12
4.6.2.8	y	12
4.7 caps_M_t Struct	t Reference	12
4.8 caps_mpi_t Struc	uct Reference	13
4.9 caps_task_t Stru	uct Reference	14
4.10 integrand_plasm	na_t Struct Reference	14
4.11 integrand_t Struc	ct Reference	14
4.11.1 Detailed	Description	15
4.12 integration_plasr	ma_t Struct Reference	15
4.13 integration_t Stru	uct Reference	16
4.13.1 Detailed	Description	16
4.14 kernel_args_t St	truct Reference	17
4.15 log_t Struct Refe	erence	17
4.15.1 Detailed	Description	17
4.15.2 Field Do	ocumentation	17
4.15.2.1	s	17
4.15.2.2	? v	17
4.16 material_t Struct	t Reference	18
4.16.1 Detailed	d Description	18
4.16.2 Field Do	ocumentation	18
4.16.2.1	calL	18
4.16.2.2	? epsm1	18
4.16.2.3	filename	18
4.16.2.4	gamma_high	19

CONTENTS

			4.16.2.5	gamma_low	19
			4.16.2.6	omegap_high	19
			4.16.2.7	omegap_low	19
			4.16.2.8	points	19
			4.16.2.9	xi	19
			4.16.2.10	xi_max	19
			4.16.2.11	xi_min	20
	4.17	matrix_	t Struct Re	eference	20
		4.17.1	Detailed I	Description	20
		4.17.2	Field Doo	eumentation	20
			4.17.2.1	dim	20
			4.17.2.2	dim2	20
			4.17.2.3	lda	20
			4.17.2.4	M	20
_					21
•	Fila	IIOCIIM	ntation		
5			entation	proneo	
5	5.1	bessel.	c File Refe	Prence	21
5		bessel.	c File Refe	Description	21 23
5		bessel.	c File Refe Detailed I	Description	21 23 23
5		bessel.	Detailed I Function 5.1.2.1	Description	21 23 23 23
5		bessel.	Detailed I Function 5.1.2.1 5.1.2.2	Description Documentation bessel_I0() bessel_I1()	21 23 23 23 24
5		bessel.	Detailed I Function 5.1.2.1 5.1.2.2 5.1.2.3	Description Documentation bessel_l0() bessel_l1() bessel_ln()	21 23 23 23 24 25
5		bessel.	Detailed I Function 5.1.2.1 5.1.2.2 5.1.2.3 5.1.2.4	Description Documentation bessel_I0() bessel_I1() bessel_In() bessel_K0()	21 23 23 23 24 25 26
5		bessel.	Detailed I Function 5.1.2.1 5.1.2.2 5.1.2.3 5.1.2.4 5.1.2.5	Description Documentation bessel_l0() bessel_l1() bessel_ln() bessel_K0() bessel_K1()	21 23 23 24 25 26
5		bessel.	Detailed I Function 5.1.2.1 5.1.2.2 5.1.2.3 5.1.2.4 5.1.2.5 5.1.2.6	Description Documentation bessel_I0() bessel_I1() bessel_K0() bessel_K1() bessel_Kn()	21 23 23 24 25 26 26 27
5		bessel.	Detailed I Function 5.1.2.1 5.1.2.2 5.1.2.3 5.1.2.4 5.1.2.5 5.1.2.6 5.1.2.7	Description Documentation bessel_I0() bessel_I1() bessel_In() bessel_K0() bessel_K1() bessel_Kn() bessel_logI0()	21 23 23 24 25 26 26 27 28
5		bessel.	Detailed I Function 5.1.2.1 5.1.2.2 5.1.2.3 5.1.2.4 5.1.2.5 5.1.2.6 5.1.2.7 5.1.2.8	Description Documentation bessel_I0() bessel_I1() bessel_In() bessel_K0() bessel_K1() bessel_Kn() bessel_logI0() bessel_logI1()	21 23 23 24 25 26 26 27 28 29
5		bessel.	Detailed I Function 5.1.2.1 5.1.2.2 5.1.2.3 5.1.2.4 5.1.2.5 5.1.2.6 5.1.2.7 5.1.2.8 5.1.2.9	Description Documentation bessel_l0() bessel_l1() bessel_K0() bessel_K1() bessel_Kn() bessel_logl0() bessel_logl1()	21 23 23 24 25 26 26 27 28 29 30
5		bessel.	Detailed I Function 5.1.2.1 5.1.2.2 5.1.2.3 5.1.2.4 5.1.2.5 5.1.2.6 5.1.2.7 5.1.2.8 5.1.2.9 5.1.2.10	Description	21 23 23 24 25 26 27 28 29 30 31
5		bessel.	Detailed I Function 5.1.2.1 5.1.2.2 5.1.2.3 5.1.2.4 5.1.2.5 5.1.2.6 5.1.2.7 5.1.2.8 5.1.2.9 5.1.2.10	Description Documentation bessel_l0() bessel_l1() bessel_K0() bessel_K1() bessel_Kn() bessel_logl0() bessel_logl1()	21 23 23 24 25 26 26 27 28 29 30

iv CONTENTS

		5.1.2.13	bessel_logInu_series()	. 34
		5.1.2.14	bessel_logK0()	. 35
		5.1.2.15	bessel_logK1()	. 36
		5.1.2.16	bessel_logKn()	. 37
		5.1.2.17	bessel_logKn_half()	. 38
		5.1.2.18	bessel_logKn_recursive()	. 39
		5.1.2.19	bessel_logKnu_asymp()	. 40
		5.1.2.20	bessel_ratiol()	. 41
		5.1.2.21	chbevI()	. 42
	5.1.3	Variable	Documentation	. 42
		5.1.3.1	I0_coeffs	. 43
		5.1.3.2	I1_coeffs	. 43
		5.1.3.3	K0_coeffsA	. 44
		5.1.3.4	K0_coeffsB	. 44
		5.1.3.5	K1_coeffsA	. 45
		5.1.3.6	K1_coeffsB	. 45
5.2	cache.	c File Refe	erence	. 46
	5.2.1	Detailed	Description	. 46
	5.2.2	Function	Documentation	. 47
		5.2.2.1	cache_free()	. 47
		5.2.2.2	cache_insert()	. 47
		5.2.2.3	cache_lookup()	. 48
		5.2.2.4	cache_new()	. 48
5.3	cquadp	oack/includ	de/quadpack.h File Reference	. 49
	5.3.1	Detailed	Description	. 50
	5.3.2	Macro De	efinition Documentation	. 50
		5.3.2.1	GK_10_21	. 50
		5.3.2.2	GK_15_31	. 50
		5.3.2.3	GK_20_41	. 50
		5.3.2.4	GK_25_51	. 51

CONTENTS

		5.3.2.5	GK_30_61	5	51
		5.3.2.6	GK_7_15	5	51
	5.3.3	Function	n Documentation	5	51
		5.3.3.1	dqage()	5	51
		5.3.3.2	dqagi()	5	52
		5.3.3.3	dqags()	5	53
5.4	fcqs.c	File Refere	rence	5	55
	5.4.1	Detailed	Description	5	55
	5.4.2	Macro D	Definition Documentation	5	56
		5.4.2.1	MMIN	5	56
	5.4.3	Function	n Documentation	5	56
		5.4.3.1	cot2()	5	56
		5.4.3.2	fcqs_finite()	5	56
		5.4.3.3	fcqs_semiinf()	5	57
		5.4.3.4	wi_finite()	5	58
		5.4.3.5	wi_semiinf()	5	58
5.5	include	e/buf.h File	e Reference	5	59
	5.5.1	Detailed	B Description	6	60
	5.5.2	Macro D	Definition Documentation	6	60
		5.5.2.1	buf_capacity	6	60
		5.5.2.2	buf_clear	6	61
		5.5.2.3	buf_free	6	61
		5.5.2.4	buf_grow	6	61
		5.5.2.5	buf_pop	6	61
		5.5.2.6	buf_push	6	61
		5.5.2.7	buf_size	6	62
		5.5.2.8	buf_trunc	6	62
5.6	include	e/constants	ts.h File Reference	6	62
	5.6.1	Detailed	Description	6	63
	5.6.2	Macro D	Definition Documentation	6	63

vi

		5.6.2.1	CAPS_c	. 63
		5.6.2.2	CAPS_hbar	. 63
		5.6.2.3	CAPS_hbar_eV	. 63
		5.6.2.4	CAPS_kB	. 63
		5.6.2.5	M_LOG2	. 63
		5.6.2.6	M_LOGPI	. 64
		5.6.2.7	M_PI	. 64
		5.6.2.8	MAX	. 64
		5.6.2.9	MIN	. 64
		5.6.2.10	pow_2	. 64
		5.6.2.11	SGN	. 64
	5.6.3	Typedef [Documentation	. 64
		5.6.3.1	sign_t	. 65
5.7	include	e/libcaps.h	File Reference	. 65
	5.7.1	Detailed	Description	. 68
	5.7.2	Macro De	efinition Documentation	. 68
		5.7.2.1	CAPS_CACHE_ELEMS	. 68
		5.7.2.2	CAPS_EPSREL	. 68
		5.7.2.3	CAPS_FACTOR_LDIM	. 68
		5.7.2.4	CAPS_MINIMUM_LDIM	. 68
	5.7.3	Typedef [Documentation	. 68
		5.7.3.1	caps_t	. 69
	5.7.4	Enumera	ation Type Documentation	. 69
		5.7.4.1	polarization_t	. 69
	5.7.5	Function	Documentation	. 69
		5.7.5.1	caps_build()	. 69
		5.7.5.2	caps_epsilonm1_drude()	. 69
		5.7.5.3	caps_epsilonm1_perf()	. 70
		5.7.5.4	caps_epsilonm1_plate()	. 71
		5.7.5.5	caps_epsilonm1_sphere()	. 71

CONTENTS vii

	5.7.5.6	caps_estimate_lminmax()	72
	5.7.5.7	caps_free()	73
	5.7.5.8	caps_fresnel()	73
	5.7.5.9	caps_get_detalg()	74
	5.7.5.10	caps_get_epsrel()	74
	5.7.5.11	caps_get_ldim()	75
	5.7.5.12	caps_ht_drude()	75
	5.7.5.13	caps_ht_perf()	76
	5.7.5.14	caps_ht_plasma()	77
	5.7.5.15	caps_info()	78
	5.7.5.16	caps_init()	78
	5.7.5.17	caps_kernel_M()	79
	5.7.5.18	caps_kernel_M0_EE()	80
	5.7.5.19	caps_kernel_M0_MM()	81
	5.7.5.20	caps_kernel_M0_MM_plasma()	82
	5.7.5.21	caps_InLambda()	83
	5.7.5.22	caps_logdetD()	84
	5.7.5.23	caps_logdetD0()	86
	5.7.5.24	caps_M_elem()	87
	5.7.5.25	caps_M_free()	88
	5.7.5.26	caps_M_init()	89
	5.7.5.27	caps_mie()	90
	5.7.5.28	caps_mie_perf()	92
	5.7.5.29	caps_set_detalg()	93
	5.7.5.30	caps_set_epsilonm1()	94
	5.7.5.31	caps_set_epsilonm1_plate()	94
	5.7.5.32	caps_set_epsilonm1_sphere()	95
	5.7.5.33	caps_set_epsrel()	96
	5.7.5.34	caps_set_ldim()	96
5.8	include/utils.h File	e Reference	97

viii CONTENTS

	5.8.1	Detailed	Description	98
	5.8.2	Macro De	efinition Documentation	98
		5.8.2.1	COMPILER	98
		5.8.2.2	TERMINATE	99
		5.8.2.3	WARN	99
		5.8.2.4	xfree	99
	5.8.3	Function	Documentation	99
		5.8.3.1	disable_buffering()	99
		5.8.3.2	now()	99
		5.8.3.3	strim()	100
		5.8.3.4	strrep()	100
		5.8.3.5	time_as_string()	100
		5.8.3.6	xcalloc()	101
		5.8.3.7	xmalloc()	101
		5.8.3.8	xrealloc()	102
5.9	integra	tion.c File	Reference	103
	5.9.1	Detailed	Description	104
	5.9.2	Function	Documentation	104
		5.9.2.1	caps_integrate_A()	105
		5.9.2.2	caps_integrate_B()	106
		5.9.2.3	caps_integrate_C()	106
		5.9.2.4	caps_integrate_D()	108
		5.9.2.5	caps_integrate_free()	108
		5.9.2.6	caps_integrate_I()	109
		5.9.2.7	caps_integrate_init()	110
		5.9.2.8	caps_integrate_K()	111
		5.9.2.9	caps_integrate_plasma()	112
		5.9.2.10	caps_integrate_plasma_free()	113
		5.9.2.11	caps_integrate_plasma_init()	114
		5.9.2.12	K_estimate()	115

CONTENTS

5.10	libcaps	File Reference	16
	5.10.1	Detailed Description	19
	5.10.2	Function Documentation	19
		5.10.2.1 caps_build()	19
		5.10.2.2 caps_epsilonm1_drude()	19
		5.10.2.3 caps_epsilonm1_perf()	20
		5.10.2.4 caps_epsilonm1_plate()	20
		5.10.2.5 caps_epsilonm1_sphere()	21
		5.10.2.6 caps_estimate_Iminmax()	22
		5.10.2.7 caps_free()	22
		5.10.2.8 caps_fresnel()	24
		5.10.2.9 caps_get_detalg()	24
		5.10.2.10 caps_get_epsrel()	25
		5.10.2.11 caps_get_ldim()	25
		5.10.2.12 caps_ht_drude()	25
		5.10.2.13 caps_ht_perf()	26
		5.10.2.14 caps_ht_plasma()	27
		5.10.2.15 caps_info()	28
		5.10.2.16 caps_init()	29
		5.10.2.17 caps_kernel_M()	30
		5.10.2.18 caps_kernel_M0_EE()	31
		5.10.2.19 caps_kernel_M0_MM()	32
		5.10.2.20 caps_kernel_M0_MM_plasma()	33
		5.10.2.21 caps_InLambda()	34
		5.10.2.22 caps_logdetD()	35
		5.10.2.23 caps_logdetD0()	36
		5.10.2.24 caps_M_elem()	38
		5.10.2.25 caps_M_free()	39
		5.10.2.26 caps_M_init()	40
		5.10.2.27 caps_mie()	41

CONTENTS

	5.10.2.28 caps_mie_perf()
	5.10.2.29 caps_set_detalg()
	5.10.2.30 caps_set_epsilonm1()
	5.10.2.31 caps_set_epsilonm1_plate()
	5.10.2.32 caps_set_epsilonm1_sphere()
	5.10.2.33 caps_set_epsrel()
	5.10.2.34 caps_set_ldim()
5.11 libho	dlr/include/hodlr.h File Reference
5.11.	1 Detailed Description
5.11.	2 Function Documentation
	5.11.2.1 hodlr_logdet()
	5.11.2.2 hodlr_logdet_diagonal()
5.12 logfa	c.c File Reference
5.12.	1 Detailed Description
5.12.	2 Function Documentation
	5.12.2.1 lfac()
	5.12.2.2 lfac2()
	5.12.2.3 logi()
5.12.	3 Variable Documentation
	5.12.3.1 lookup_lfac
	5.12.3.2 lookup_logi
5.13 mate	rial.c File Reference
5.13.	1 Detailed Description
5.13.	2 Function Documentation
	5.13.2.1 _parse()
	5.13.2.2 material_epsilonm1()
	5.13.2.3 material_free()
	5.13.2.4 material_get_extrapolation()
	5.13.2.5 material_info()
	5.13.2.6 material_init()

CONTENTS xi

5.14 matrix	.c File Reference
5.14.1	Detailed Description
5.14.2	Function Documentation
	5.14.2.1 kernel_logdet()
	5.14.2.2 matrix_alloc()
	5.14.2.3 matrix_copy()
	5.14.2.4 matrix_free()
	5.14.2.5 matrix_load_from_file()
	5.14.2.6 matrix_load_from_stream()
	5.14.2.7 matrix_logdet_cholesky()
	5.14.2.8 matrix_logdet_dense()
	5.14.2.9 matrix_logdet_lu()
	5.14.2.10 matrix_logdet_qr()
	5.14.2.11 matrix_logdet_triangular()
	5.14.2.12 matrix_mult()
	5.14.2.13 matrix_norm_frobenius()
	5.14.2.14 matrix_save_to_file()
	5.14.2.15 matrix_save_to_stream()
	5.14.2.16 matrix_setall()
	5.14.2.17 matrix_trace()
	5.14.2.18 matrix_trace2()
5.15 misc.c	File Reference
5.15.1	Detailed Description
5.15.2	Function Documentation
	5.15.2.1 kahan_sum()
	5.15.2.2 logadd()
	5.15.2.3 logadd_ms()
	5.15.2.4 sqrtpm1()
5.16 plm.c	File Reference
5.16.1	Detailed Description

xii CONTENTS

5.16.2.	1 _fn()	178
5.16.2.	2 _PI1()	179
5.16.2.	3 _PI2()	179
5.16.2.	4 _PI3()	180
5.16.2.	5 dlnPlm()	181
5.16.2.	6 InPI()	182
5.16.2.	7 InPlm()	183
5.16.2.	8 InPlm_downwards()	184
5.16.2.	9 InPlm_upwards()	185
5.16.2.	10 Plm_continued_fraction()	186
5.17 psd.c File Refe	erence	187
5.17.1 Detaile	ed Description	188
5.17.2 Function	on Documentation	188
5.17.2.	1 _eta()	188
5.17.2.	2 dstemr_()	189
5.17.2.	3 psd()	190
5.18 utils.c File Refe	erence	190
5.18.1 Detaile	ed Description	191
5.18.2 Function	on Documentation	192
5.18.2.	1 disable_buffering()	192
5.18.2.	2 now()	192
5.18.2.	3 strim()	192
5.18.2.	4 strrep()	192
5.18.2.	5 time_as_string()	193
5.18.2.	6 xcalloc()	193
5.18.2.	7 xmalloc()	194
5.18.2.	8 xrealloc()	195
Index		197

Chapter 1

CaPS

1.1 Overview

CaPS implements the numerics for the Casimir effect in the plane-sphere geometry for arbitrary materials at zero and finite temperature using the scattering approach.

This document describes the API of the CaPS library. The compilation of the software and the usage of the programs are described in the user manual located in the directory docs/.

You can re-generate this documentation running doxygen doxygen.conf in src/. The output directory of the documentation is docs/api.

1.2 Files

file	description
caps.c	command line interface to libcaps (see also user manual and usage)
caps_logdetD.c	command line interface to compute determinants of the scattering matrix (see also user manual and usage)
capc.cpp	command line interface to compute Casimir interaction in the plane-cylinder geometry (see also user manual and usage)
cquadpack/src/*.c	integration routines (CQUADPACK), see cquadpack/include/quadpack.h
libhodlr/src/hodlr.cpp	C wrapper for the HODLR library (see libhodlr/include/hodlr.h)
libcaps.c	main part of the library
plm.c	functions to compute Legendre polynomials and associated Legendre polynomials
bessel.c	functions to compute modified Bessel functions
matrix.c	linear algebra functions; in particular computation of determinants
integration.c	routines to compute integrals that appear in the matrix elements of the round-trip operator
fcqs.c	integration routines using adapative convergent Fourier-Chebshev quadrature scheme
utils.c	wrappers for malloc, calloc realloc, and a few more useful functions
cache.c	implementation of a simple cache using a hash table
logfac.c	fast computation of $\log(n)$, $\log(n!)$, and $\log(n!!)$ for integer n
psd.c	weights and poles for Pade spectrum decomposition
misc.c	various mathematical functions
material.c	support for arbitrary dielectric functions
argparse.c	library to parse command line parameters

2 CaPS

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

argparse	. 7
argparse_option	. 8
buf	
cache_entry_t	. 9
$cache_t \ \ldots $. 10
caps	. 10
$caps_M_t \ \dots $	
caps_mpi_t	
caps_task_t	. 14
integrand_plasma_t	. 14
integrand_t	
integration_plasma_t	. 15
integration_t	. 16
kernel_args_t	. 17
$log_t \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots$. 17
material_t	. 18
matrix t	20

Data Structure Index

Chapter 3

File Index

3.1 File List

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

bessel.c		
Computation of Bessel functions		21
cache.c Implementation of a simple cache using a hash table		46
fcqs.c	• •	40
Exponentially convergent Fourier-Chebshev quadrature scheme (experimental)		55
integration.c		
Perform integration for arbitrary materials		103
libcaps.c		
Library to calculate the free Casimir energy in the plane-sphere geometry		116
logfac.c		149
Computation of logarithm and factorial for integer arguments; created by logfac.py material.c		149
Support for arbitrary dielectric functions		153
matrix.c	• •	100
Matrix functions		157
misc.c	• •	
Various mathematical functions		173
plm.c		
Computation of Legendre and associated Legendre polynomials		177
psd.c		
Expansion coefficients and poles for Pade spectrum decomposition		187
utils.c		
Wrappers for malloc, calloc realloc, and a few more useful functions		190
cquadpack/include/quadpack.h		
Library for numerical integration of one-dimensional functions		
include/ argparse.h		??
include/bessel.h		??
include/buf.h Growable memory buffers for C99		59
include/cache.h		
include/capc.h		
include/caps.h		??
include/ clapack.h		??
include/constants.h		
Define macros and constants		62

6 File Index

blude/ fcqs.h	?'
clude/integration.h	?'
clude/libcaps.h	6
clude/ logfac.h	?
clude/material.h	?1
clude/matrix.h	?1
clude/misc.h	?1
blude/ plm.h	?1
blude/ psd.h	?1
clude/utils.h	
Wrappers for malloc, calloc and realloc, assert-like macros, now()-function	97
hodlr/include/hodlr.h	
C wrapper for HODLR library	47

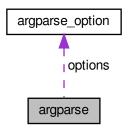
Chapter 4

Data Structure Documentation

4.1 argparse Struct Reference

```
#include <argparse.h>
```

Collaboration diagram for argparse:



Data Fields

- const struct argparse_option * options
- const char *const * usages
- int flags
- const char * description
- const char * epilog
- int argc
- const char ** argv
- const char ** out
- int cpidx
- const char * optvalue

4.1.1 Detailed Description

argpparse

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

· include/argparse.h

4.2 argparse_option Struct Reference

```
#include <argparse.h>
```

Data Fields

- enum argparse_option_type type
- · const char short name
- const char * long_name
- void * value
- const char * help
- argparse callback * callback
- intptr_t data
- · int flags

4.2.1 Detailed Description

argparse option

type: holds the type of the option, you must have an ARGPARSE_OPT_END last in your array.

short_name: the character to use as a short option name, '\0' if none.

long_name: the long option name, without the leading dash, NULL if none.

value: stores pointer to the value to be filled.

help: the short help message associated to what the option does. Must never be NULL (except for ARGPARS \leftarrow E OPT END).

 $\verb|callback|: function| is called when corresponding argument is parsed.$

data: associated data. Callbacks can use it like they want.

flags: option flags.

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

· include/argparse.h

4.3 buf Struct Reference 9

4.3 buf Struct Reference

Data Fields

- size_t capacity
- size_t size
- char buffer []

4.3.1 Field Documentation

```
4.3.1.1 buffer
```

```
char buf::buffer[]
```

buffer

4.3.1.2 capacity

```
size_t buf::capacity
```

total capacity of buffer

4.3.1.3 size

```
size_t buf::size
```

size / number of elements

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

• include/buf.h

4.4 cache_entry_t Struct Reference

Data Fields

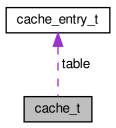
- uint64_t key
- · double value

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

• include/cache.h

4.5 cache_t Struct Reference

Collaboration diagram for cache_t:



Data Fields

- · unsigned int num_entries
- cache_entry_t * table

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

· include/cache.h

4.6 caps Struct Reference

#include <libcaps.h>

Data Fields

geometry

- double L
- double R
- double calL
- double LbyR
- double y

dielectric function of the plate

- double(* epsilonm1_plate)(double xi_, void *userdata)
- void * userdata_plate

dielectric function of the sphere

- double(* epsilonm1_sphere)(double xi_, void *userdata)
- void * userdata_sphere

accuracy and numerical parameters

- int Idim
- double epsrel
- detalg_t detalg

4.6.1 Detailed Description

The CaPS object. This structure stores all essential information about temperature, geometry and the reflection properties of the mirrors.

Do not modify the attributes of the structure yourself!

4.6.2 Field Documentation

```
4.6.2.1 calL
double caps::calL
L + R
4.6.2.2 detalg
detalg_t caps::detalg
algorithm to calculate determinant
4.6.2.3 epsrel
double caps::epsrel
relative error for integration
4.6.2.4 L
double caps::L
separation of plane and sphere
4.6.2.5 LbyR
double caps::LbyR
L/R
```

Generated by Doxygen

truncation value for vector space ℓ_{max}

int caps::ldim

4.6.2.6 Idim

4.6.2.7 R

double caps::R

radius of sphere

4.6.2.8 y

double caps::y

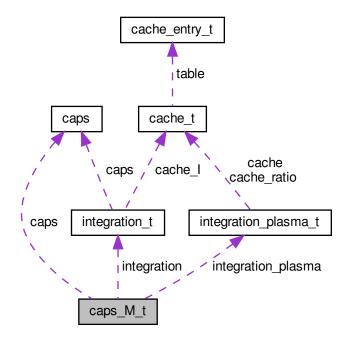
log(R/(R+L)/2)

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

• include/libcaps.h

4.7 caps_M_t Struct Reference

Collaboration diagram for caps_M_t:



Data Fields

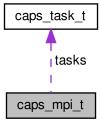
- caps t * caps
- int **m**
- int Imin
- int Idim
- integration_t * integration
- integration_plasma_t * integration_plasma
- · double xi_
- · double * al
- double * bl

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

• include/libcaps.h

4.8 caps_mpi_t Struct Reference

Collaboration diagram for caps_mpi_t:



Data Fields

- double L
- double R
- · double T
- · double omegap
- double gamma
- · double cutoff
- · double iepsrel
- double alpha
- int Idim
- · int cores
- bool verbose
- caps_task_t ** tasks
- int determinants
- char filename [512]
- double cache [4096][2]
- int cache_elems

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

· include/caps.h

4.9 caps_task_t Struct Reference

Data Fields

- int index
- int m
- double xi_
- · double recv
- double value
- MPI_Request request
- int state

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

• include/caps.h

4.10 integrand_plasma_t Struct Reference

Data Fields

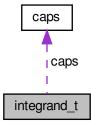
- int **nu**
- · double omegap
- double log_prefactor

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

· integration.c

4.11 integrand_t Struct Reference

 $Collaboration\ diagram\ for\ integrand_t:$



Data Fields

- int nu
- int m
- polarization_t p
- · double factor
- double alpha
- double log_normalization
- caps_t * caps

4.11.1 Detailed Description

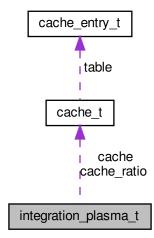
arguments for integrand in function K_integrand

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

· integration.c

4.12 integration_plasma_t Struct Reference

Collaboration diagram for integration_plasma_t:



Data Fields

- · double LbyR
- · double alpha
- double omegap_
- double epsrel
- cache_t * cache
- cache_t * cache_ratio

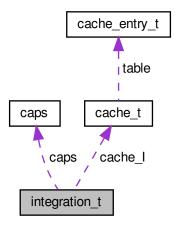
The documentation for this struct was generated from the following file:

• include/libcaps.h

4.13 integration_t Struct Reference

```
#include <libcaps.h>
```

Collaboration diagram for integration_t:



Data Fields

- caps_t * caps
- int m
- · double alpha
- double epsrel
- cache_t * cache_I
- double * cache_K [2]
- size_t elems_cache_K
- bool is_pr

4.13.1 Detailed Description

object for integration over k in matrix elements of round-trip operator

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

• include/libcaps.h

4.14 kernel_args_t Struct Reference

Data Fields

- int Imax
- int type
- · char DN
- double alpha
- double * cache_ratio
- double * cache_K

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

• include/capc.h

4.15 log_t Struct Reference

```
#include <misc.h>
```

Data Fields

- sign_t s
- double v

4.15.1 Detailed Description

represent number v by its sign and $\log |v|$

4.15.2 Field Documentation

```
4.15.2.1 s
sign_t log_t::s
sign of number
4.15.2.2 v
```

double log_t::v

logarithm of absolute value of number

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

· include/misc.h

4.16 material_t Struct Reference

```
#include <material.h>
```

Data Fields

- char filename [512]
- double calL
- double xi_min
- double xi_max
- size_t points
- double * xi
- double * epsm1
- double omegap_low
- double gamma_low
- double omegap_high
- double gamma_high

4.16.1 Detailed Description

material_t data type

4.16.2 Field Documentation

4.16.2.1 calL

double material_t::calL

L+R

4.16.2.2 epsm1

double* material_t::epsm1

tabulated dielectric function, $\epsilon(\mathrm{i}\xi)-1$

4.16.2.3 filename

char material_t::filename[512]

material filename or \0\0...

```
4.16.2.4 gamma_high
double material_t::gamma_high
relaxation frequency for hight frequency extrapolation
4.16.2.5 gamma_low
double material_t::gamma_low
relaxation frequency for low frequency extrapolation
4.16.2.6 omegap_high
double material_t::omegap_high
plasma frequency for high frequency extrapolation
4.16.2.7 omegap_low
double material_t::omegap_low
plasma frequency for low frequency extrapolation
4.16.2.8 points
size_t material_t::points
number of points
4.16.2.9 xi
double* material_t::xi
tabulated frequencies \xi
4.16.2.10 xi_max
double material_t::xi_max
```

upper border of tabulated frequencies

4.16.2.11 xi_min

```
double material_t::xi_min
```

lower border of tabulated frequencies

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

• include/material.h

4.17 matrix_t Struct Reference

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

Data Fields

- size_t dim
- size_t dim2
- size_t lda
- double * M

4.17.1 Detailed Description

define matrix type

4.17.2 Field Documentation

```
4.17.2.1 dim
```

```
size_t matrix_t::dim
```

dimension of matrix

4.17.2.2 dim2

size_t matrix_t::dim2

square of dimension of matrix

4.17.2.3 Ida

size_t matrix_t::lda

leading order

4.17.2.4 M

double* matrix_t::M

pointer to data

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

· include/matrix.h

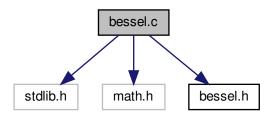
Chapter 5

File Documentation

5.1 bessel.c File Reference

Computation of Bessel functions.

```
#include <stdlib.h>
#include <math.h>
#include "bessel.h"
Include dependency graph for bessel.c:
```



Functions

modified Bessel functions for integer orders

- double bessel_In (int n, double x)
 - Modified Bessel function $I_n(x)$ for integer order n.
- double bessel_Kn (int n, double x)
 - Modified Bessel function $K_n(x)$ for integer order n.
- double bessel_logKn_recursive (int n, double x)
 - Logarithm of modified Bessel functions $K_n(x)$.
- double bessel_logKn (int n, double x)
 - Logarithm of modified Bessel function $K_n(x)$ for integer order n.
- double bessel_logIn (int n, double x)
 - Logarithm of modified Bessel function $I_n(x)$ for integer order n.

22 File Documentation

modified Bessel functions for arbitrary orders

```
• double bessel_ratiol (double nu, double x)  \textit{Calculate } I_{\nu}(x)/I_{\nu+1}(x). 
• double bessel_logInu_asymp (double nu, double x)  \textit{Compute modified Bessel function } I_{\nu}(x) \textit{ using asymptotic expansion.} 
• double bessel_logKnu_asymp (double nu, double x)  \textit{Compute modified Bessel function } K_{\nu}(x) \textit{ using asymptotic expansion.} 
• double bessel_logInu_series (double nu, double x)  \textit{Compute modified Bessel functions } I_{\nu}(x) \textit{ using series expansion.}
```

modified Bessel functions for half-integer orders

```
• void bessel_logInKn_half (int n, const double x, double *logIn_p, double *logKn_p) 
Compute modified Bessel functions of first and second kind for half-integer orders. 
• double bessel_logIn_half (int n, double x) 
Compute \log I_{n+1/2}(x). 
• double bessel_logKn_half (int n, double x) 
Compute \log K_{n+1/2}(x).
```

modified Bessel functions for orders \f\$n=0,1\f\$

```
• static double IO_coeffs []

    static double K0 coeffsA []

• static double K0 coeffsB []
static double I1_coeffs []
• static double K1_coeffsA []

    static double K1_coeffsB []

    static double chbevl (double x, double array[], int n)

      Evaluate Chebyshev series.
• double bessel 10 (double x)
      Modified Bessel function I_0(x).
• double bessel logI0 (double x)
     Logarithm of modified Bessel function I_0(x).
• double bessel_K0 (double x)
      Modified Bessel function K_0(x).
• double bessel_logK0 (double x)
     Logarithm of modified Bessel function K_0(x).

    double bessel_l1 (double x)

     Modified Bessel function I_1(x).

    double bessel_logI1 (double x)

     Logarithm of modified Bessel function I_1(x).

    double bessel_K1 (double x)

     Modified Bessel function K_1(x).

    double bessel logK1 (double x)
```

Logarithm of modified Bessel function $K_1(x)$.

5.1.1 Detailed Description

Computation of Bessel functions.

Author

Stephen L. Moshier, Cephes Math Library Release 2.8, June 2000 Michael Hartmann caps@speicherleck.de

Date

October, 2019

5.1.2 Function Documentation

```
5.1.2.1 bessel_I0()
```

```
double bessel_I0 ( double x )
```

Modified Bessel function $I_0(x)$.

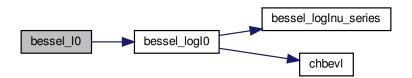
See bessel_logI0.

Parameters

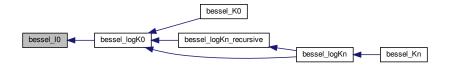
in	X	argument
----	---	----------

Return values

```
IO \mid I_0(x)
```



Here is the caller graph for this function:



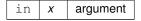
5.1.2.2 bessel_l1()

```
\begin{array}{c} \text{double bessel\_I1 (} \\ \text{double $x$)} \end{array}
```

Modified Bessel function $I_1(x)$.

See bessel_logI1.

Parameters

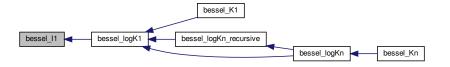


Return values

II
$$I_1(x)$$



Here is the caller graph for this function:



5.1.2.3 bessel_ln()

Modified Bessel function $I_n(x)$ for integer order n.

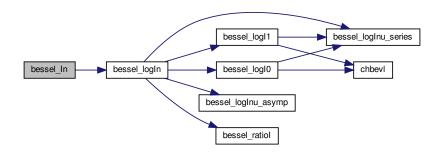
See bessel_logIn.

Parameters

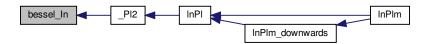
in	n	order
in	Χ	argument

Return values

In
$$I_n(x)$$



Here is the caller graph for this function:



5.1.2.4 bessel_K0()

```
double bessel_K0 ( \label{eq:condition} \mbox{double } x \mbox{ )}
```

Modified Bessel function $K_0(x)$.

See bessel_logK0.

Parameters

in	X	argument
----	---	----------

Return values



Here is the call graph for this function:



5.1.2.5 bessel_K1()

```
double bessel_K1 ( \label{eq:condition} \mbox{double } x \mbox{ )}
```

Modified Bessel function $K_1(x)$.

See bessel_logK1.

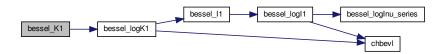
Parameters

in	Х	argument
----	---	----------

Return values

```
K1 K_1(x)
```

Here is the call graph for this function:



5.1.2.6 bessel_Kn()

```
double bessel_Kn ( \inf \ n, double x )
```

Modified Bessel function $K_n(x)$ for integer order n.

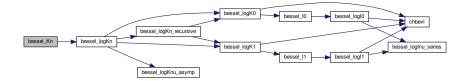
See bessel_logKn.

Parameters

in	n	order
in	X	argument

Return values





5.1.2.7 bessel_logI0()

```
double bessel_logI0 ( double x )
```

Logarithm of modified Bessel function $I_0(x)$.

- For x < 0 NAN (not a number) is returned.
- For x=0 the value $\log I_0(0)=\log(1)=0$ is returned.
- For 0 < x < 8 a series expansion is used, see bessel_logInu_series.
- For $8 \le x < 800$ a Chebychev expansion is used.
- For $x \ge 800$ the Hankel expansion

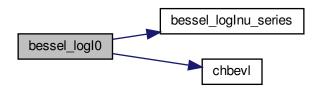
$$I_0(x) \approx \frac{e^x}{\sqrt{2\pi x}} \left(1 + k + \frac{9}{2}k^2 + \frac{225}{6}k^3 \right), \quad k = \frac{1}{8x}$$

is used.

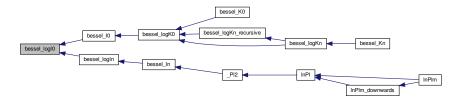
Parameters

in	X	argument
----	---	----------

Return values



Here is the caller graph for this function:



5.1.2.8 bessel_logI1()

```
double bessel_logI1 ( double x )
```

Logarithm of modified Bessel function $I_1(x)$.

- For x < 0 NAN (not a number) is returned.
- For 0 < x < 8 a series expansion is used, see bessel_logInu_series.
- For $8 \leq x < 800$ a Chebychev expansion is used.
- For $x \ge 800$ the Hankel expansion

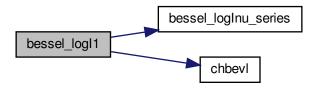
$$I_0(x) \approx \frac{e^x}{\sqrt{2\pi x}} \left(1 - 3k - \frac{15}{2}k^2 - \frac{105}{2}k^3 \right), \quad k = \frac{1}{8x}$$

is used.

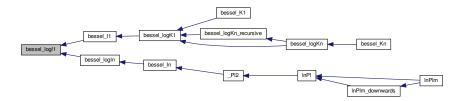
Parameters

in	Χ	argument
----	---	----------

Here is the call graph for this function:



Here is the caller graph for this function:



5.1.2.9 bessel_logIn()

```
double bessel_logIn (  \label{eq:logIn} \text{ int } n, \\  \mbox{ double } x \ )
```

Logarithm of modified Bessel function $I_n(x)$ for integer order n.

- For n=0 and n=1 the function calls <code>bessel_logI0</code> or <code>bessel_logI1</code>.
- For $n \geq 100$ an asymptotic expansion is used, see <code>bessel_logInu_asymp</code>.
- For n < 100 and $x < 5\sqrt{n}$ a series expansion is used, see <code>bessel_logInu_series</code>.
- Otherwise, the function $I_n(\boldsymbol{x})$ is computed using the recurrence relation

$$I_{n-1}(x) = I_{n+1}(x) + \frac{2n}{x}I_n(x)$$

in downwards direction using Miller's algorithm.

See also bessel_logI0, bessel_logI1, bessel_logInu_asymp, bessel_logInu_series, and bessel_ratiol.

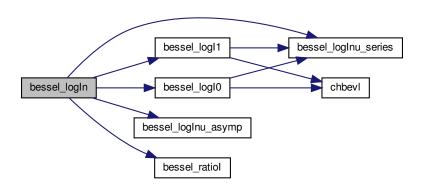
Parameters

in	n	order
in	X	argument

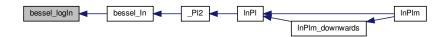
Return values

In	$\log I_n(x)$
----	---------------

Here is the call graph for this function:



Here is the caller graph for this function:



5.1.2.10 bessel_logIn_half()

Compute $\log I_{n+1/2}(x)$.

Compute logarithm of modified Bessel function of the first kind for half-integer order $I_{n+1/2}(x)$.

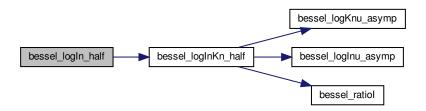
Parameters

in	n	order
in	X	argument

Return values

logI	$\log I_{n+1/2}(x)$
------	---------------------

Here is the call graph for this function:



5.1.2.11 bessel_logInKn_half()

```
void bessel_logInKn_half (
                int n,
                const double x,
                double * logIn_p,
                double * logKn_p )
```

Compute modified Bessel functions of first and second kind for half-integer orders.

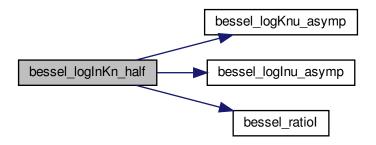
This function computes the logarithm of the modified Bessel functions $I_{n+1/2}(x)$ and $K_{n+1/2}(x)$. The results are saved in logIn_p and logKn_p.

If logIn_p or logKn_p is NULL, the variable is not referenced.

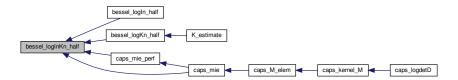
Parameters

in	n	order
in	X	argument
out	logIn⊷	pointer for $\log I_{n+1/2}(x)$
	_p	
out	logKn←	pointer for $\log K_{n+1/2}(x)$
	_p	,

Here is the call graph for this function:



Here is the caller graph for this function:



5.1.2.12 bessel_logInu_asymp()

```
double bessel_logInu_asymp ( \label{eq:condition} \mbox{double } nu, \\ \mbox{double } x \mbox{ )}
```

Compute modified Bessel function $I_{\nu}(x)$ using asymptotic expansion.

For $n \geq 100$ the asymptotic expansion is accurate.

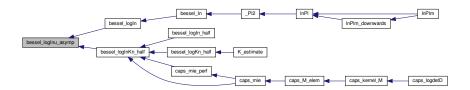
See also https://dlmf.nist.gov/10.41#ii.

Parameters

in	nu	order
in	Χ	argument

logI	$\log I_{\nu}(x)$
------	-------------------

Here is the caller graph for this function:



5.1.2.13 bessel_logInu_series()

```
double bessel_logInu_series ( \label{eq:constraint} \mbox{double } nu, \\ \mbox{double } x \mbox{ )}
```

Compute modified Bessel functions $I_{\nu}(x)$ using series expansion.

The modified Bessel function is computed using the series expansion

$$I_{\nu}(x) = \sum_{m=0}^{\infty} \frac{1}{m!\Gamma(1+m+\nu)} \left(\frac{x}{2}\right)^{2m+\nu}.$$

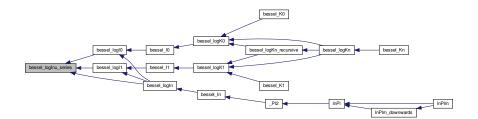
The functions succeeds for orders up to $\nu \leq 100000$ when $x \leq 10\sqrt{\nu}$. For larger values of x the function might return NAN.

Parameters

in	nu	order
in	X	argument

Return values

Int	ı	$I_{ u}(x)$ if successful or NAN otherwise
-----	---	--



5.1.2.14 bessel_logK0()

```
double bessel_logK0 ( double x )
```

Logarithm of modified Bessel function $K_0(x)$.

- For small arguments $0 < x < 10^{-8}$, the limiting form

$$K_0(x) \approx -\log(x/2) - \gamma$$

for $x\to 0$ where γ denotes the Euler-Mascheroni constant is used.

• For large arguments $x \ge 800$, the Hankel expansion

$$K_0(x) \approx \sqrt{\frac{\pi}{2x}} e^{-x} \left(1 - k + \frac{9}{2}k^2 - \frac{225}{6}k^3 \right), \quad k = \frac{1}{8x}$$

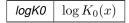
is used.

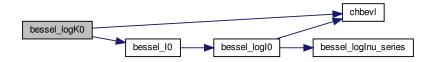
• For intermediate values, the range is partitioned into the two intervals $[10^{-8}, 2)$ and (2, 800) and Chebyshev polynomial expansions are employed in each interval.

Parameters

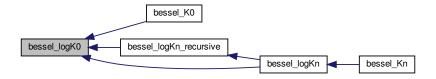
in	X	argument
----	---	----------

Return values





Here is the caller graph for this function:



5.1.2.15 bessel_logK1()

```
double bessel_logK1 ( double x )
```

Logarithm of modified Bessel function $K_1(x)$.

- For small arguments $x < 10^{-8}$, the limiting form

$$K_1(x) \approx 1/x$$

for $x \to 0$ is used.

- For large arguments $x \geq 800,$ the Hankel expansion

$$K_1(x) \approx \sqrt{\frac{\pi}{2x}} e^{-x} \left(1 + 3k - \frac{15}{2}k^2 + \frac{315}{6}k^3 \right), \quad k = \frac{1}{8x}$$

is used.

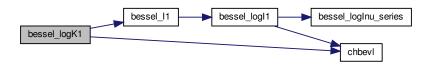
• For intermediate values, the range is partitioned into the two intervals $[10^{-8}, 8)$ and [8, 800) and Chebyshev polynomial expansions are employed in each interval.

Parameters

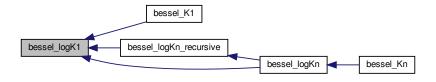
in	X	argument

logK1
$$\log K_1(x)$$

Here is the call graph for this function:



Here is the caller graph for this function:



5.1.2.16 bessel_logKn()

```
double bessel_logKn (  \label{eq:logKn}  \mbox{int } n, \\  \mbox{double } x \mbox{ )}
```

Logarithm of modified Bessel function $K_n(x)$ for integer order n.

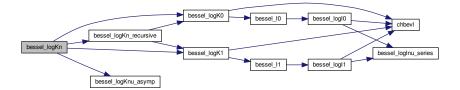
- For n=0 and n=1 the function calls <code>bessel_logK0</code> or <code>bessel_logK1</code>.
- For $n \geq 100$ an asymptotic expansion is used, see <code>bessel_logKnu_asymp</code>.
- Otherwise, the function is computed using a recursion relation, see bessel_logKn_recursive.

Parameters

in	n	order
in	Χ	argument



Here is the call graph for this function:



Here is the caller graph for this function:



5.1.2.17 bessel_logKn_half()

Compute $\log K_{n+1/2}(x)$.

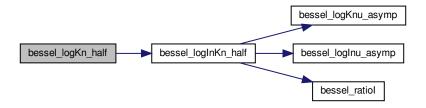
Compute logarithm of modified Bessel function of the second kind $K_{n+1/2}(x)$.

Parameters

in	n	order
in	X	argument

$$\log K \mid K_{n+1/2}(x)$$

Here is the call graph for this function:



Here is the caller graph for this function:



5.1.2.18 bessel_logKn_recursive()

Logarithm of modified Bessel functions $K_n(x)$.

The Bessel function $K_n(x)$ for integer order n is computed using the recurrence relation

$$K_{j+1}(x) = K_{j-1}(x) + \frac{2j}{x}K_j(x)$$

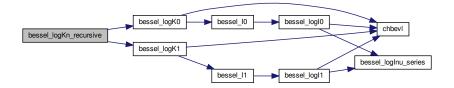
in upwards direction. The Bessel functions $K_0(x)$ and $K_1(x)$ are computed using bessel_logK0 and bessel_logK1.

Parameters

in	n	order
in	X	argument

logKn	$K_n(x)$

Here is the call graph for this function:



Here is the caller graph for this function:



5.1.2.19 bessel_logKnu_asymp()

```
double bessel_logKnu_asymp ( \label{eq:constraint} \mbox{double } nu, \\ \mbox{double } x \mbox{ )}
```

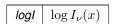
Compute modified Bessel function $K_{\nu}(x)$ using asymptotic expansion.

For $n \geq 100$ the asymptotic expansion is accurate.

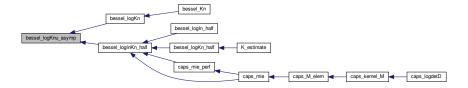
See also https://dlmf.nist.gov/10.41#ii.

Parameters

in	nu	order
in	X	argument



Here is the caller graph for this function:



5.1.2.20 bessel_ratiol()

```
double bessel_ratioI ( \label{eq:condition} \text{double } \textit{nu}, \\ \text{double } \textit{x} \ )
```

Calculate $I_{\nu}(x)/I_{\nu+1}(x)$.

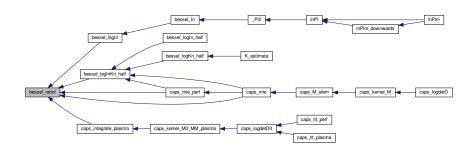
Compute the ratio of the modified Bessel functions of the first kind $I_{\nu}(x)/I_{\nu+1}(x)$ using a continued fraction, see https://dlmf.nist.gov/10.33.

Parameters

nu	order
X	argument

Return values

ratio
$$I_{\nu}(x)/I_{\nu+1}(x)$$



5.1.2.21 chbevl()

Evaluate Chebyshev series.

Evaluates the series

$$y = \sum_{i=0}^{N-1} \operatorname{coef}[i] \cdot T_i(x/2)$$

of Chebyshev polynomials T_i at argument x/2. The prime indicates that the term for i=0 has to be weighted by a factor 1/2.

Coefficients are stored in reverse order, i.e. the zero order term is last in the array. Note: n is the number of coefficients, not the order.

If coefficients are for the interval a to b, x must have been transformed to $x \to 2(2x-b-a)/(b-a)$ before entering the routine. This maps x from (a,b) to (-1,1), over which the Chebyshev polynomials are defined.

If the coefficients are for the inverted interval, in which (a,b) is mapped to (1/b,1/a), the transformation required is $x \to 2(2ab/x - b - a)/(b - a)$. If b is infinity, this becomes $x \to 4a/x - 1$.

Speed: Taking advantage of the recurrence properties of the Chebyshev polynomials, the routine requires one more addition per loop than evaluating a nested polynomial of the same degree.

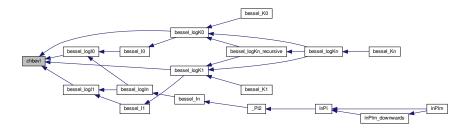
Parameters

in	X	Chebyshev series is evaluated at this point
in	array	Chebyshev coefficients
in	n	number of Chebyshev coefficients, number of elements of array

Return values

eval	Chebychev series evaluated at x

Here is the caller graph for this function:



5.1.3 Variable Documentation

5.1.3.1 I0_coeffs

```
double I0_coeffs[] [static]
```

Initial value:

```
-7.23318048787475395456E-18.
-4.83050448594418207126E-18,
4.46562142029675999901E-17.
3.46122286769746109310E-17,
-2.82762398051658348494E-16,
-3.42548561967721913462E-16,
1.77256013305652638360E-15,
3.81168066935262242075E-15,
-9.55484669882830764870E-15,
-4.15056934728722208663E-14,
1.54008621752140982691E-14,
 3.85277838274214270114E-13,
 7.18012445138366623367E-13,
-1.79417853150680611778E-12,
-1.32158118404477131188E-11.
-3.14991652796324136454E-11,
1.18891471078464383424E-11,
 4.94060238822496958910E-10,
 3.39623202570838634515E-9,
 2.26666899049817806459E-8.
2.04891858946906374183E-7.
 2.89137052083475648297E-6,
 6.88975834691682398426E-5,
 3.36911647825569408990E-3,
 8.04490411014108831608E-1
```

Chebyshev coefficients for $\exp(-x)\sqrt{x}I_0(x)$ in the inverted interval $[8,\infty]$.

$$\lim_{x \to \infty} \exp(-x)\sqrt{x}I_0(x) = 1/\sqrt{2\pi}.$$

5.1.3.2 I1_coeffs

```
double I1_coeffs[] [static]
```

Initial value:

```
7.51729631084210481353E-18,
 4.41434832307170791151E-18,
-4.65030536848935832153E-17,
-3.20952592199342395980E-17,
2.96262899764595013876E-16,
3.30820231092092828324E-16,
-1.88035477551078244854E-15,
-3.81440307243700780478E-15,
1.04202769841288027642E-14,
4.27244001671195135429E-14,
-2.10154184277266431302E-14,
-4.08355111109219731823E-13,
-7.19855177624590851209E-13,
2.03562854414708950722E-12,
 1.41258074366137813316E-11,
3.25260358301548823856E-11.
-1.89749581235054123450E-11,
-5.58974346219658380687E-10,
-3.83538038596423702205E-9,
-2.63146884688951950684E-8,
-2.51223623787020892529E-7,
-3.88256480887769039346E-6,
-1.10588938762623716291E-4,
-9.76109749136146840777E-3,
 7.78576235018280120474E-1
```

Chebyshev coefficients for $\exp(-x)\sqrt{x}I_1(x)$ in the inverted interval $[8,\infty]$.

$$\lim_{x \to \infty} \exp(-x)\sqrt{x}I_1(x) = 1/\sqrt{2\pi}.$$

5.1.3.3 K0_coeffsA

```
double K0_coeffsA[] [static]
```

Initial value:

```
1.37446543561352307156E-16,

4.25981614279661018399E-14,

1.03496952576338420167E-11,

1.90451637722020886025E-9,

2.53479107902614945675E-7,

2.28621210311945178607E-5,

1.26461541144692592338E-3,

3.59799365153615016266E-2,

3.44289899924628486886E-1,

-5.35327393233902768720E-1
```

Chebyshev coefficients for $K_0(x) + \log(x/2)I_0(x)$ in the interval [0,2]. The odd order coefficients are all zero; only the even order coefficients are listed.

$$\lim_{x \to 0} (K_0(x) + \log(x/2)I_0(x)) = -\gamma.$$

5.1.3.4 K0_coeffsB

```
double K0_coeffsB[] [static]
```

Initial value:

```
5.30043377268626276149E-18,
-1.64758043015242134646E-17,
5.21039150503902756861E-17,
-1.67823109680541210385E-16,
5.51205597852431940784E-16,
-1.84859337734377901440E-15,
6.34007647740507060557E-15,
-2.22751332699166985548E-14,
8.03289077536357521100E-14,
-2.98009692317273043925E-13,
1.14034058820847496303E-12,
-4.51459788337394416547E-12.
1.85594911495471785253E-11,
-7.95748924447710747776E-11,
3.57739728140030116597E-10,
-1.69753450938905987466E-9,
8.57403401741422608519E-9.
-4.66048989768794782956E-8.
2.76681363944501510342E-7.
-1.83175552271911948767E-6,
1.39498137188764993662E-5,
-1.28495495816278026384E-4,
1.56988388573005337491E-3,
-3.14481013119645005427E-2,
 2.44030308206595545468E0
```

Chebyshev coefficients for $\exp(x)\sqrt{x}K_0(x)$ in the inverted interval $[2,\infty]$.

$$\lim_{x \to \infty} \exp(x) \sqrt{x} K_0(x) = \sqrt{\pi/2}.$$

5.1.3.5 K1_coeffsA

```
double K1_coeffsA[] [static]
```

Initial value:

Chebyshev coefficients for $x (K_1(x) - \log(x/2)I_1(x))$ in the interval [0,2].

$$\lim_{x \to 0} x \left(K_1(x) - \log(x/2) I_1(x) \right) = 1.$$

5.1.3.6 K1_coeffsB

```
double K1_coeffsB[] [static]
```

Initial value:

```
-5.75674448366501715755E-18,
1.79405087314755922667E-17,
-5.68946255844285935196E-17,
1.83809354436663880070E-16,
-6.05704724837331885336E-16,
2.03870316562433424052E-15.
-7.01983709041831346144E-15,
2.47715442448130437068E-14,
-8.97670518232499435011E-14,
3.34841966607842919884E-13,
-1.28917396095102890680E-12,
5.13963967348173025100E-12,
-2.12996783842756842877E-11,
9.21831518760500529508E-11,
-4.19035475934189648750E-10,
2.01504975519703286596E-9,
-1.03457624656780970260E-8,
5.74108412545004946722E-8,
-3.50196060308781257119E-7,
2.40648494783721712015E-6,
-1.93619797416608296024E-5,
 1.95215518471351631108E-4,
-2.85781685962277938680E-3,
1.03923736576817238437E-1,
 2.72062619048444266945E0
```

Chebyshev coefficients for $\exp(x)\sqrt{x}K_1(x)$ in the interval $[2,\infty]$.

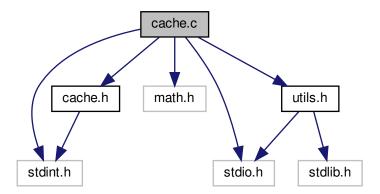
$$\lim_{x \to \infty} \exp(x) \sqrt{x} K_1(x) = \sqrt{\pi/2}.$$

5.2 cache.c File Reference

implementation of a simple cache using a hash table

```
#include <stdint.h>
#include <stdio.h>
#include <math.h>
#include "utils.h"
#include "cache.h"
```

Include dependency graph for cache.c:



Functions

• cache_t * cache_new (unsigned int entries)

Create a new cache.

void cache_free (cache_t *cache)

Free cache instance.

• void cache_insert (cache_t *cache, uint64_t key, double value)

Insert element into cache.

• double cache_lookup (cache_t *cache, uint64_t key)

Find element corresponding to key key.

5.2.1 Detailed Description

implementation of a simple cache using a hash table

Author

Michael Hartmann caps@speicherleck.de

Date

February, 2019

5.2 cache.c File Reference 47

5.2.2 Function Documentation

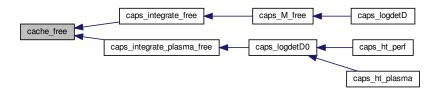
5.2.2.1 cache_free()

Free cache instance.

Parameters

cache	cache instance
-------	----------------

Here is the caller graph for this function:



5.2.2.2 cache_insert()

Insert element into cache.

Insert the element value with key key to the cache.

Parameters

cache	cache instance
key	key
value	value

Here is the caller graph for this function:



5.2.2.3 cache_lookup()

Find element corresponding to key key.

Parameters

cache	cache instance
key	key

Return values

element	if found
NAN	otherwise

Here is the caller graph for this function:



5.2.2.4 cache_new()

```
cache_t* cache_new (
          unsigned int entries )
```

Create a new cache.

Create a new cache instance.

The cache is implemented as a hash map. Collisions are not treated, the value will be overwritten.

Parameters

entries	maximum number of entries the cache can store
---------	---

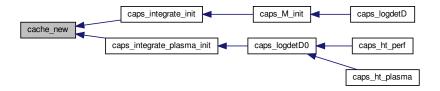
Return values



Here is the call graph for this function:



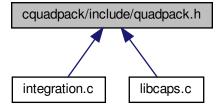
Here is the caller graph for this function:



5.3 cquadpack/include/quadpack.h File Reference

library for numerical integration of one-dimensional functions

This graph shows which files directly or indirectly include this file:



Macros

- #define GK_7_15 1
- #define GK_10_21 2
- #define GK_15_31 3
- #define GK 20 41 4
- #define GK 25 51 5
- #define GK_30_61 6

Functions

• double dqagi (double f(double, void *), double bound, int inf, double epsabs, double epsrel, double *abserr, int *neval, int *ier, void *user data)

Integration over (semi-) infinite intervals.

• double dqags (double f(double, void *), double a, double b, double epsabs, double epsrel, double *abserr, int *neval, int *ier, void *user_data)

Integration over finite intervals.

• double dqage (double f(double, void *), double a, double b, double epsabs, double epsrel, int irule, double *abserr, int *neval, int *ier, int *last, void *user_data)

5.3.1 Detailed Description

library for numerical integration of one-dimensional functions

Date

January, 2019

5.3.2 Macro Definition Documentation

```
5.3.2.1 GK_10_21
```

#define GK_10_21 2

Gauss-Kronrod 10-21 rule

5.3.2.2 GK_15_31

#define GK_15_31 3

Gauss-Kronrod 15-31 rule

5.3.2.3 GK_20_41

#define GK_20_41 4

Gauss-Kronrod 20-41 rule

```
5.3.2.4 GK_25_51

#define GK_25_51 5

Gauss-Kronrod 25-51 rule

5.3.2.5 GK_30_61

#define GK_30_61 6

Gauss-Kronrod 30-61 rule

5.3.2.6 GK_7_15

#define GK_7_15 1
```

5.3.3 Function Documentation

Gauss-Kronrod 7-15 rule

5.3.3.1 dqage()

Approximation to definite integral

Allows user's choice of Gauss-Kronrod integration rule.

error messages:

- ier=1: Maximum number of subdivisions allowed has been achieved. It is advised to analyze the integrand in order to determine the integration difficulties.
- ier=2: The occurrence of roundoff error is detected, which prevents the requested tolerance from being achieved.
- ier=3: Extremely bad integrand behaviour occurs at some points of the integration interval.
- ier=6: The input is invalid.

Parameters

in	f	double precision function to be integrated
in	а	lower limit of integration
in	b	upper limit of integration
in	epsabs	absolute accuracy requested
in	epsrel	relative accuracy requested
in	epsrel	relative accuracy requested
in	irule	integration rule to be used (GK_7_15, GK_7_15, GK_10_21, GK_15_31, GK_20_41,
		GK_25_51, or GK_30_61)
out	abserr	estimate of the modulus of the absolute error, which should equal or exceed abs(I-result)
out	neval	number of integrand evaluations
out	ier	error message; ier=0 for normal and reliable termination, otherwise ier>0
out	last	number of subintervals actually produced in the subdivision process
out	user_data	pointer that will be passed as second argument to integrand function f

Return values

result	approximation to the integral
--------	-------------------------------

5.3.3.2 dqagi()

Integration over (semi-) infinite intervals.

Adaptive integration routine which handles functions to be integrated between -infinity to +infinity, or between either of those limits and some finite, real boundary.

The adaptive strategy compares results of integration over the interval with the sum of results obtained from integration of bisected interval. Since error estimates are available from each regional integration, the interval with the largest error is bisected and new results are computed. This bisection process is continued until the error is less than the prescribed limit or convergence failure is determined.

Note that bisection, in the sense used above, refers to bisection of the transformed interval.

error messages:

• ier=0: Normal and reliable termination of the routine. It is assumed that the requested accuracy has been achieved.

- ier=1: Maximum number of subdivisions allowed has been achieved. It is advised to analyze the integrand in order to determine the integration difficulties.
- ier=2: The occurrence of roundoff error is detected, which prevents the requested tolerance from being achieved. The error may be under-estimated.
- ier=3: Extremely bad integrand behaviour occurs at some points of the integration interval.
- ier=4: The algorithm does not converge. Roundoff error is detected in the extrapolation table. It is assumed that the requested tolerance cannot be achieved, and that the returned result is the best which can be obtained.
- ier=5: The integral is probably divergent, or slowly convergent. It must be noted that divergence can occur with any other value of ier.
- ier=6: The input is invalid.

Parameters

in	f	double precision function to be integrated
in	bound	optional finite bound on integral
in	inf	specifies range of integration as follows:
		inf=-1: range is from -infinity to bound,
		 inf=+1: range is from bound to +infinity,
		 inf=+2: range is from -infinity to +infinity, (bound is ignored in this case)
in	epsabs	absolute accuracy requested
in	epsrel	relative accuracy requested
out	abserr	estimate of the modulus of the absolute error, which should equal or exceed abs(I-result)
out	neval	number of integrand evaluations
out	ier	error message; ier=0 for normal and reliable termination, otherwise ier>0
out	user_data	pointer that will be passed as second argument to integrand function f

Return values

```
result approximation to the integral
```

5.3.3.3 dqags()

Integration over finite intervals.

Adaptive integration routine which handles functions to be integrated between two finite bounds.

The adaptive strategy compares results of integration over the given interval with the sum of results obtained from integration over a bisected interval. Since error estimates are available from each regional integration, the region with the largest error is bisected and new results are computed. This bisection process is continued until the error is less than the prescribed limit or convergence failure is determined.

error messages:

- ier=1 Maximum number of subdivisions allowed has been achieved. It is advised to analyze the integrand in order to determine the integration difficulties.
- ier=2: The occurrence of roundoff error is detected, which prevents the requested tolerance from being achieved. The error may be under-estimated.
- ier=3: Extremely bad integrand behaviour occurs at some points of the integration interval.
- ier=4: The algorithm does not converge. Roundoff error is detected in the extrapolation table. It is presumed that the requested tolerance cannot be achieved, and that the returned result is the best which can be obtained.
- ier=5: The integral is probably divergent, or slowly convergent. It must be noted that divergence can occur with any other value of ier.
- ier=6: The input is invalid.

Parameters

in	f	double precision function to be integrated
in	а	lower limit of integration
in	b	upper limit of integration
in	epsabs	absolute accuracy requested
in	epsrel	relative accuracy requested
out	abserr	estimate of the modulus of the absolute error, which should equal or exceed abs(I-result)
out	neval	number of integrand evaluations
out	ier	error message; ier=0 for normal and reliable termination, otherwise ier>0
out	user_data	pointer that will be passed as second argument to integrand function f

Return values

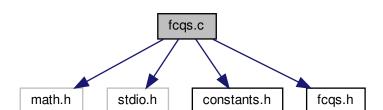
result	approximation to the integral
--------	-------------------------------



5.4 fcqs.c File Reference

exponentially convergent Fourier-Chebshev quadrature scheme (experimental)

```
#include <math.h>
#include <stdio.h>
#include "constants.h"
#include "fcqs.h"
Include dependency graph for fcqs.c:
```



Macros

- #define MMIN 5
- #define MMAX 2560

Functions

- static double cot2 (double x) Squared cotangent.
- static double wi_semiinf (double ti, double L, double N)

Weights for quadrature scheme (semiinfinite interval)

- static double wi_finite (double ti, double N)
 - Weights for quadrature scheme (infinite interval)
- double fcqs_semiinf (double f(double, void *), void *args, double *epsrel, int *neval, double L, int *ier) Integrate function f(x) over interval $[0,\infty)$.
- double fcqs_finite (double f(double, void *), void *args, double a, double b, double *epsrel, int *neval, int *ier) Integrate function f(x) over interval [a,b].

5.4.1 Detailed Description

exponentially convergent Fourier-Chebshev quadrature scheme (experimental)

Author

Michael Hartmann caps@speicherleck.de

Date

December, 2018

5.4.2 Macro Definition Documentation

5.4.2.1 MMIN

```
#define MMIN 5
```

MMIN and MMAX must be chosen in a way that there exists a positive integer k such that MMAX = MMIN * 2**k.

5.4.3 Function Documentation

5.4.3.1 cot2()

```
static double cot2 ( double x ) [static]
```

Squared cotangent.

Compute square of cotangent of x, i.e. $(\cos x/\sin x)^2$

Parameters

```
in x argument
```

Return values

```
\cot 2 \cot^2(x)
```

5.4.3.2 fcqs_finite()

Integrate function f(x) over interval [a, b].

This method uses an adaptive exponentially convergent Fourier-Chebshev quadrature to compute the integral over the interval [a,b]. The method approximately doubles the number of nodes until the desired accuracy is achieved.

Values of ier after integration:

- · ier=0: evaluation successful
- ier=1: relative accuracy epsrel must be positive
- ier=2: integrand returned NAN
- ier=3: integrand returned +inf or -inf
- ier=4: could not achieve desired accuracy

Parameters

in	f	integrand	
in	args	pointer given to f when called	
in	а	left border of integration	
in	b	right border of integration	
in,out	epsrel	epsrel on begin desired accuracy, afterwards achieved accuracy	
out	neval number of evaluations of integrand (may be set to NULL)		
out	ier	exit code	

Return values

integral numerical	value of integral
--------------------	-------------------

5.4.3.3 fcqs_semiinf()

Integrate function f(x) over interval $[0, \infty)$.

This method uses an adaptive exponentially convergent Fourier-Chebshev quadrature to compute the integral over the interval $[0, \infty)$. The method approximately doubles the number of nodes until the desired accuracy is achieved.

Values of ier after integration:

- ier=0: evaluation successful
- ier=1: relative accuracy epsrel must be positive
- ier=2: integrand returned NAN
- ier=3: integrand returned +inf or -inf
- ier=4: could not achieve desired accuracy

Parameters

in	f	integrand	
in	args	pointer given to f when called	
in,out	epsrel	on begin desired accuracy, afterwards achieved accuracy	
in	neval	eval number of evaluations of integrand (may be set to NULL)	
in	L	boosting parameter	
out	ier	exit code	

Return values

integral numerical value	of integral
--------------------------	-------------

5.4.3.4 wi_finite()

```
static double wi_finite ( \label{eq:constraint} \mbox{double } ti, \\ \mbox{double } N \;) \;\; \mbox{[static]}
```

Weights for quadrature scheme (infinite interval)

The weights correspond to (3.1e) of [1]. Here we have used that $\cos(j\pi)=(-1)^j$.

References:

• [1] Boyd, Exponentially Convergent Fourier-Chebychev Quadrature Schemes on Bounded and Infinite Intervals, Journal of Scientific Computing, Vol. 2, No. 2 (1987)

Parameters

in	ti	node
in	7	order / number of points

Return values

```
wi weight
```

5.4.3.5 wi_semiinf()

Weights for quadrature scheme (semiinfinite interval)

The weights correspond to (3.2e) of [1]. Here we have used that $\cos(j\pi) = (-1)^j$.

References:

• [1] Boyd, Exponentially Convergent Fourier-Chebychev Quadrature Schemes on Bounded and Infinite Intervals, Journal of Scientific Computing, Vol. 2, No. 2 (1987)

Parameters

in	ti	node
in	L	boosting parameter
in	N	order / number of points

Return values

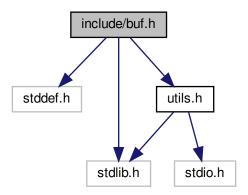
wi	weight
----	--------

5.5 include/buf.h File Reference

growable memory buffers for C99

```
#include <stddef.h>
#include <stdlib.h>
#include "utils.h"
```

Include dependency graph for buf.h:



Data Structures

struct buf

Macros

```
#define BUF_INIT_CAPACITY 32
#define buf_ptr(v) ((struct buf *)((char *)(v) - offsetof(struct buf, buffer)))
#define buf_free(v)
#define buf_size(v) ((v) ? buf_ptr((v))->size : 0)
#define buf_capacity(v) ((v) ? buf_ptr((v))->capacity : 0)
#define buf_push(v, e)
#define buf_pop(v) ((v)[--buf_ptr(v)->size])
#define buf_grow(v, n) ((v) = buf_grow1((v), sizeof(*(v)), n))
#define buf_trunc(v, n) ((v) = buf_grow1((v), sizeof(*(v)), n - buf_capacity(v)))
```

Functions

static void * buf_grow1 (void *v, size_t esize, ptrdiff_t n)

• #define buf_clear(v) ((v) ? (buf_ptr((v))->size = 0) : 0)

5.5.1 Detailed Description

growable memory buffers for C99

Author

Christopher Wellons

Date

September, 2018 This is free and unencumbered software released into the public domain.

Original version from https://github.com/skeeto/growable-buf.

Note: buf_push(), buf_grow(), buf_trunc(), and buf_free() may change the buffer pointer, and any previously-taken pointers should be considered invalidated.

Example usage:

```
float *values = 0;
for (size_t i = 0; i < 25; i++)
    buf_push(values, rand() / (float)RAND_MAX);
for (size_t i = 0; i < buf_size(values); i++)
    printf("values[%zu] = %f\n", i, values[i]);
buf_free(values);</pre>
```

5.5.2 Macro Definition Documentation

5.5.2.1 buf_capacity

return the total capacity of the buffer (size_t)

```
5.5.2.2 buf_clear
```

```
#define buf_clear(  v \ ) \ ((v) \ ? \ (buf_ptr((v))->size = 0) \ : \ 0)
```

set buffer size to 0 (for push/pop)

5.5.2.3 buf_free

```
#define buf_free( v )
```

Value:

```
do {
      if (v) {
          free(buf_ptr((v))); \
          (v) = 0; \
      } \
      while (0)
```

destroy and free the buffer

5.5.2.4 buf_grow

```
#define buf_grow(  v, \\  n ) \ ((v) = buf_growl((v), \, sizeof(*(v)), \, n))
```

increase buffer capactity by (ptrdiff_t) N elements

5.5.2.5 buf_pop

remove and return an element E from the end

5.5.2.6 buf_push

```
#define buf_push( v, e )
```

Value:

append an element E to the end

5.5.2.7 buf_size

return the number of elements in the buffer (size_t)

5.5.2.8 buf_trunc

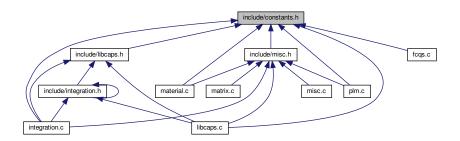
```
#define buf_trunc(  v, \\  n ) \ ((v) = buf_growl((v), sizeof(*(v)), n - buf_capacity(v)))
```

set buffer capactity to exactly (ptrdiff_t) N elements

5.6 include/constants.h File Reference

define macros and constants

This graph shows which files directly or indirectly include this file:



Macros

- #define MIN(a, b) (((a)<(b))?(a):(b))
- #define MAX(a, b) ((((a))>((b)))?((a)):((b)))
- #define SGN(val) ((0 < (val)) ((val) < 0))
- #define pow_2(x) ((x)*(x))
- #define M_PI 3.14159265358979323846
- #define M_LOG2 0.6931471805599453
- #define M_LOGPI 1.1447298858494002
- #define CAPS_hbar 1.0545718e-34
- #define CAPS_hbar_eV 6.582119514e-16
- #define CAPS_kB 1.38064852e-23
- #define CAPS_c 299792458.

Typedefs

typedef signed char sign_t

5.6.1 Detailed Description

define macros and constants

Author

Michael Hartmann caps@speicherleck.de

Date

December, 2018

5.6.2 Macro Definition Documentation

```
5.6.2.1 CAPS_c
```

#define CAPS_c 299792458.

speed of light c in vacuum [m/s]

5.6.2.2 CAPS_hbar

#define CAPS_hbar 1.0545718e-34

reduced Planck constant \hbar [m² kg / s]

5.6.2.3 CAPS_hbar_eV

#define CAPS_hbar_eV 6.582119514e-16

reduced Planck constant \hbar [eV s/rad]

5.6.2.4 CAPS_kB

#define CAPS_kB 1.38064852e-23

Boltzman constant $k_{\rm B}$ [m² kg / (K s²)]

5.6.2.5 M_LOG2

#define M_LOG2 0.6931471805599453

 $\log(2)$

```
5.6.2.6 M_LOGPI
#define M_LOGPI 1.1447298858494002
\log(\pi)
5.6.2.7 M_PI
#define M_PI 3.14159265358979323846
value for \pi=3.141592...
5.6.2.8 MAX
#define MAX(
              b ) ((((a))>((b)))?((a)):((b)))
macro to get maximum of two numbers
5.6.2.9 MIN
#define MIN(
              b) (((a)<(b))?(a):(b))
macro to get minimum of two numbers
5.6.2.10 pow_2
#define pow_2(
              x ) ((x)*(x))
compute x^2
5.6.2.11 SGN
#define SGN(
               val ) ((0 < (val)) - ((val) < 0))</pre>
```

5.6.3 Typedef Documentation

macro to get sign of numbers

5.6.3.1 sign_t

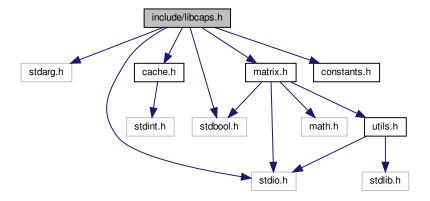
```
typedef signed char sign_t
```

define sign_t as a signed char, because "char can be either signed or unsigned depending on the implementation"

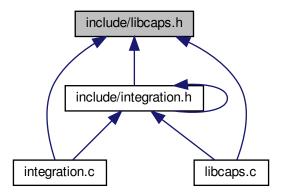
include/libcaps.h File Reference 5.7

```
#include <stdarg.h>
#include <stdbool.h>
#include <stdio.h>
#include "cache.h"
#include "matrix.h"
#include "constants.h"
```

Include dependency graph for libcaps.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- · struct caps
- · struct integration_t
- · struct integration_plasma_t
- struct caps_M_t

Macros

- #define CAPS MINIMUM LDIM 20
- #define CAPS FACTOR LDIM 5
- #define CAPS_EPSREL 1e-8
- #define CAPS_CACHE_ELEMS 10000000

Typedefs

typedef struct caps caps t

Enumerations

enum polarization_t { TE, TM }

Functions

void caps_build (FILE *stream, const char *prefix)

Print information on build to stream.

void caps_info (caps_t *self, FILE *stream, const char *prefix)

Print object information to stream.

double caps_InLambda (int I1, int I2, int m)

Calculate logarithm $\Lambda_{\ell_1\ell_2}^{(m)}$.

• caps_t * caps_init (double R, double L)

Create a new CaPS object.

void caps_free (caps_t *self)

Free memory for CaPS object.

double caps_epsilonm1_plate (caps_t *self, double xi_)

Evaluate dielectric function of the plate.

double caps_epsilonm1_sphere (caps_t *self, double xi_)

Evaluate dielectric function of the sphere.

- void caps_set_epsilonm1 (caps_t *self, double(*epsilonm1)(double xi_, void *userdata), void *userdata)

 Set dielectric function for plate and sphere.
- void caps_set_epsilonm1_plate (caps_t *self, double(*epsilonm1)(double xi_, void *userdata), void *userdata)

Set dielectric function of plate.

 void caps_set_epsilonm1_sphere (caps_t *self, double(*epsilonm1)(double xi_, void *userdata), void *userdata)

Set dielectric function of sphere.

int caps get ldim (caps t *self)

Get dimension of vector space.

int caps_set_ldim (caps_t *self, int ldim)

```
Set dimension of vector space.
```

detalg_t caps_get_detalg (caps_t *self)

Get algorithm to calculate determinant.

int caps_set_detalg (caps_t *self, detalg_t detalg)

Set algorithm to calculate deterimant.

double caps_get_epsrel (caps_t *self)

Get relative error for numerical integration.

int caps_set_epsrel (caps_t *self, double epsrel)

Set relative error for numerical integration.

• void caps_mie (caps_t *self, double xi_, int l, double *lna, double *lnb)

Return logarithm of Mie coefficients a_{ℓ} , b_{ℓ} for arbitrary metals.

void caps_mie_perf (caps_t *self, double xi_, int I, double *Ina, double *Inb)

Calculate Mie coefficients a_{ℓ} , b_{ℓ} for perfect reflectors.

• double caps_kernel_M (int i, int j, void *args_)

Kernel of round-trip matrix.

• caps_M_t * caps_M_init (caps_t *self, int m, double xi_)

Initialize caps_M_t object.

• double caps_M_elem (caps_M_t *self, int l1, int l2, char p1, char p2)

Compute matrix elements of round-trip operator.

• void caps_M_free (caps_M_t *self)

Free caps_M_t object.

double caps_logdetD (caps_t *self, double xi_, int m)

Compute $\log \det \mathcal{D}^{(m)}\left(\frac{\xi \mathcal{L}}{2}\right)$.

void caps fresnel (caps t *self, double xi , double k, double *r TE, double *r TM)

Calculate Fresnel coefficients r_{TE} and r_{TM} for arbitrary metals.

• int caps_estimate_lminmax (caps_t *self, int m, size_t *lmin_p, size_t *lmax_p)

Estimate ℓ_{\min} and ℓ_{\max} .

• double caps_epsilonm1_perf (__attribute__((unused)) double xi_, __attribute__((unused)) void *userdata)

Dielectric function for perfect reflectors.

• double caps_epsilonm1_drude (double xi_, void *userdata)

Dielectric function for Drude reflectors.

double caps_ht_drude (caps_t *caps)

Compute high-temperature limit for Drude metals.

double caps_ht_perf (caps_t *caps, double eps)

Compute free energy in the high-temperature limit for perfect reflectors.

• double caps_ht_plasma (caps_t *caps, double omegap_, double eps)

Compute free energy in the high-temperature limit for plasma model.

double caps_kernel_M0_EE (int i, int j, void *args)

Kernel for EE block.

• double caps_kernel_M0_MM (int i, int j, void *args)

Kernel for MM block.

double caps_kernel_M0_MM_plasma (int i, int j, void *args_)

Kernel for MM block (plasma model)

• void caps_logdetD0 (caps_t *self, int m, double omegap, double *EE, double *MM, double *MM_plasma)

Compute $\log \det \mathcal{D}^{(m)}(\xi=0)$ for EE and/or MM contribution.

5.7.1 Detailed Description

Author

Michael Hartmann caps@speicherleck.de

Date

February, 2019

5.7.2 Macro Definition Documentation

5.7.2.1 CAPS_CACHE_ELEMS

```
#define CAPS_CACHE_ELEMS 10000000
```

default number of elems of the cache for I integrals

5.7.2.2 CAPS_EPSREL

```
#define CAPS_EPSREL 1e-8
```

default relative error for integration

5.7.2.3 CAPS_FACTOR_LDIM

```
#define CAPS_FACTOR_LDIM 5
```

by default: Imax=ceil(5/LbyR)

5.7.2.4 CAPS_MINIMUM_LDIM

```
#define CAPS_MINIMUM_LDIM 20
```

minimum value for Imax

5.7.3 Typedef Documentation

```
5.7.3.1 caps_t
```

```
typedef struct caps caps_t
```

The CaPS object. This structure stores all essential information about temperature, geometry and the reflection properties of the mirrors.

Do not modify the attributes of the structure yourself!

5.7.4 Enumeration Type Documentation

5.7.4.1 polarization_t

```
enum polarization_t
```

typoe for polarization: either TE or TM.

5.7.5 Function Documentation

5.7.5.1 caps_build()

Print information on build to stream.

The information contains compiler, build time, git head and git branch if available. If prefix is not NULL, the string prefix will added in front of each line.

Parameters

stream	output stream	
prefix	prefix of each line or NULL	

5.7.5.2 caps_epsilonm1_drude()

Dielectric function for Drude reflectors.

Dielectric function for Drude

$$\epsilon(\xi) - 1 = \frac{\omega_{\rm P}^2}{\xi(\xi + \gamma)}$$

The parameters $\omega_{\rm P}$ and γ must be provided by userdata:

- userdata[0] = $\omega_{\rm P}$ in rad/s
- userdata[1] = γ in rad/s

Parameters

in	xi	frequency in rad/s
in	userdata	userdata

Return values

epsilon	epsilon(xi)
---------	-------------

5.7.5.3 caps_epsilonm1_perf()

Dielectric function for perfect reflectors.

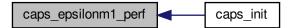
Parameters

in	xi_	ignored
in	userdata	ignored

Return values

$$\inf \quad \epsilon(\xi) = \infty$$

Here is the caller graph for this function:



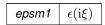
5.7.5.4 caps_epsilonm1_plate()

Evaluate dielectric function of the plate.

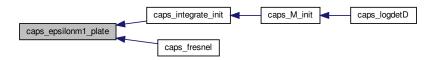
Parameters

in	self	CaPS object
in	xi⇔	$\xi \mathcal{L}/c$
	_	

Return values



Here is the caller graph for this function:



5.7.5.5 caps_epsilonm1_sphere()

```
double caps_epsilonm1_sphere (  {\it caps\_t * self,} \\ {\it double xi\_} \ )
```

Evaluate dielectric function of the sphere.

Parameters

in	self	CaPS object
in	xi⊷	$\xi \mathcal{L}/c$
	_	

Return values

```
epsm1 \epsilon(\mathrm{i}\xi)
```

Here is the caller graph for this function:



5.7.5.6 caps_estimate_lminmax()

Estimate ℓ_{\min} and ℓ_{\max} .

Estimate the vector space: The main contributions comes from the vicinity $\ell_1=\ell_2=X$ and only depend on geometry, L/R, and the quantum number m. This function calculates X using the formula in the high-temperature limit and calculates ℓ_{\min} , ℓ_{\max} .

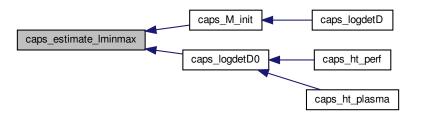
Parameters

in	self	CaPS object	
in	m	quantum number	
out	lmin←	minimum value of ℓ	
	_p		
out	lmax⊷	maximum value of ℓ	
	_p		

Return values

I approximately the value of ℓ where $\mathcal{M}^m_{\ell\ell}$ is maximal

Here is the caller graph for this function:



5.7.5.7 caps_free()

Free memory for CaPS object.

Free allocated memory for the CaPS object self.

Parameters

```
in, out self CaPS object
```

5.7.5.8 caps_fresnel()

Calculate Fresnel coefficients r_{TE} and r_{TM} for arbitrary metals.

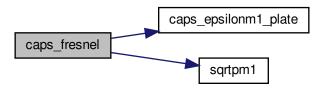
This function calculates the Fresnel coefficients $r_p=r_p(i\xi,k)$ for $p={\rm TE,TM}.$

Parameters

in	self	CaPS object	
in	xi_	$\xi \mathcal{L}/c$	
in	k_	$k\mathcal{L}$	
in,out	r_TE	Fresnel coefficient for TE mode	
in-out	r TM	Fresnel coefficient for TM mode	

Generated by Doxygen

Here is the call graph for this function:



5.7.5.9 caps_get_detalg()

Get algorithm to calculate determinant.

Parameters

```
in self CaPS object
```

Return values

```
detalg
```

5.7.5.10 caps_get_epsrel()

Get relative error for numerical integration.

See caps_set_epsrel.

Return values

epsrel	relative error

5.7.5.11 caps_get_ldim()

Get dimension of vector space.

See caps set ldim.

Parameters

in,out	self	CaPS object
--------	------	-------------

Return values

	ldim	dimension of vector space
--	------	---------------------------

5.7.5.12 caps_ht_drude()

Compute high-temperature limit for Drude metals.

For Drude metals the Fresnel coefficients become $r_{\rm TM}=1$, $r_{\rm TE}=0$ for $\xi\to0$, i.e. only the EE polarization block needs to be considered.

For Drude the free energy for $\xi=0$ can be computed analytically. We use Eq. (8) from Ref. [1] to compute the contribution.

References:

• [1] Bimonte, Emig, "Exact results for classical Casimir interactions: Dirichlet and Drude model in the sphere-sphere and sphere-plane geometry", Phys. Rev. Lett. 109 (2012), https://doi.org/10.1103/←PhysRevLett.109.160403

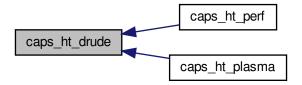
Parameters

in ca	aps C	aPS object
-------	-------	------------

Return values

 ${\it F}$ | free energy in units of $k_{
m B}T$

Here is the caller graph for this function:



5.7.5.13 caps_ht_perf()

Compute free energy in the high-temperature limit for perfect reflectors.

For perfect reflectors the Fresnel coefficients become $r_{\rm TM}=1, r_{\rm TE}=-1$ in the limit $\xi\to 0$, and only the polarization blocks EE and MM need to be considered.

The contribution for EE, i.e. Drude, can be computed analytically, see caps_ht_drude. For the MM block we numerically compute the determinants up to m=M until

$$\frac{\log \det \mathcal{D}^{(M)}(0)}{\sum_{m=0}^{M} \log \det \mathcal{D}^{(m)}(0)} < \epsilon.$$

We use Ref. [1] to compute the contribution for m=0.

References:

• [1] Bimonte, Classical Casimir interaction of perfectly conducting sphere and plate (2017), https://arxiv.org/abs/1701.06461

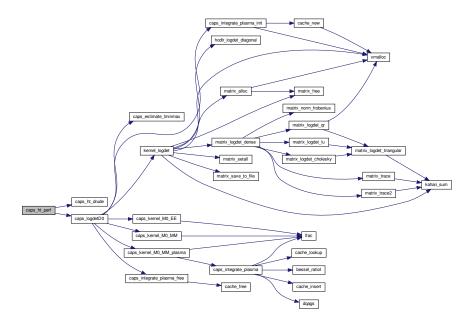
Parameters

in	caps	CaPS object
in	eps	ϵ abort criterion

Return values

<i>energy</i> free energy in units of $k_{ m B}T$

Here is the call graph for this function:



5.7.5.14 caps_ht_plasma()

Compute free energy in the high-temperature limit for plasma model.

The abort criterion eps is the same as in caps_ht_perf.

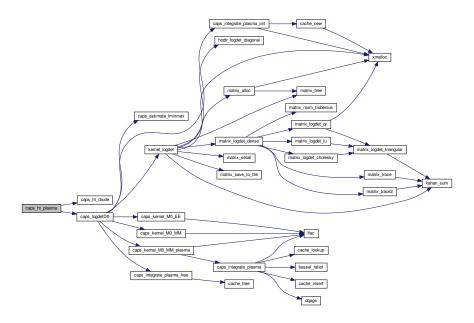
Parameters

in	caps	CaPS object
in	omegap	plasma frequency in rad/s
in	eps	abort criterion

Return values

F	free energy in units of $k_{\mathrm{B}}T$
---	---

Here is the call graph for this function:



5.7.5.15 caps_info()

Print object information to stream.

Print information about the object self to stream.

Parameters

self	CaPS object
stream where to print the string	
prefix	if prefix != NULL: start every line with the string contained in prefix

5.7.5.16 caps_init()

Create a new CaPS object.

This function will initialize a CaPS object. By default the dielectric function corresponds to perfect reflectors, i.e. $\epsilon(\xi)=\infty$.

By default, the value of $\ell_{\rm dim}$ is chosen by:

$$\ell_{\text{dim}} = \text{ceil}\left(\max\left(\text{CAPS_MINIMUM_LDIM}, \text{CAPS_FACTOR_LDIM} \cdot \frac{R}{L}\right)\right)$$

Restrictions: L/R > 0

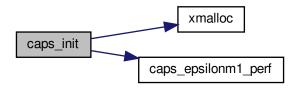
Parameters

in	R	radius of sphere in m
in	L	smallest separation between sphere and plate in m

Return values

_	CaPS object if successful
NULL	if an error occured

Here is the call graph for this function:



5.7.5.17 caps_kernel_M()

Kernel of round-trip matrix.

This function returns the matrix elements of the round-trip operator $\mathcal{M}^{(m)}.$

The round-trip matrix is a $2\ell_{\rm dim} \times 2\ell_{\rm dim}$ matrix, the matrix elements start at 0, i.e. $0 \le i, j < 2\ell_{\rm dim}$.

This function is intended to be passed as a callback to kernel_logdet. If you want to compute matrix elements of the round-trip operator, it is probably simpler to use caps_M_elem.

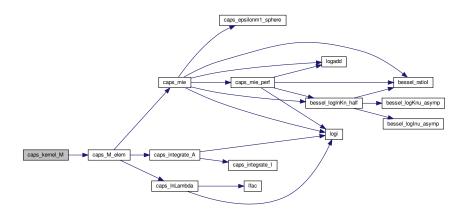
Parameters

in	i	row
in	j	column
in	args⇔	caps_M_t object, see caps_M_init

Return values

Mij
$$\mathcal{M}_{ij}^{(m)}(\xi)$$

Here is the call graph for this function:



Here is the caller graph for this function:



5.7.5.18 caps_kernel_M0_EE()

```
double caps_kernel_M0_EE (  \label{eq:caps_kernel_M0_EE}  \mbox{ int } i, \\ \mbox{int } j, \\ \mbox{void } * args_ \mbox{ )}
```

Kernel for EE block.

Function that returns matrix elements of the round-trip matrix \mathcal{M} for $\xi=0$ and polarization $p_1=p_2=\mathrm{E}$. See also caps_logdetD0.

Parameters

in	i	row (starting from 0)
in	j	column (starting from 0)
in	args⊷	pointer to caps_M_t object

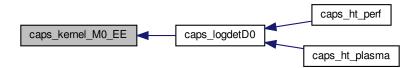
Return values

Mij	matrix element
-----	----------------

Here is the call graph for this function:



Here is the caller graph for this function:



5.7.5.19 caps_kernel_M0_MM()

Kernel for MM block.

Function that returns matrix elements of round-trip matrix \mathcal{M} for $\xi = 0$ and polarization $p_1 = p_2 = M$.

See also caps_logdetD0.

Parameters

in	i	row (starting from 0)
in	j	column (starting from 0)
in	args⊷	pointer to caps_M_t object
	_	

Return values

Mij	matrix element
-----	----------------

Here is the call graph for this function:



Here is the caller graph for this function:



5.7.5.20 caps_kernel_M0_MM_plasma()

Kernel for MM block (plasma model)

Function that returns matrix elements of round-trip matrix \mathcal{M} for $\xi=0$ and polarization $p_1=p_2=\mathrm{M}$ (plasma model).

See also caps_logdetD0.

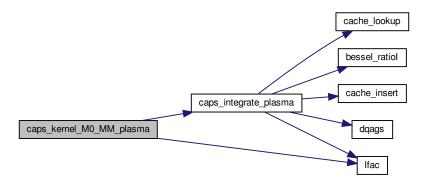
Parameters

in	i	row (starting from 0)
in	j	column (starting from 0)
in	args⊷	pointer to caps_M_t object

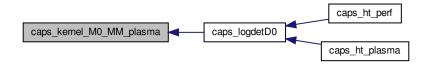
Return values

Mij	matrix element
-----	----------------

Here is the call graph for this function:



Here is the caller graph for this function:



5.7.5.21 caps_InLambda()

Calculate logarithm $\Lambda_{\ell_1\ell_2}^{(m)}$.

This function returns the logarithm of $\Lambda_{\ell_1\ell_2}^{(m)}$ for $\ell_1,\ell_2,m.$

$$\Lambda_{\ell_1,\ell_2}^{(m)} = \frac{2N_{\ell_1,m}N_{\ell_2,m}}{\sqrt{\ell_1(\ell_1+1)\ell_2(\ell_2+1)}}$$

Symmetries: $\Lambda_{\ell_1,\ell_2}^{(m)} = \Lambda_{\ell_2,\ell_1}^{(m)}$

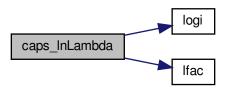
Parameters

in	11	I1>0
in	12	12>0
in	m	$m \le 11$ and $m \le 12$

Return values

InLambda	$\log \Lambda_{\ell_1,\ell_2}^{(m)}$
----------	--------------------------------------

Here is the call graph for this function:



Here is the caller graph for this function:



5.7.5.22 caps_logdetD()

Compute $\log \det \mathcal{D}^{(m)}\left(\frac{\xi \mathcal{L}}{c}\right)$.

This function computes the logarithm of the determinant of the scattering matrix for the frequency $\xi \mathcal{L}/c$ and quantum number m.

For $\xi = 0$ see caps_logdetD0.

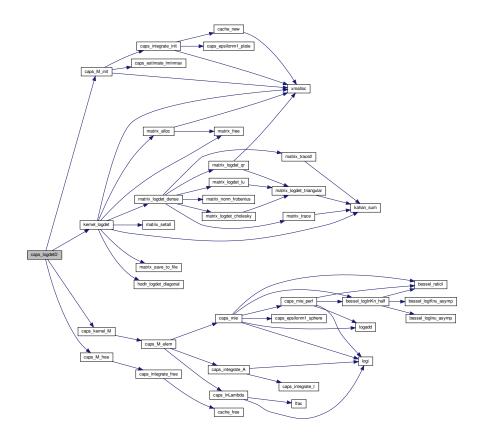
Parameters

self	CaPS object		
xi⊷	$\xi \mathcal{L}/c > 0$		
_			
m	quantum number m		

Return values



Here is the call graph for this function:



5.7.5.23 caps_logdetD0()

Compute $\log \det \mathcal{D}^{(m)}(\xi=0)$ for EE and/or MM contribution.

Compute numerically for a given value of m the contribution of the polarization block EE and/or MM. If EE, MM or MM_plasma is NULL, the value will not be computed.

For Drude metals there exists an analytical formula to compute logdetD, see caps_ht_drude.

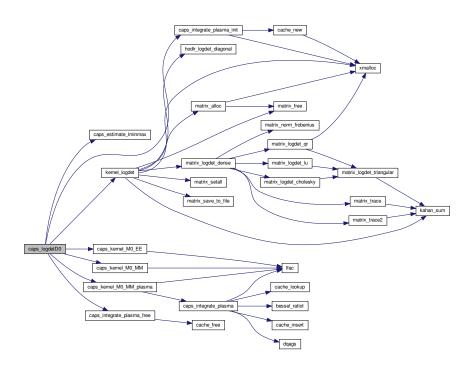
For perfect reflectors see also caps_ht_perf.

For the Plasma model see also caps_ht_plasma.

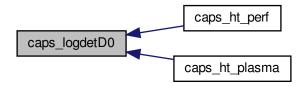
Parameters

in	self CaPS object		
in	m	quantum number m	
in	omegap	plasma frequency in rad/s (only used to compute MM_plasma)	
out	EE	pointer to store contribution for EE block	
out	MM	pointer to store contribution for MM block	
out	MM_plasma pointer to store contribution for MM block (Plasma model)		

Here is the call graph for this function:



Here is the caller graph for this function:



5.7.5.24 caps_M_elem()

Compute matrix elements of round-trip operator.

This function computes matrix elements of the round-trip operator.

Warning: Make sure that $lmin \le 11, l2 \le lmax$ or otherwise the behavior of this function is undefined. You can get lmin and lmax using $caps_estimate_lminmax$.

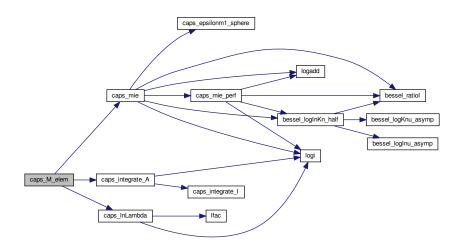
Parameters

in	self	caps_M_t object, see caps_M_init	
in	<i>l</i> 1	angular momentum ℓ_1	
in	12	angular momentum ℓ_2	
in	р1	polarization p_1 (E or M)	
in	p2	polarization p_2 (E or M)	

Return values

elem
$$\mathcal{M}_{\ell_1,\ell_2}^{(m)}(p_1,p_2)$$

Here is the call graph for this function:



Here is the caller graph for this function:

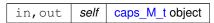


5.7.5.25 caps_M_free()

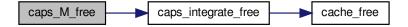
Free caps_M_t object.

Frees memory allocated by caps_M_init.

Parameters



Here is the call graph for this function:



Here is the caller graph for this function:



5.7.5.26 caps_M_init()

Initialize caps_M_t object.

This object contains all information necessary to compute the matrix elements of the round-trip operator $\mathcal{M}^{(m)}(\xi)$. It also contains a cache for the Mie coefficients.

The returned object can be given to caps_kernel_M to compute the matrix elements of the round-trip operator.

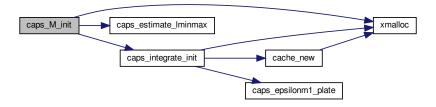
Parameters

in	caps	CaPS object	
in	m	azimuthal quantum number m	
in	xi←	$\xi \mathcal{L}/c$	
	_		

Return values

```
obj caps_M_t object that can be given to caps_kernel_M
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.7.5.27 caps_mie()

Return logarithm of Mie coefficients a_{ℓ} , b_{ℓ} for arbitrary metals.

For $\omega_P = \infty$ the Mie coefficient for perfect reflectors are returned (see caps_mie_perf).

Ina and Inb must be valid pointers.

For generic metals, we calculate the Mie coefficients a_{ℓ} und b_{ℓ} using the expressions taken from [1]. Ref. [1] is the erratum to [2]. Please note that the equations (3.30) and (3.31) in [3] are wrong. The formulas are corrected in [4].

Note: If sla \approx slb or slc \approx sld, there is a loss of significance when calculating sla-slb or slc-sld.

The signs are given by $\operatorname{sgn}(a_{\ell}) = (-1)^{\ell}$, $\operatorname{sgn}(b_{\ell}) = (-1)^{\ell+1}$.

References:

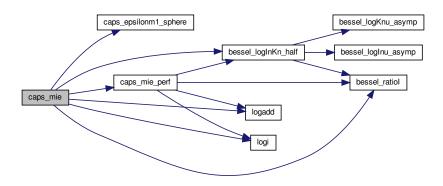
• [1] Erratum: Thermal Casimir effect for Drude metals in the plane-sphere geometry, Canaguier-Durand, Neto, Lambrecht, Reynaud (2010) http://journals.aps.org/pra/abstract/10.1103/Phys&RevA.83.039905

- [2] Thermal Casimir effect for Drude metals in the plane-sphere geometry, Canaguier-Durand, Neto, Lambrecht, Reynaud (2010), http://journals.aps.org/pra/abstract/10.1103/PhysRev← A.82.012511
- [3] Negative Casimir entropies in the plane-sphere geometry, Hartmann, 2014
- [4] Casimir effect in the plane-sphere geometry: Beyond the proximity force approximation, Hartmann, 2018

Parameters

in,out	self	CaPS object
in	xi⊷	$\xi \mathcal{L}/c$
	_	
in	1	angular momentum ℓ
out	Ina	logarithm of Mie coefficient a_ℓ
out	Inb	logarithm of Mie coefficient b_ℓ

Here is the call graph for this function:



Here is the caller graph for this function:



5.7.5.28 caps_mie_perf()

Calculate Mie coefficients $a_\ell,\,b_\ell$ for perfect reflectors.

This function calculates the logarithms of the Mie coefficients $a_\ell(i\chi)$ and $b_\ell(i\chi)$ for perfect reflectors. The Mie coefficients are evaluated at the argument $\chi=\xi R/c$.

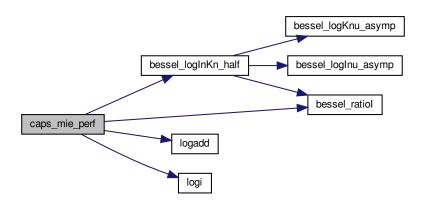
The signs are given by $\mathrm{sgn}(a_\ell) = (-1)^\ell, \, \mathrm{sgn}(b_\ell) = (-1)^{\ell+1}.$

Ina and Inb must be valid pointers and must not be NULL.

Parameters

in,out	self	CaPS object
in	xi⊷	$\xi \mathcal{L}/c > 0$
	_	
in	1	angular momentum $\ell>0$
out	Ina	logarithm of $ a_\ell $
out	Inb	logarithm of $ b_\ell $

Here is the call graph for this function:



Here is the caller graph for this function:



5.7.5.29 caps_set_detalg()

Set algorithm to calculate deterimant.

The algorithm is given by detalg. Usually you don't want to change the algorithm to compute the determinant.

detaig may be: DETALG_HODLR or DETALG_LU, DETALG_QR, DETALG_CHOLESKY.

If successul, the function returns 1. If the algorithm is not supported because of missing LAPACK support, 0 is returned.

Parameters

in,out	self	CaPS object
in	detalg	algorithm to compute determinant

Return values

	success	1 if successful, 0 if not successful
--	---------	--------------------------------------

5.7.5.30 caps_set_epsilonm1()

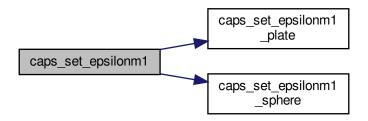
Set dielectric function for plate and sphere.

See also caps_set_epsilonm1_plate and caps_set_epsilonm1_sphere.

Parameters

in,c	out	self	CaPS object	
in		epsilonm1	callback to the function that calculates $\epsilon(\mathrm{i}\xi)-1$	
in		userdata	arbitrary pointer to data that is passwd to epsilonm1 whenever the function is called	

Here is the call graph for this function:



5.7.5.31 caps_set_epsilonm1_plate()

```
double(*)(double xi_, void *userdata) epsilonm1,
void * userdata)
```

Set dielectric function of plate.

The Fresnel coefficient r_p depend on the dielectric function $\epsilon(i\xi)$. By default, perfect reflectors with a dielectric function $\epsilon(i\xi) = \infty$ are used.

However, you can also specify an arbitrary function for $\epsilon(i\xi)$. userdata is an arbitrary pointer that will be given to the callback function

Parameters

in,out	self	CaPS object
in	epsilonm1	callback to the function that calculates $\epsilon(\mathrm{i}\xi)-1$
in	userdata	arbitrary pointer to data that is passwd to epsilonm1 whenever the function is called

Here is the caller graph for this function:



5.7.5.32 caps_set_epsilonm1_sphere()

Set dielectric function of sphere.

The Mie coefficient a_ℓ, b_ℓ depend on the dielectric function $\epsilon(i\xi)$. By default, perfect reflectors with a dielectric function $\epsilon(i\xi) = \infty$ are used.

However, you can also specify an arbitrary function for $\epsilon(i\xi)$. userdata is an arbitrary pointer that will be given to the callback function.

in,out	self	CaPS object
in	epsilonm1	callback to the function that calculates $\epsilon(\mathrm{i}\xi)-1$
in	userdata	arbitrary pointer to data that is passwd to epsilonm1 whenever the function is called

Here is the caller graph for this function:



5.7.5.33 caps_set_epsrel()

Set relative error for numerical integration.

Set relative error for numerical integration.

Parameters

in	self	CaPS object
in	epsrel	relative error

Return values

0	if an error occured
1	on success

5.7.5.34 caps_set_ldim()

Set dimension of vector space.

The round trip matrices are infinite. For a numerical evaluation the dimension has to be truncated to a finite value. The accuracy of the result depends on the truncation of the vector space. Idim determines the dimension in the angular momentum ℓ that is used.

Parameters

in,out	self	CaPS object
in	ldim	dimension in angular momentum ℓ

Return values

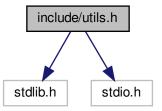
1	if successful
0	if Idim < 1

5.8 include/utils.h File Reference

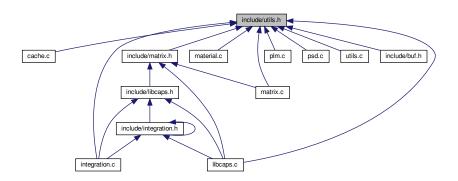
wrappers for malloc, calloc and realloc, assert-like macros, now()-function

#include <stdlib.h>
#include <stdio.h>

Include dependency graph for utils.h:



This graph shows which files directly or indirectly include this file:



Macros

```
• #define COMPILER "unknown"
```

- #define TERMINATE(cond, ...) if(cond) { fprintf(stderr, "Fatal error: "); fprintf(stderr, __VA_ARGS__←); fprintf(stderr, " (in %s, %s:%d)\n", __func__, __FILE__, __LINE__); abort(); }
- #define WARN(cond, ...) if(cond) { fprintf(stderr, "Warning: "); fprintf(stderr, __VA_ARGS__); fprintf(stderr, " (in %s, %s:%d)\n", __func__, __FILE__, __LINE__); }
- #define xfree(p) do { free(p); p = NULL; } while(0)

Functions

· double now (void)

Seconds since 01/01/1970.

void time_as_string (char *s, size_t len)

Write time into string.

void * xmalloc (size_t size)

Wrapper for malloc.

void * xrealloc (void *p, size_t size)

Wrapper for realloc.

void * xcalloc (size t nmemb, size t size)

Wrapper for calloc.

· void disable_buffering (void)

Disable buffering to stderr and stdout.

• void strrep (char *s, const char a, const char b)

Replace character by different character in string.

void strim (char *str)

Remove whitespace at beginng and end of string.

5.8.1 Detailed Description

wrappers for malloc, calloc and realloc, assert-like macros, now()-function

Author

Michael Hartmann caps@speicherleck.de

Date

July, 2017

5.8.2 Macro Definition Documentation

5.8.2.1 **COMPILER**

#define COMPILER "unknown"

name of compile

5.8.2.2 TERMINATE

Macro similar to assert that prints a warning to stderr and aborts

5.8.2.3 WARN

macro similar to assert that prints a warning to stderr

5.8.2.4 xfree

```
#define xfree( p ) do { free(p); p = NULL; } while(0)
```

macro for free that sets pointer p to NULL after freeing memory

5.8.3 Function Documentation

5.8.3.1 disable_buffering()

Disable buffering to stderr and stdout.

5.8.3.2 now()

```
double now (
```

Seconds since 01/01/1970.

This function returns the seconds since 1st Jan 1970 in μs precision.

Return values

time	seconds since 1st Jan 1970
------	----------------------------

5.8.3.3 strim()

```
void strim ( {\tt char} \ * \ str \ )
```

Remove whitespace at beginng and end of string.

If str is NULL the function doesn't do anything. Otherwise, trailing whitespace and whitespace at the beginning of the string are removed.

Parameters

```
str string
```

5.8.3.4 strrep()

Replace character by different character in string.

Replace occurence of a by b in the string s.

Parameters

in, out	s	string, terminated by \0
in	а	character to replace
in	b	substitute

5.8.3.5 time_as_string()

Write time into string.

Write current time in a human readable format into string s. The output is similar to "Aug 30 2018 14:37:35".

Parameters

s	string
len	maximum length of array s

5.8.3.6 xcalloc()

Wrapper for calloc.

This function is a wrapper for calloc. If calloc fails TERMINATE is called.

Parameters

nmemb	number of elements
size	size of each element

Return values

ptr pointer to n	nemory
------------------	--------

Here is the caller graph for this function:



5.8.3.7 xmalloc()

Wrapper for malloc.

This function is a wrapper for malloc. If malloc fails TERMINATE is called.

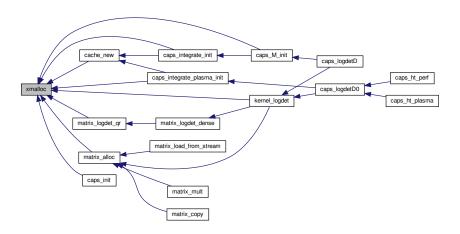
Parameters

size	size of bytes to allocate
------	---------------------------

Return values

```
ptr pointer to memory
```

Here is the caller graph for this function:



5.8.3.8 xrealloc()

Wrapper for realloc.

This function is a wrapper for realloc. If realloc fails TERMINATE is called.

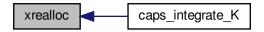
Parameters

р	ptr to old memory
size	size

Return values

newptr	pointer to new memory
--------	-----------------------

Here is the caller graph for this function:

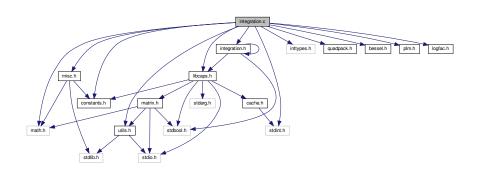


5.9 integration.c File Reference

Perform integration for arbitrary materials.

```
#include <math.h>
#include <stdint.h>
#include <inttypes.h>
#include "quadpack.h"
#include "constants.h"
#include "bessel.h"
#include "plm.h"
#include "utils.h"
#include "misc.h"
#include "libcaps.h"
#include "logfac.h"
#include "integration.h"
```

Include dependency graph for integration.c:



Data Structures

- struct integrand_t
- struct integrand_plasma_t

Macros

• #define $f(x) _f((x), nu, m, alpha)$

Functions

```
    static uint64 t hash (uint64 t l1, uint64 t l2, uint64 t p)

• static double f (double x, int nu, int m, double alpha)

    double K estimate (int nu, int m, double alpha, double eps, double *a, double *b, double *approx)

      Estimate position and width of peak.
• static double K_integrand (double x, void *args )
• static double caps integrate K (integration t *self, int nu, polarization t p, sign t *sign)

    double caps integrate K (integration t *self, int nu, polarization t p, sign t *sign)

      Compute integral \mathcal{K}_{\nu,p}^{(m)}(\alpha).
• static double _alpha (double p, double n, double nu)
  static double _caps_integrate_I (integration_t *self, int I1, int I2, polarization_t p_, sign_t *sign)
• double caps_integrate_I (integration_t *self, int I1, int I2, polarization_t p, sign_t *sign)
      Compute integral \mathcal{I}_{\ell_1,\ell_2,p}^{(m)}(\alpha).
• integration_t * caps_integrate_init (caps_t *caps, double xi_, int m, double epsrel)
      Initialize integration.

    void caps integrate free (integration t *integration)

      Free integration object.
• double caps_integrate_A (integration_t *self, int I1, int I2, polarization_t p, sign_t *sign)

    double caps integrate B (integration t *self, int I1, int I2, polarization t p, sign t *sign)

• double caps_integrate_C (integration_t *self, int I1, int I2, polarization_t p, sign_t *sign)
• double caps_integrate_D (integration_t *self, int I1, int I2, polarization_t p, sign_t *sign)
      Compute integral D_{\ell_1,\ell_2,p}^{(m)}(\xi).

    integration plasma t * caps integrate plasma init (caps t *caps, double omegap, double epsrel)

      Initialize integration object for plasma high temperature limit ( \xi = 0)
• static double _integrand_plasma (double t, void *args_)

    double caps integrate plasma (integration plasma t *self, int I1, int I2, int m, double *ratio1, double *ratio2)

      Compute integral for plasma high temperatures.
• void caps_integrate_plasma_free (integration_plasma_t *self)
      Free plasma integration object.
   Detailed Description
```

5.9.1

Perform integration for arbitrary materials.

Author

Michael Hartmann caps@speicherleck.de

Date

December, 2018

5.9.2 Function Documentation

5.9.2.1 caps_integrate_A()

Compute integral $A_{\ell_1,\ell_2,p}^{(m)}(\xi)$

Compute the integral

$$A_{\ell_1,\ell_2,p}^{(m)}(\xi) = \frac{m^2 \xi}{c} \int_0^\infty \mathrm{d}k \frac{r_p}{k \kappa} e^{-2\kappa \mathcal{L}} P_{\ell_1}^m \left(\frac{\kappa c}{\xi}\right) P_{\ell_2}^m \left(\frac{\kappa c}{\xi}\right)$$

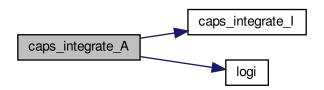
Parameters

in	self	integration object	
in	<i>l</i> 1	parameter	
in	12	parameter	
in	р	polarization; either TE or TM	
out	sign	sign of integral $\operatorname{sgn}\left(A_{\ell_1,\ell_2,p}^{(m)}(\xi)\right)$	

Return values

$$\log A \left| \log \left| A_{\ell_1,\ell_2,p}^{(m)}(\xi) \right| \right|$$

Here is the call graph for this function:



Here is the caller graph for this function:



5.9.2.2 caps_integrate_B()

Compute integral $B_{\ell_1,\ell_2,p}^{(m)}(\xi)$

Compute the integral

$$B_{\ell_1,\ell_2,p}^{(m)}(\xi) = \frac{c^3}{\xi^3} \int_0^\infty \mathrm{d}k \frac{k^3}{\kappa} r_p e^{-2\kappa \mathcal{L}} P_{\ell_1}^{m\prime}\left(\frac{\kappa c}{\xi}\right) P_{\ell_2}^{m\prime}\left(\frac{\kappa c}{\xi}\right)$$

Parameters

in	self	integration object	
in	<i>l</i> 1	parameter	
in	12	parameter	
in	р	polarization; either TE or TM	
out	sign	sign of integral $\mathrm{sgn}\left(B_{\ell_1,\ell_2,p}^{(m)}(\xi)\right)$	

Return values

$$oxed{\log B \mid \log \left| B_{\ell_1,\ell_2,p}^{(m)}(\xi)
ight|}$$

Here is the call graph for this function:

```
caps_integrate_B caps_integrate_I
```

5.9.2.3 caps_integrate_C()

Compute integral $C_{\ell_1,\ell_2,p}^{(m)}(\xi)$

Compute the integral

$$C_{\ell_1,\ell_2,p}^{(m)}(\xi) = \frac{mc}{\xi} \int_0^\infty \mathrm{d}k \frac{k}{\kappa} r_p e^{-2\kappa \mathcal{L}} P_{\ell_1}^m \left(\frac{\kappa c}{\xi}\right) P_{\ell_2}^{m'} \left(\frac{\kappa c}{\xi}\right)$$

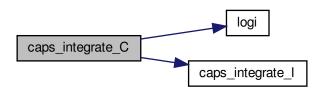
Parameters

in	self	integration object
in	<i>l</i> 1	parameter
in	12	parameter
in	р	polarization; either TE or TM
out	sign	sign of integral $\operatorname{sgn}\left(C_{\ell_1,\ell_2,p}^{(m)}(\xi)\right)$

Return values

$\log C \mid \log \left C_{\ell_1,\ell_2,p}^{(m)} \right $	(ξ)
---	---------

Here is the call graph for this function:



Here is the caller graph for this function:



5.9.2.4 caps_integrate_D()

```
double caps_integrate_D (
    integration_t * self,
    int 11,
    int 12,
    polarization_t p,
    sign_t * sign )
```

Compute integral $D_{\ell_1,\ell_2,p}^{(m)}(\xi).$

Compute

$$D_{\ell_1,\ell_2,p}^{(m)}(\xi) = C_{\ell_2,\ell_2,1}^{(m)}(\xi)$$

This function calls caps_integrate_C.

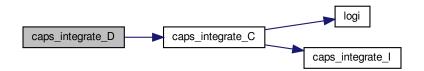
Parameters

in	self	integration object
in	<i>l</i> 1	parameter
in	12	parameter
in	р	polarization; either TE or TM
out	sign	sign of integral $\mathrm{sgn}\left(D_{\ell_1,\ell_2,p}^{(m)}(\xi)\right)$

Return values

$$\log D \left| \log \left| D_{\ell_1,\ell_2,p}^{(m)}(\xi) \right| \right|$$

Here is the call graph for this function:



5.9.2.5 caps_integrate_free()

Free integration object.

Parameters

in, out integration integration object
--

Here is the call graph for this function:



Here is the caller graph for this function:



5.9.2.6 caps_integrate_I()

Compute integral $\mathcal{I}_{\ell_1,\ell_2,p}^{(m)}(\alpha).$

Compute the integral

$$\mathcal{I}_{\ell_1,\ell_2,p}^{(m)}(\alpha) = \int_0^\infty \mathrm{d}x \, r_p \frac{e^{-\alpha x}}{x^2 - 1} P_{\ell_1}^m(x) P_{\ell_2}^m(x)$$

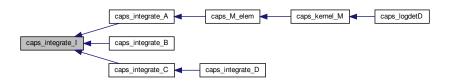
This function returns the sign of the integral and its logarithmic value.

in	self	integration object
in	<i>l</i> 1	parameter
in	12	parameter
in	р	polarization; either TE or TM
Generated	ь ў <mark>Б</mark>И худ	sign of integral $\operatorname{sgn}\left(\mathcal{I}_{\ell_1,\ell_2,p}^{(m)}(lpha) ight)$

Return values

logI
$$\left| \log \left| \mathcal{I}_{\ell_1,\ell_2,p}^{(m)}(lpha) \right| \right|$$

Here is the caller graph for this function:



5.9.2.7 caps_integrate_init()

Initialize integration.

The aspect ratio L/R and the dielectric function of the metals $\epsilon(i\xi)$ are taken from the caps object. The integration is performed to a relative accuracy of epsrel.

This function returns an object in order to compute the actual integrals. The memory of this object has to be freed after use by a call to caps_integrate_free.

The computation is sped up using caches. The number of elements of the cache for the K integrals are proportional to Idim, the elements for the I integrals are fixed. This value can be changed using the environmental variable CAPS_CACHE_ELEMS.

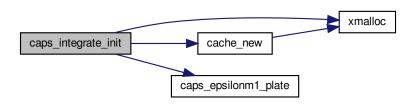
Parameters

in	caps	CaPS object
in	xi_	$\xi \mathcal{L}/c$
in	m	magnetic quantum number
in	epsrel	relative accuracy of integration

Return values

integration	object

Here is the call graph for this function:



Here is the caller graph for this function:



5.9.2.8 caps_integrate_K()

Compute integral $\mathcal{K}_{\nu,p}^{(m)}(\alpha)$.

This function solves for m>0 the integral

$$\mathcal{K}_{\nu,p}^{(m)}(\alpha) = \int_{1}^{\infty} dx \, r_{p} \frac{e^{-\alpha x}}{x^{2} - 1} P_{\nu}^{2m}(x)$$

and for m=0 the integral

$$\mathcal{K}_{\nu,p}^{(0)}(\alpha) = \int_{1}^{\infty} \mathrm{d}x \, r_p e^{-\alpha x} P_{\nu}^2(x) \,.$$

The function returns the logarithm of the value of the integral and its sign.

The projection of the wavevector onto the xy-plane is given by $k=\frac{\xi}{c}\sqrt{x^2-1}$ and $\alpha=2\xi\mathcal{L}/c$.

in	self	integration object
in	nu	parameter
Ge <u>n</u> erated	ррОоху	enpolarization, either TE or TM
out	sign	sign of $\mathcal{K}_{ u,p}^{(m)}(lpha)$

Return values

logK
$$\log \left| \mathcal{K}_{\nu,p}^{(m)}(\alpha) \right|$$

Here is the call graph for this function:



5.9.2.9 caps_integrate_plasma()

Compute integral for plasma high temperatures.

Compute the integral

$$\int_0^\infty \mathrm{d}x \, x^{\ell_1 + \ell_2} \mathrm{e}^{-x} r_{\mathrm{TE}}$$

where

$$r_{\rm TE} = \frac{\sqrt{x^2 + \beta^2} - x}{\sqrt{x^2 + \beta^2} + x}$$

and
$$\beta = 2\omega_{\rm P}(L+R)/c$$
.

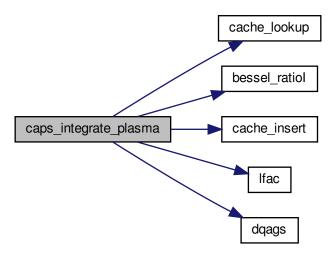
- If ratio1 is not NULL, ratio1 will be set to $I_{\ell_1-1/2}(\alpha)/I_{\ell_1+1/2}(\alpha)$ where $\alpha=2\xi(L+R)/c$.
- If ratio2 is not NULL, ratio2 will be set to $I_{\ell_2-1/2}(\alpha)/I_{\ell_2+1/2}(\alpha)$ where $\alpha=2\xi(L+R)/c$.

in	self	plasma integration object
in	11	ℓ_1
in	12	ℓ_2
in	m	m
out	ratio1	
out	ratio2	

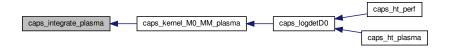
Return values

```
I value of integral
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.9.2.10 caps_integrate_plasma_free()

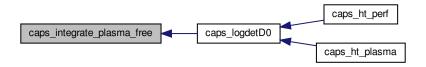
Free plasma integration object.

in,out	self	plasma integration object

Here is the call graph for this function:



Here is the caller graph for this function:



5.9.2.11 caps_integrate_plasma_init()

Initialize integration object for plasma high temperature limit ($\xi=0$)

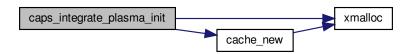
Parameters

	in	caps	CaPS object
ĺ	in	omegap	plasma frequency in rad/s
ſ	in	epsrel	relative error for integration

Return values

self	plasma integration object
------	---------------------------

Here is the call graph for this function:



Here is the caller graph for this function:



5.9.2.12 K_estimate()

```
double K_estimate (
    int nu,
    int m,
    double alpha,
    double eps,
    double * a,
    double * b,
    double * approx )
```

Estimate position and width of peak.

We want to estimate the position and the width of the peak of the integrand for $m>0\,$

$$\int_1^\infty \mathrm{d}x\, r_p \frac{e^{-\alpha x}}{x^2-1} P_\nu^{2m}(x) = \int_1^\infty \mathrm{d}x\, r_p g(x) = \int_1^\infty \mathrm{d}x\, r_p e^{-f(x)}$$
 and for $m=0$
$$\int_1^\infty \mathrm{d}x\, r_p e^{-\alpha x} P_\nu^2(x) = \int_1^\infty \mathrm{d}x\, r_p g(x) = \int_1^\infty \mathrm{d}x\, r_p e^{-f(x)}$$
 with ($m>0$)
$$f(x) = \alpha x - \log P_\nu^{2m}(x) + \log(x^2-1),$$
 and ($m=0$)
$$f(x) = \alpha x - \log P_\nu^2(x) \,.$$

We will assume that the Fresnel coefficient r_p varies slowly with respect to the width of the peak and set it to 1.

We find the maximum of f(x) using Newton's method on f'(x). With the maximum x_{max} and the second derivative at the maximum $f''(x_{\text{max}})$, we estimate the width of the peak and the value of the integral using Laplace's method:

$$\int_{1}^{\infty} \mathrm{d}x \, e^{-f(x)} \approx \sqrt{\frac{2\pi}{-f''(x_{\text{max}})}} e^{-f(x_{\text{max}})}$$

The left border a and the right border b are determined by eps, such that

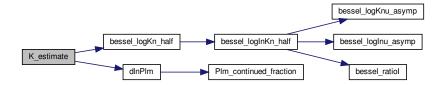
$$e^{-f(a)} \approx e^{-f(b)} \approx \epsilon e^{-f(x_{\text{max}})}$$
.

However, a cannot be smaller than 1.

Parameters

in	nu	parameter ν	
in	m	parameter m	
in	alpha	α	
in	eps	ϵ	
out	а	left border	
out	b	right border	
out	approx	logarithm of estimated value of integral	

Here is the call graph for this function:



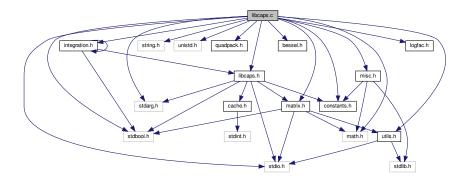
5.10 libcaps.c File Reference

library to calculate the free Casimir energy in the plane-sphere geometry

```
#include <math.h>
#include <stdarg.h>
#include <stdbool.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include "quadpack.h"
#include "constants.h"
#include "bessel.h"
#include "libcaps.h"
#include "matrix.h"
#include "logfac.h"
#include "misc.h"
```

#include "utils.h"

Include dependency graph for libcaps.c:



Functions

various functions

- double caps_InLambda (int I1, int I2, int m)
 - Calculate logarithm $\Lambda_{\ell_1\ell_2}^{(m)}$
- int caps_estimate_lminmax (caps_t *self, int m, size_t *lmin_p, size_t *lmax_p)

Estimate ℓ_{\min} and ℓ_{\max} .

Dielectric functions

- double caps_epsilonm1_plate (caps_t *self, double xi_)
 - Evaluate dielectric function of the plate.
- double caps_epsilonm1_sphere (caps_t *self, double xi_)
 - Evaluate dielectric function of the sphere.
- double caps_epsilonm1_perf (__attribute__((unused)) double xi_, __attribute__((unused)) void *userdata) Dielectric function for perfect reflectors.
- double caps epsilonm1 drude (double xi, void *userdata)

Dielectric function for Drude reflectors.

initialization and setting parameters

- caps t * caps init (double R, double L)
 - Create a new CaPS object.
- void caps_free (caps_t *self)

Free memory for CaPS object.

void caps_build (FILE *stream, const char *prefix)

Print information on build to stream.

void caps_info (caps_t *self, FILE *stream, const char *prefix)

Print object information to stream.

• int caps_set_epsrel (caps_t *self, double epsrel)

Set relative error for numerical integration.

• double caps_get_epsrel (caps_t *self)

Get relative error for numerical integration.

- void caps_set_epsilonm1 (caps_t *self, double(*epsilonm1)(double xi_, void *userdata), void *userdata)

 Set dielectric function for plate and sphere.
- void caps_set_epsilonm1_plate (caps_t *self, double(*epsilonm1)(double xi_, void *userdata), void *userdata)

Set dielectric function of plate.

 void caps_set_epsilonm1_sphere (caps_t *self, double(*epsilonm1)(double xi_, void *userdata), void *userdata)

Set dielectric function of sphere.

• int caps_set_detalg (caps_t *self, detalg_t detalg)

Set algorithm to calculate deterimant.

detaig_t caps_get_detaig (caps_t *self)

Get algorithm to calculate determinant.

int caps set ldim (caps t *self, int ldim)

Set dimension of vector space.

int caps_get_ldim (caps_t *self)

Get dimension of vector space.

Mie and Fresnell coefficients

• void caps_mie_perf (caps_t *self, double xi_, int I, double *lna, double *lnb)

Calculate Mie coefficients a_{ℓ} , b_{ℓ} for perfect reflectors.

void caps mie (caps t *self, double xi , int l, double *lna, double *lnb)

Return logarithm of Mie coefficients a_{ℓ} , b_{ℓ} for arbitrary metals.

• void caps_fresnel (caps_t *self, double xi_, double k_, double *r_TE, double *r_TM)

Calculate Fresnel coefficients r_{TE} and r_{TM} for arbitrary metals.

Kernels

• caps M t * caps M init (caps t *caps, int m, double xi)

Initialize caps M t object.

double caps_kernel_M (int i, int j, void *args_)

Kernel of round-trip matrix.

double caps_M_elem (caps_M_t *self, int I1, int I2, char p1, char p2)

Compute matrix elements of round-trip operator.

void caps_M_free (caps_M_t *self)

Free caps M t object.

double caps kernel M0 EE (int i, int j, void *args)

Kernel for EE block.

double caps_kernel_M0_MM_plasma (int i, int j, void *args_)

Kernel for MM block (plasma model)

• double caps_kernel_M0_MM (int i, int j, void *args_)

Kernel for MM block.

Compute determinants

- double caps_logdetD (caps_t *self, double xi_, int m)

Compute $\log \det \mathcal{D}^{(m)}\left(\frac{\xi \mathcal{L}}{\epsilon}\right)$.

• void caps_logdetD0 (caps_t *self, int m, double omegap, double *EE, double *MM, double *MM_plasma) Compute $\log \det \mathcal{D}^{(m)}(\xi=0)$ for EE and/or MM contribution.

high-temperature limit

double caps_ht_drude (caps_t *caps)

Compute high-temperature limit for Drude metals.

double caps_ht_perf (caps_t *caps, double eps)

Compute free energy in the high-temperature limit for perfect reflectors.

double caps_ht_plasma (caps_t *caps, double omegap, double eps)

Compute free energy in the high-temperature limit for plasma model.

5.10.1 Detailed Description

library to calculate the free Casimir energy in the plane-sphere geometry

Author

Michael Hartmann caps@speicherleck.de

Date

December, 2017

5.10.2 Function Documentation

5.10.2.1 caps_build()

Print information on build to stream.

The information contains compiler, build time, git head and git branch if available. If prefix is not NULL, the string prefix will added in front of each line.

Parameters

stream	output stream	
prefix	prefix of each line or NULL	

5.10.2.2 caps_epsilonm1_drude()

```
double caps_epsilonm1_drude ( \label{eq:caps_epsilonm1} \mbox{double $xi$,} \\ \mbox{void} * \mbox{userdata} )
```

Dielectric function for Drude reflectors.

Dielectric function for Drude

$$\epsilon(\xi) - 1 = \frac{\omega_{\rm P}^2}{\xi(\xi + \gamma)}$$

The parameters ω_P and γ must be provided by userdata:

- userdata[0] = ω_{P} in rad/s
- userdata[1] = γ in rad/s

Parameters

in	xi	frequency in rad/s
in	userdata	userdata

Return values

```
epsilon epsilon(xi)
```

5.10.2.3 caps_epsilonm1_perf()

```
double caps_epsilonm1_perf (
    __attribute__((unused)) double xi_,
    __attribute__((unused)) void * userdata )
```

Dielectric function for perfect reflectors.

Parameters

in	xi_	ignored
in	userdata	ignored

Return values

$$\quad \quad \textit{inf} \quad \epsilon(\xi) = \infty$$

Here is the caller graph for this function:



5.10.2.4 caps_epsilonm1_plate()

Evaluate dielectric function of the plate.

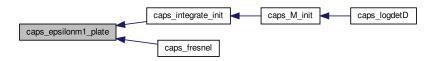
Parameters

in	self	CaPS object
in	xi⊷	$\xi \mathcal{L}/c$
	_	

Return values

```
epsm1 \epsilon(\mathrm{i}\xi)
```

Here is the caller graph for this function:



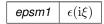
5.10.2.5 caps_epsilonm1_sphere()

Evaluate dielectric function of the sphere.

Parameters

in	self	CaPS object
in	xi⊷	$\xi \mathcal{L}/c$

Return values



Here is the caller graph for this function:



5.10.2.6 caps_estimate_lminmax()

Estimate ℓ_{min} and $\ell_{max}.$

Estimate the vector space: The main contributions comes from the vicinity $\ell_1=\ell_2=X$ and only depend on geometry, L/R, and the quantum number m. This function calculates X using the formula in the high-temperature limit and calculates ℓ_{\min} , ℓ_{\max} .

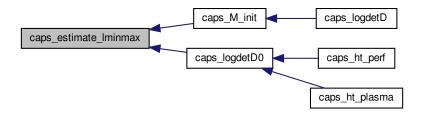
Parameters

in	self	CaPS object	
in	m	quantum number	
out	lmin←	minimum value of ℓ	
	_p		
out	_p Imax⊷	maximum value of ℓ	

Return values

```
I approximately the value of \ell where \mathcal{M}^m_{\ell\ell} is maximal
```

Here is the caller graph for this function:



5.10.2.7 caps_free()

Free memory for CaPS object.

Free allocated memory for the CaPS object self.

Parameters

in,out	self	CaPS object
--------	------	-------------

5.10.2.8 caps_fresnel()

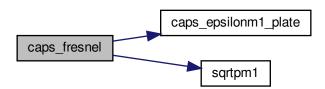
Calculate Fresnel coefficients $r_{\rm TE}$ and $r_{\rm TM}$ for arbitrary metals.

This function calculates the Fresnel coefficients $r_p=r_p(i\xi,k)$ for $p={\rm TE,TM}.$

Parameters

in	self	CaPS object
in	xi_	$\xi \mathcal{L}/c$
in	k_	$k\mathcal{L}$
in,out	r_TE	Fresnel coefficient for TE mode
in,out	r_TM	Fresnel coefficient for TM mode

Here is the call graph for this function:



5.10.2.9 caps_get_detalg()

Get algorithm to calculate determinant.

Parameters

in self CaPS object

Return values

```
detalg
```

5.10.2.10 caps_get_epsrel()

Get relative error for numerical integration.

See caps_set_epsrel.

Return values

5.10.2.11 caps_get_ldim()

Get dimension of vector space.

See caps_set_ldim.

Parameters

```
in, out self CaPS object
```

Return values

```
Idim dimension of vector space
```

5.10.2.12 caps_ht_drude()

Compute high-temperature limit for Drude metals.

For Drude metals the Fresnel coefficients become $r_{\rm TM}=1$, $r_{\rm TE}=0$ for $\xi\to0$, i.e. only the EE polarization block needs to be considered.

For Drude the free energy for $\xi=0$ can be computed analytically. We use Eq. (8) from Ref. [1] to compute the contribution.

References:

• [1] Bimonte, Emig, "Exact results for classical Casimir interactions: Dirichlet and Drude model in the sphere-sphere and sphere-plane geometry", Phys. Rev. Lett. 109 (2012), https://doi.org/10.1103/←PhysRevLett.109.160403

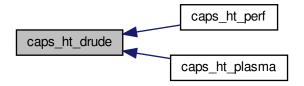
Parameters

```
in caps CaPS object
```

Return values

```
{\it F} \mid free energy in units of k_{
m B}T
```

Here is the caller graph for this function:



5.10.2.13 caps_ht_perf()

Compute free energy in the high-temperature limit for perfect reflectors.

For perfect reflectors the Fresnel coefficients become $r_{\rm TM}=1,\,r_{\rm TE}=-1$ in the limit $\xi\to0$, and only the polarization blocks EE and MM need to be considered.

The contribution for EE, i.e. Drude, can be computed analytically, see caps_ht_drude. For the MM block we numerically compute the determinants up to m=M until

$$\frac{\log \det \mathcal{D}^{(M)}(0)}{\sum_{m=0}^{M} ' \log \det \mathcal{D}^{(m)}(0)} < \epsilon \,.$$

We use Ref. [1] to compute the contribution for m=0.

References:

• [1] Bimonte, Classical Casimir interaction of perfectly conducting sphere and plate (2017), https://arxiv.org/abs/1701.06461

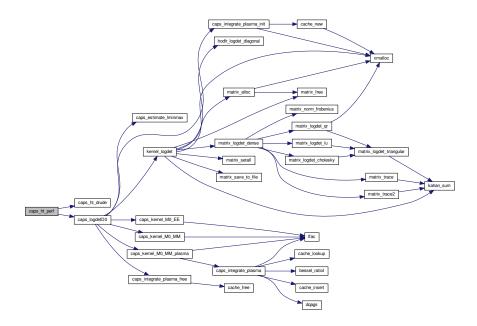
Parameters

in	caps	CaPS object	
in	eps	ϵ abort criterion	

Return values

energy	free energy in units of $k_{ m B}T$
--------	-------------------------------------

Here is the call graph for this function:



5.10.2.14 caps_ht_plasma()

```
double omegap,
double eps )
```

Compute free energy in the high-temperature limit for plasma model.

The abort criterion eps is the same as in caps_ht_perf.

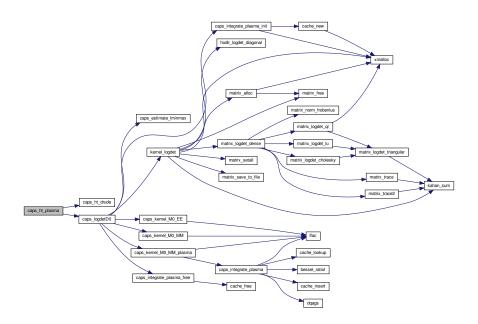
Parameters

in	caps	CaPS object
in	omegap	plasma frequency in rad/s
in	eps	abort criterion

Return values

F free energy in units of k	$\epsilon_{ m B}T$
-----------------------------	--------------------

Here is the call graph for this function:



5.10.2.15 caps_info()

Print object information to stream.

Print information about the object self to stream.

Parameters

self	CaPS object	
stream	where to print the string	
prefix if prefix != NULL: start every line with the string contained in		

5.10.2.16 caps_init()

Create a new CaPS object.

This function will initialize a CaPS object. By default the dielectric function corresponds to perfect reflectors, i.e. $\epsilon(\xi)=\infty$.

By default, the value of $\ell_{\rm dim}$ is chosen by:

$$\ell_{\text{dim}} = \text{ceil}\left(\text{max}\left(\text{CAPS_MINIMUM_LDIM}, \text{CAPS_FACTOR_LDIM} \cdot \frac{R}{L}\right)\right)$$

Restrictions: L/R > 0

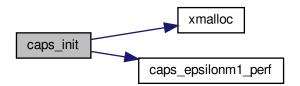
Parameters

in	R	radius of sphere in m
in	L	smallest separation between sphere and plate in m

Return values

object	CaPS object if successful
NULL	if an error occured

Here is the call graph for this function:



5.10.2.17 caps_kernel_M()

Kernel of round-trip matrix.

This function returns the matrix elements of the round-trip operator $\mathcal{M}^{(m)}.$

The round-trip matrix is a $2\ell_{\rm dim} \times 2\ell_{\rm dim}$ matrix, the matrix elements start at 0, i.e. $0 \le i,j < 2\ell_{\rm dim}$.

This function is intended to be passed as a callback to kernel_logdet. If you want to compute matrix elements of the round-trip operator, it is probably simpler to use caps_M_elem.

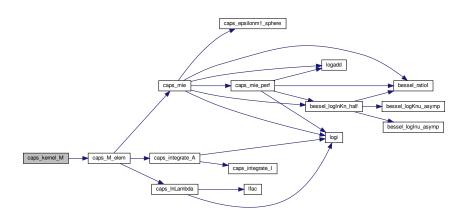
Parameters

in	i	row
in	j	column
in	args⊷	caps_M_t object, see caps_M_init
	_	

Return values

Mij
$$\mathcal{M}_{ij}^{(m)}(\xi)$$

Here is the call graph for this function:



Here is the caller graph for this function:



5.10.2.18 caps_kernel_M0_EE()

Kernel for EE block.

Function that returns matrix elements of the round-trip matrix \mathcal{M} for $\xi=0$ and polarization $p_1=p_2=\mathrm{E}$.

See also caps_logdetD0.

Parameters

in	i	row (starting from 0)
in	j	column (starting from 0)
in	args⊷	pointer to caps_M_t object
	_	

Return values



Here is the call graph for this function:



Here is the caller graph for this function:



5.10.2.19 caps_kernel_M0_MM()

Kernel for MM block.

Function that returns matrix elements of round-trip matrix \mathcal{M} for $\xi=0$ and polarization $p_1=p_2=\mathrm{M}$.

See also caps_logdetD0.

Parameters

in	i	row (starting from 0)
in	j	column (starting from 0)
in	args⊷	pointer to caps_M_t object

Return values

Mij	matrix element

Here is the call graph for this function:



Here is the caller graph for this function:



5.10.2.20 caps_kernel_M0_MM_plasma()

Kernel for MM block (plasma model)

Function that returns matrix elements of round-trip matrix \mathcal{M} for $\xi=0$ and polarization $p_1=p_2=\mathrm{M}$ (plasma model).

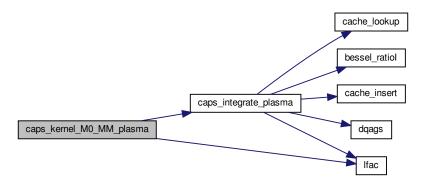
See also caps_logdetD0.

Parameters

in	i	row (starting from 0)
in	j	column (starting from 0)
in	args⇔	pointer to caps_M_t object
	_	

Mij	matrix element
-----	----------------

Here is the call graph for this function:



Here is the caller graph for this function:



5.10.2.21 caps_InLambda()

Calculate logarithm $\Lambda_{\ell_1\ell_2}^{(m)}.$

This function returns the logarithm of $\Lambda_{\ell_1\ell_2}^{(m)}$ for $\ell_1,\ell_2,m.$

$$\Lambda_{\ell_1,\ell_2}^{(m)} = \frac{2N_{\ell_1,m}N_{\ell_2,m}}{\sqrt{\ell_1(\ell_1+1)\ell_2(\ell_2+1)}}$$

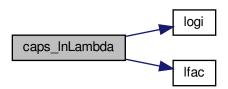
Symmetries: $\Lambda_{\ell_1,\ell_2}^{(m)} = \Lambda_{\ell_2,\ell_1}^{(m)}$

in	<i>l</i> 1	l1>0
in	12	12>0
in	m	$m \le 11$ and $m \le 12$

Return values

InLambda	$\log \Lambda_{\ell_1,\ell_2}^{(m)}$
----------	--------------------------------------

Here is the call graph for this function:



Here is the caller graph for this function:



5.10.2.22 caps_logdetD()

Compute $\log \det \mathcal{D}^{(m)}\left(\frac{\xi \mathcal{L}}{c}\right)$.

This function computes the logarithm of the determinant of the scattering matrix for the frequency $\xi \mathcal{L}/c$ and quantum number m.

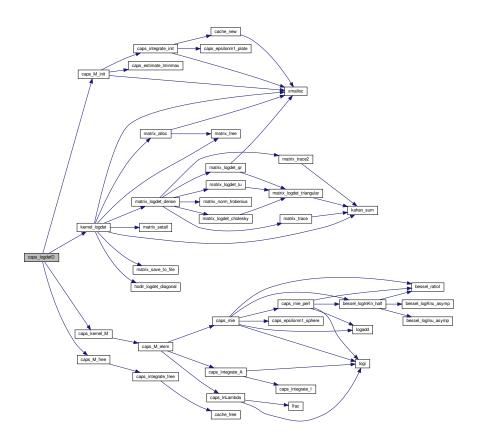
For $\xi = 0$ see caps_logdetD0.

self	CaPS object
xi←	$\xi \mathcal{L}/c > 0$
_	
m	quantum number m

Return values

logdetD

Here is the call graph for this function:



5.10.2.23 caps_logdetD0()

Compute $\log \det \mathcal{D}^{(m)}(\xi=0)$ for EE and/or MM contribution.

Compute numerically for a given value of m the contribution of the polarization block EE and/or MM. If EE, MM or MM_plasma is NULL, the value will not be computed.

For Drude metals there exists an analytical formula to compute logdetD, see caps_ht_drude.

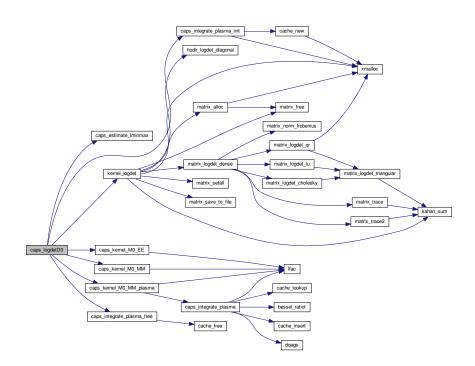
For perfect reflectors see also caps_ht_perf.

For the Plasma model see also caps_ht_plasma.

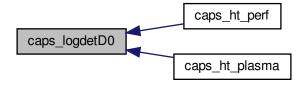
Parameters

in	self	CaPS object
in	т	quantum number m
in	omegap	plasma frequency in rad/s (only used to compute MM_plasma)
out	EE pointer to store contribution for EE block	
out	MM	pointer to store contribution for MM block
out	MM_plasma	pointer to store contribution for MM block (Plasma model)

Here is the call graph for this function:



Here is the caller graph for this function:



5.10.2.24 caps_M_elem()

Compute matrix elements of round-trip operator.

This function computes matrix elements of the round-trip operator.

Warning: Make sure that $lmin \le 11, l2 \le lmax$ or otherwise the behavior of this function is undefined. You can get lmin and lmax using $caps_estimate_lminmax$.

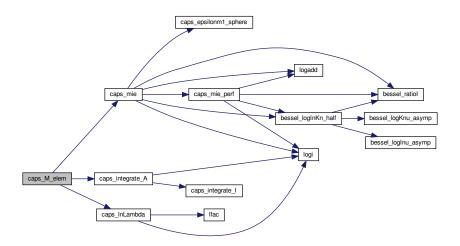
Parameters

in	self	caps_M_t object, see caps_M_init
in	<i>l</i> 1	angular momentum ℓ_1
in	12	angular momentum ℓ_2
in	p1	polarization p_1 (E or M)
in	p2	polarization p_2 (E or M)

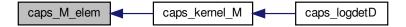
Return values

elem	$\mathcal{M}_{\ell_1,\ell_2}^{(m)}(p_1,p_2)$
------	--

Here is the call graph for this function:



Here is the caller graph for this function:



5.10.2.25 caps_M_free()

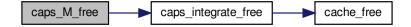
Free caps_M_t object.

Frees memory allocated by caps_M_init.

Parameters



Here is the call graph for this function:



Here is the caller graph for this function:



5.10.2.26 caps_M_init()

Initialize caps_M_t object.

This object contains all information necessary to compute the matrix elements of the round-trip operator $\mathcal{M}^{(m)}(\xi)$. It also contains a cache for the Mie coefficients.

The returned object can be given to caps_kernel_M to compute the matrix elements of the round-trip operator.

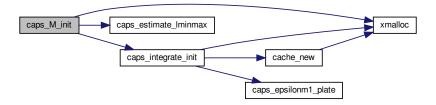
Parameters

in	caps	CaPS object
in	m	azimuthal quantum number m
in	xi⊷ –	$\xi \mathcal{L}/c$

Return values

```
obj caps_M_t object that can be given to caps_kernel_M
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.10.2.27 caps_mie()

Return logarithm of Mie coefficients a_{ℓ} , b_{ℓ} for arbitrary metals.

For $\omega_P = \infty$ the Mie coefficient for perfect reflectors are returned (see caps_mie_perf).

Ina and Inb must be valid pointers.

For generic metals, we calculate the Mie coefficients a_{ℓ} und b_{ℓ} using the expressions taken from [1]. Ref. [1] is the erratum to [2]. Please note that the equations (3.30) and (3.31) in [3] are wrong. The formulas are corrected in [4].

Note: If $sla \approx slb$ or $slc \approx sld$, there is a loss of significance when calculating sla-slb or slc-sld.

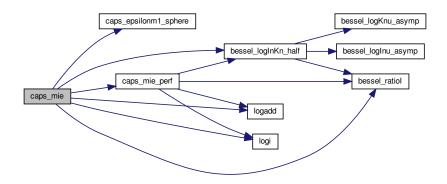
```
The signs are given by \operatorname{sgn}(a_{\ell}) = (-1)^{\ell}, \operatorname{sgn}(b_{\ell}) = (-1)^{\ell+1}.
```

References:

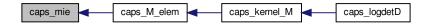
- [1] Erratum: Thermal Casimir effect for Drude metals in the plane-sphere geometry, Canaguier-Durand, Neto, Lambrecht, Reynaud (2010) http://journals.aps.org/pra/abstract/10.1103/Physcore.83.039905
- [2] Thermal Casimir effect for Drude metals in the plane-sphere geometry, Canaguier-Durand, Neto, Lambrecht, Reynaud (2010), http://journals.aps.org/pra/abstract/10.1103/PhysRev← A.82.012511
- [3] Negative Casimir entropies in the plane-sphere geometry, Hartmann, 2014
- [4] Casimir effect in the plane-sphere geometry: Beyond the proximity force approximation, Hartmann, 2018

in,out	self	CaPS object
in	xi⊷	$\xi \mathcal{L}/c$
	_	
in	1	angular momentum ℓ
out	Ina	logarithm of Mie coefficient a_ℓ
out	Inb	logarithm of Mie coefficient b_ℓ

Here is the call graph for this function:



Here is the caller graph for this function:



5.10.2.28 caps_mie_perf()

Calculate Mie coefficients $a_\ell,\,b_\ell$ for perfect reflectors.

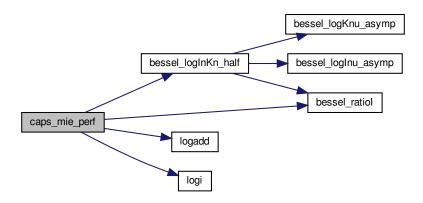
This function calculates the logarithms of the Mie coefficients $a_{\ell}(i\chi)$ and $b_{\ell}(i\chi)$ for perfect reflectors. The Mie coefficients are evaluated at the argument $\chi=\xi R/c$.

The signs are given by $\operatorname{sgn}(a_\ell) = (-1)^\ell$, $\operatorname{sgn}(b_\ell) = (-1)^{\ell+1}$.

Ina and Inb must be valid pointers and must not be NULL.

in,out	self	CaPS object
in	xi⊷	$\xi \mathcal{L}/c > 0$
	_	
in	1	angular momentum $\ell>0$
out	Ina	logarithm of $ a_\ell $
out	Inb	logarithm of $ b_\ell $

Here is the call graph for this function:



Here is the caller graph for this function:



5.10.2.29 caps_set_detalg()

Set algorithm to calculate deterimant.

The algorithm is given by detalg. Usually you don't want to change the algorithm to compute the determinant.

detaig may be: DETALG_HODLR or DETALG_LU, DETALG_QR, DETALG_CHOLESKY.

If successul, the function returns 1. If the algorithm is not supported because of missing LAPACK support, 0 is returned.

in,out	self	CaPS object
in	detalg	algorithm to compute determinant

Return values

success	1 if successful, 0 if not successful
---------	--------------------------------------

5.10.2.30 caps_set_epsilonm1()

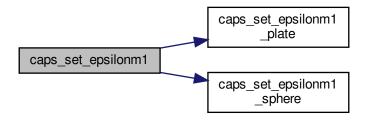
Set dielectric function for plate and sphere.

See also caps_set_epsilonm1_plate and caps_set_epsilonm1_sphere.

Parameters

in,out	self	CaPS object
in	epsilonm1	callback to the function that calculates $\epsilon(\mathrm{i}\xi)-1$
in	userdata	arbitrary pointer to data that is passwd to epsilonm1 whenever the function is called

Here is the call graph for this function:



5.10.2.31 caps_set_epsilonm1_plate()

Set dielectric function of plate.

The Fresnel coefficient r_p depend on the dielectric function $\epsilon(\mathrm{i}\xi)$. By default, perfect reflectors with a dielectric function $\epsilon(\mathrm{i}\xi)=\infty$ are used.

However, you can also specify an arbitrary function for $\epsilon(i\xi)$. userdata is an arbitrary pointer that will be given to the callback function.

Parameters

in,out	self	CaPS object
in	epsilonm1	callback to the function that calculates $\epsilon(\mathrm{i}\xi)-1$
in	userdata	arbitrary pointer to data that is passwd to epsilonm1 whenever the function is called

Here is the caller graph for this function:



5.10.2.32 caps_set_epsilonm1_sphere()

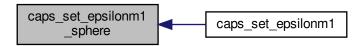
Set dielectric function of sphere.

The Mie coefficient a_ℓ, b_ℓ depend on the dielectric function $\epsilon(i\xi)$. By default, perfect reflectors with a dielectric function $\epsilon(i\xi) = \infty$ are used.

However, you can also specify an arbitrary function for $\epsilon(i\xi)$. userdata is an arbitrary pointer that will be given to the callback function.

in,out	self	CaPS object
in	epsilonm1	callback to the function that calculates $\epsilon(\mathrm{i}\xi)-1$
in	userdata	arbitrary pointer to data that is passwd to epsilonm1 whenever the function is called

Here is the caller graph for this function:



5.10.2.33 caps_set_epsrel()

Set relative error for numerical integration.

Set relative error for numerical integration.

Parameters

in	self	CaPS object
in	epsrel	relative error

Return values

0	if an error occured
1	on success

5.10.2.34 caps_set_ldim()

Set dimension of vector space.

The round trip matrices are infinite. For a numerical evaluation the dimension has to be truncated to a finite value. The accuracy of the result depends on the truncation of the vector space. Idim determines the dimension in the angular momentum ℓ that is used.

Parameters

in,out	self	CaPS object
in	ldim	dimension in angular momentum ℓ

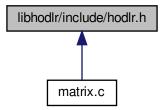
Return values

1	if successful
0	if $Idim < 1$

5.11 libhodlr/include/hodlr.h File Reference

C wrapper for HODLR library.

This graph shows which files directly or indirectly include this file:



Functions

• EXTERNC double hodlr_logdet_diagonal (int dim, double(*callback)(int, int, void *), void *args, double *diagonal, unsigned int nLeaf, double tolerance, int is_symmetric)

Calculate $\log \det(1-M)$ using HODLR approach.

• EXTERNC double hodlr_logdet (int dim, double(*callback)(int, int, void *), void *args, unsigned int nLeaf, double tolerance, int is_symmetric)

Calculate log(det(Id-M)) using HODLR approach.

5.11.1 Detailed Description

C wrapper for HODLR library.

Date

January, 2019

5.11.2 Function Documentation

5.11.2.1 hodlr_logdet()

Calculate log(det(Id-M)) using HODLR approach.

See hodlr_logdet_diagonal for more information.

Parameters

dim	dimension of matrix M
callback	function that returns matrix elements of M
args	pointer that is passed as third argument to callback
nLeaf	nLeaf is the dimension of the smallest block at the leaf level
tolerance	requested accuracy of result
sym_spd	specifiy whether matrix is generic (0), symmetric (1) or spd (2)

Return values

```
| logdet | \log \det(1-M)
```

5.11.2.2 hodlr_logdet_diagonal()

```
EXTERNC double hodlr_logdet_diagonal (
    int dim,
    double(*)(int, int, void *) callback,
    void * args,
    double * diagonal,
    unsigned int nLeaf,
    double tolerance,
    int is_symmetric )
```

Calculate $\log \det (1-M)$ using HODLR approach.

Compute log det(1-A) for a matrix A of dimension given by dim. The diagonal elements of A are given by diagonal which is an array of dim elements. Arbitrary matrix elements A_ij are given by the callback(i,j,args). The requested numerical precision is given by tolerance. returns the matrix elements of A.

nLeaf is the size (number of rows of the matrix) of the smallest block at the leaf level. The number of levels in the tree is given by n_levels=log_2(dim/nLeaf).

Values for sym_psd:

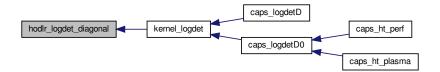
- 0: generic matrix
- 1: matrix is symmetric
- 2: matrix is symmetric and positive definite

Parameters

dim	dimension of matrix M
callback	function that returns matrix elements of M
args	pointer that is passed as third argument to callback
diagonal	array with the diagonal elements of M
nLeaf	nLeaf is the dimension of the smallest block at the leaf level
tolerance	requested accuracy of result
sym_spd	specifiy whether matrix is generic (0), symmetric (1) or spd (2)

Return values

Here is the caller graph for this function:

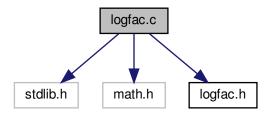


5.12 logfac.c File Reference

computation of logarithm and factorial for integer arguments; created by logfac.py

```
#include <stdlib.h>
#include <math.h>
#include "logfac.h"
```

Include dependency graph for logfac.c:



Functions

```
• double logi (unsigned int n) 
 Calculate \log(n) for integer n.
```

```
• double lfac (unsigned int n)  \textit{Calculate} \log(n!) = \log(\Gamma(n+1)).
```

• double Ifac2 (unsigned int n)

Calculate $\log(n!!)$.

Variables

- static double lookup_logi []
- static double lookup_lfac []
- const size_t __lookup_logi_elems = sizeof(lookup_logi)/sizeof(lookup_logi[0])
- const size_t __lookup_lfac_elems = sizeof(lookup_lfac)/sizeof(lookup_lfac[0])

5.12.1 Detailed Description

computation of logarithm and factorial for integer arguments; created by logfac.py

Author

Michael Hartmann caps@speicherleck.de

Date

January, 2019

5.12.2 Function Documentation

```
Calculate \log(n!) = \log(\Gamma(n+1)).
```

unsigned int n)

This function computes the logarithm of the factorial n!. This function uses a lookup table for $n \leq 1024$

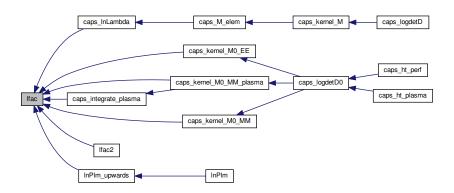
Parameters

in	n	integer
----	---	---------

Return values



Here is the caller graph for this function:



5.12.2.2 Ifac2()

```
double lfac2 ( \label{eq:constraint} \text{unsigned int } n \text{ )}
```

Calculate $\log(n!!)$.

This function computes the logarithm of the double factorial n!!.

Parameters

in	n	argument
		9



Here is the call graph for this function:



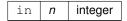
5.12.2.3 logi()

```
double logi ( \quad \quad \text{unsigned int } n \ )
```

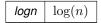
Calculate $\log(n)$ for integer n.

This function uses a lookup table to avoid calling log() for $n \leq 65536$

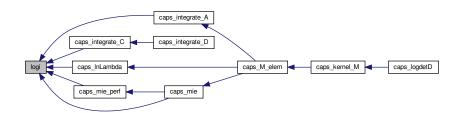
Parameters



Return values



Here is the caller graph for this function:



5.12.3 Variable Documentation

5.12.3.1 lookup_lfac

```
double lookup_lfac[] [static]
```

lookup table for n!, see Ifac

5.12.3.2 lookup_logi

```
double lookup_logi[] [static]
```

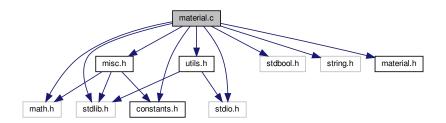
lookup table for $\log(n)$, see \log i

5.13 material.c File Reference

support for arbitrary dielectric functions

```
#include <math.h>
#include <stdbool.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "constants.h"
#include "material.h"
#include "utils.h"
#include "misc.h"
```

Include dependency graph for material.c:



Functions

- static bool _parse (const char *line, const char *key, const char separator, double *value) Helper function to parse strings.
- material_t * material_init (const char *filename, double calL)

Initialize material.

• void material_get_extrapolation (material_t *material, double *omegap_low, double *gamma_low, double *omegap_high, double *gamma_high)

Get extrapolation parameters.

void material_free (material_t *material)

Free material object.

• void material_info (material_t *material, FILE *stream, const char *prefix)

Print information about object to stream.

double material_epsilonm1 (double xi, void *args)

Dielectric function for material.

5.13.1 Detailed Description

support for arbitrary dielectric functions

Author

```
Michael Hartmann caps@speicherleck.de
```

Date

January, 2019

5.13.2 Function Documentation

5.13.2.1 _parse()

Helper function to parse strings.

Parse a string in the form of "key separator value" where key and value represent floating numbers. If key or separator is not found, false is returned. If the string is matched successfully, value is set.

Parameters

in	line	string to parse
in	key	key
in	separator	separator
out	value	numerical value of the string "value"

Return values

true	parsing successful
false	parsing not successful

5.13.2.2 material_epsilonm1()

```
double material_epsilonm1 (  double \ xi, \\ void * args )
```

Dielectric function for material.

Return the dielectric function $\epsilon(i\xi)-1$ for the material. For frequencies greater (smaller) than the maximum (minimum) tabulated frequency, an extrapolation using a Drude model is used. For the tabulated values linear interpolation is used.

Parameters

in	xi	frequency in rad/s
in	args	material (must be of type material_t *)

5.13.2.3 material_free()

Free material object.

Parameters

material	material object
----------	-----------------

5.13.2.4 material_get_extrapolation()

```
void material_get_extrapolation (
    material_t * material,
    double * omegap_low,
    double * gamma_low,
    double * omegap_high,
    double * gamma_high )
```

Get extrapolation parameters.

For frequencies where there is no tabulated data available, the value of the dielectric function will be extrapolated assuming Drude behaviour:

$$\epsilon(i\xi) = 1 + \frac{\omega_P^2}{\xi(\xi + \gamma)}$$

The parameters for the plasma frequency ω_P and the relaxation frequency γ for $\xi > \xi_{\rm max}$ and $\xi < \xi_{\rm min}$ will be stored into omegap_high, gamma_high, and omegap_low, gamma_low. If a pointer is NULL, the memory is not referenced.

in	material	material object
out	omegap_low	plasma frequency for high-frequency extrapolation (in rad/s)
out	gamma_low	relaxation frequency for high-frequency extrapolation (in rad/s)
out	omegap_high	plasma frequency for low-frequency extrapolation (in rad/s)
Generated	byg@ppaymen_high	relaxation frequency for low-frequency extrapolation (in rad/s)

5.13.2.5 material_info()

Print information about object to stream.

Print information (filename, number of points, ξ_{\min} , ξ_{\max} , ...) to stream. If prefix is not NULL, each line will start with the string given in prefix.

Parameters

in	material	material object
in	stream	output stream (e.g. stdout)
in	prefix	prefix for each line or NULL

5.13.2.6 material_init()

Initialize material.

The material properties are read from the file given by filename.

This function temporarily overwrites the value of LC_NUMERIC in the environment. LC_NUMERIC is restored before returning from the function.

Be aware that this function does not check every corner case, so it is dangerous to read untrusted files.

Parameters

in	filename	path to material specification
in	calL	L+R, separation between plane and center of sphere

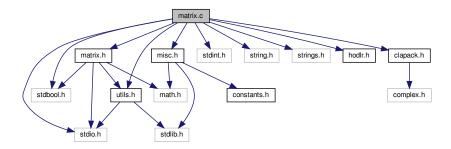
material	if successful
NULL	if file cannot be read or is in wrong format

5.14 matrix.c File Reference

Matrix functions.

```
#include <stdbool.h>
#include <stdint.h>
#include <stdio.h>
#include <string.h>
#include <strings.h>
#include <hodlr.h>
#include "matrix.h"
#include "misc.h"
#include "utils.h"
#include "clapack.h"
```

Include dependency graph for matrix.c:



Functions

- double kernel_logdet (int dim, double(*kernel)(int, int, void *), void *args, int sym_spd, detalg_t detalg)
 Compute log det(1 A).
- matrix_t * matrix_alloc (const size_t dim)

Create new matrix object.

void matrix_free (matrix_t *A)

Free matrix.

• int matrix_save_to_stream (matrix_t *A, FILE *stream)

Save matrix to stream.

int matrix_save_to_file (matrix_t *A, const char *filename)

Save matrix to file.

matrix_t * matrix_load_from_stream (FILE *stream)

Load matrix from stream.

matrix_t * matrix_load_from_file (const char *filename)

Load matrix from file.

void matrix setall (matrix t *A, double z)

Set all matrix elements to value z.

double matrix_trace (matrix_t *A)

Calculate trace of matrix.

double matrix trace2 (matrix t *A)

Calculate trace of A^2 .

double matrix_norm_frobenius (matrix_t *A)

```
Calculate Frobenius norm of A.

    double matrix logdet triangular (matrix t *A)

      Calculate \log \det A for triangular matrix A.
• double matrix logdet dense (matrix t *A, double z, detalg t detalg)
      Calculate \log \det(1+zA) for matrix A.

    double matrix logdet lu (matrix t *A)

      Calculate \log \det A using LU decomposition.

    double matrix_logdet_cholesky (matrix_t *A, char uplo)

      Calculate \log \det A using Cholesky decomposition.

    double matrix logdet qr (matrix t *A)

      Calculate \log \det A using QR decomposition.

    matrix_t * matrix_mult (matrix_t *A, matrix_t *B, double alpha)

      Compute the matrix multiplication alpha*A*B.
```

matrix_t * matrix_copy (matrix_t *A)

Copy matrix.

5.14.1 Detailed Description

Matrix functions.

Author

Michael Hartmann caps@speicherleck.de

Date

January, 2019

5.14.2 Function Documentation

```
5.14.2.1 kernel_logdet()
```

```
double kernel_logdet (
             int dim,
             double(*)(int, int, void *) kernel,
             void * args,
             int sym_spd,
             detalg_t detalg )
```

Compute $\log \det(1-A)$.

This function computes $\log \det(1-A)$ using either the HODLR approach or LU decomposition. The matrix A is given as a callback function. This callback accepts two integers, the row and the column of the matrix entry (starting from 0), and a pointer to args. The callback returns the corresponding matrix element.

If the matrix elements of A are small, i.e., if the modulus of the trace is smaller than 1e-8, the trace will be used as an approximation to prevent a loss of significance. If the modulus of the trace is larger than the modulus of the value computed using HODLR, the trace approximation is returned.

If the determinant is not computed using the HODLR approach, all matrix elements have to be computed. In this case the matrix A is written to the filesystem if the environment variable CAPS DUMP is set. If the variable is set, the matrix will be stored in the path given by CAPS DUMP as a two-dimensional numpy array (npy). This option might be useful for debugging. Also note that if detailg is CHOLESKY, only the upper half of the matrix will be initialized.

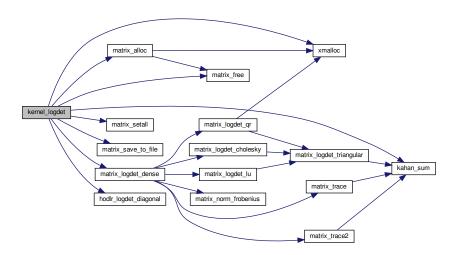
Parameters

in	dim	dimension of matrix	
in	kernel	callback function that returns matrix elements of \boldsymbol{A}	
in	args	pointer given to callback function kernel	
in	sym_spd	matrix is generic (0), symmetric (1), symmetric positive definite (2)	
in	detalg	algorithm (DETALG_HODLR, DETALG_LU, DETALG_QR, DETALG_CHOLESKY)	

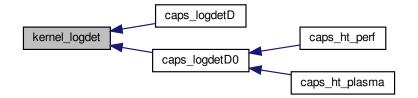
Return values

logdet	$\log \det(1-A)$
--------	------------------

Here is the call graph for this function:



Here is the caller graph for this function:



5.14.2.2 matrix_alloc()

Create new matrix object.

Create a new square matrix with dimension dim x dim. The matrix will not be initialized.

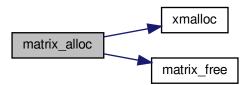
Parameters

in	dim	dimension of square matrix
----	-----	----------------------------

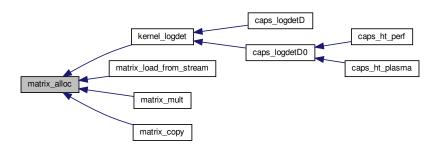
Return values



Here is the call graph for this function:



Here is the caller graph for this function:

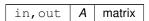


5.14.2.3 matrix_copy()

Copy matrix.

The function returns a copy of the input matrix A.

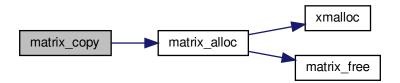
Parameters



Return values



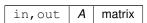
Here is the call graph for this function:



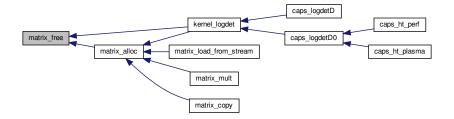
5.14.2.4 matrix_free()

Free matrix.

This function frees the memory allocated for the matrix A.



Here is the caller graph for this function:



5.14.2.5 matrix_load_from_file()

Load matrix from file.

Load matrix matrix from file filename. See matrix_load_from_stream for more information.

Parameters

in	filename	filename of output file
----	----------	-------------------------

Return values

Α	matrix if successful
NULL	if an error occured

5.14.2.6 matrix_load_from_stream()

Load matrix from stream.

This function loads a matrix from a given stream. The input must be in .npy format. The input matrix must be a square matrix.

The function will rudimentary parse the description string and abort if an error occures. Do not use this function on untrusted data.

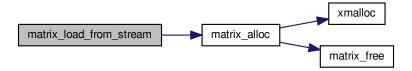
Parameters

in <i>stream</i> stream

Return values

Α	matrix if successful
NULL	if an error occured

Here is the call graph for this function:



5.14.2.7 matrix_logdet_cholesky()

Calculate $\log \det A$ using Cholesky decomposition.

Calculate Cholesky decomposition of A and use $\operatorname{matrix_logdet_triangular}$ to calculate $\log \det A$.

Only the lower part of the matrix (uplo=L) or the upper part of the matrix (uplo=U) are used.

Parameters

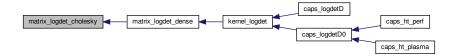
in,out	Α	matrix
in	uplo	L or U

logdet	$\log \det A$

Here is the call graph for this function:



Here is the caller graph for this function:



5.14.2.8 matrix_logdet_dense()

Calculate $\log \det(1+zA)$ for matrix A.

Compute $\log \det(1+zA)$ using LAPACK. The algorithm is chosen by detailg and may be DETALG_QR, DETAL \leftarrow G_LU or DETALG_CHOLESKY.

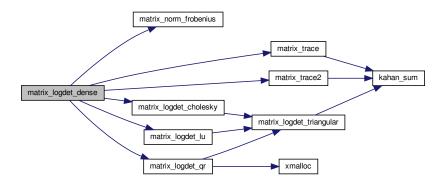
If the Frobenius norm of zA is smaller than 1, the function tries to approximate $\log \det A$ using a Mercator series (if possible) to reduce the complexity for an $N \times N$ matrix A from $\mathcal{O}(N^3)$ to $\mathcal{O}(N^2)$.

Parameters

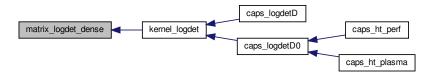
in,out	Α	matrix; will be overwritten.
in	Z	factor z
in	detalg	algorithm to use (cholesky, lu or qr)

logdet	$\log \det(1 + zA)$
--------	---------------------

Here is the call graph for this function:



Here is the caller graph for this function:

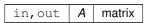


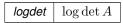
5.14.2.9 matrix_logdet_lu()

Calculate $\log \det A$ using LU decomposition.

Calculate LU decomposition of A and use matrix_logdet_triangular to calculate $\log \det A$.

Parameters





Here is the call graph for this function:



Here is the caller graph for this function:

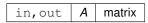


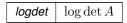
5.14.2.10 matrix_logdet_qr()

Calculate $\log \det A$ using QR decomposition.

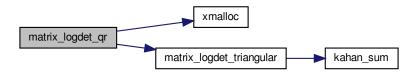
Calculate QR decomposition of A and use $matrix_logdet_triangular$ to calculate $\log \det A$.

Parameters

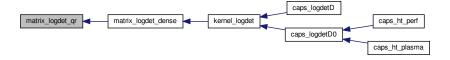




Here is the call graph for this function:



Here is the caller graph for this function:



5.14.2.11 matrix_logdet_triangular()

Calculate $\log \det A$ for triangular matrix A.

This function calculates the logarithm of the determinant of the matrix A assuming A is upper or lower triangular:

$$\log \det A = \log \prod_{j} A_{jj} = \sum_{j} \log A_{jj}$$

Parameters

in A triangular matrix

logdet	$\log \det A$

Here is the call graph for this function:



Here is the caller graph for this function:

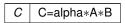


5.14.2.12 matrix_mult()

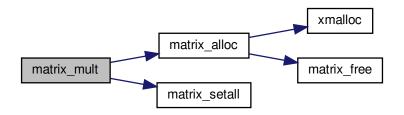
Compute the matrix multiplication alpha*A*B.

Parameters

in	Α	matrix
in	В	matrix
in	alpha	scalar



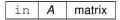
Here is the call graph for this function:



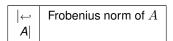
5.14.2.13 matrix_norm_frobenius()

Calculate Frobenius norm of A.

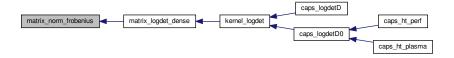
Parameters



Return values



Here is the caller graph for this function:



5.14.2.14 matrix_save_to_file()

Save matrix to file.

Save matrix A to file filename. See ${\sf matrix_save_to_stream}$ for more information.

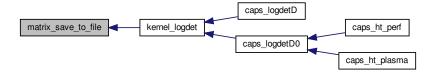
Parameters

in	Α	matrix
in	filename	filename of output file

Return values



Here is the caller graph for this function:



5.14.2.15 matrix_save_to_stream()

Save matrix to stream.

This function saves the matrix A to the stream given by stream. The output is in the numpy .npy format.

Parameters

in	Α	matrix
in	stream	stream



5.14.2.16 matrix_setall()

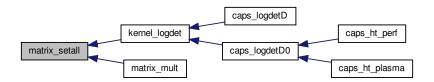
```
void matrix_setall ( \label{eq:matrix_t} \mbox{ matrix\_t * A,} \\ \mbox{ double } z \mbox{ )}
```

Set all matrix elements to value z.

Parameters

in,out	Α	matrix
in	Z	value

Here is the caller graph for this function:

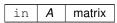


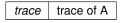
5.14.2.17 matrix_trace()

Calculate trace of matrix.

This function uses Kahan sumation (see kahan_sum) to reduce rounding errors.

Parameters

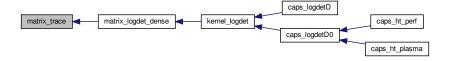




Here is the call graph for this function:



Here is the caller graph for this function:



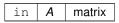
5.14.2.18 matrix_trace2()

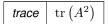
Calculate trace of A^2 .

This function uses Kahan sumation (see kahan_sum) to reduce rounding errors.

The function needs $\mathcal{O}(N^2)$ operation for an $N \times N$ matrix.

Parameters



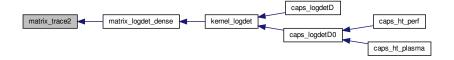


5.15 misc.c File Reference 173

Here is the call graph for this function:



Here is the caller graph for this function:

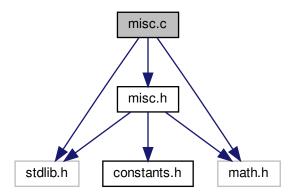


5.15 misc.c File Reference

various mathematical functions

```
#include <stdlib.h>
#include <math.h>
#include "misc.h"
```

Include dependency graph for misc.c:



Functions

```
• double kahan_sum (double input[], size_t N)
```

Compute sum of array elements.

```
• double sqrtpm1 (double x)
```

```
Compute \sqrt{1+x}-1.
```

• double logadd (const double log_a, const double log_b)

Add two numbers given by their logarithms.

• double logadd_ms (log_t list[], const int N, sign_t *sign)

Add N numbers given by their logarithms.

5.15.1 Detailed Description

various mathematical functions

Author

Michael Hartmann caps@speicherleck.de

Date

July, 2017

5.15.2 Function Documentation

5.15.2.1 kahan_sum()

Compute sum of array elements.

This function calculates the sum of the elements of the array input. This function uses the Kahan summation algorithm to reduce numerical error.

The algorithm is taken from Wikipedia, see https://en.wikipedia.org/wiki/Kahan_summation←_algorithm.

Parameters

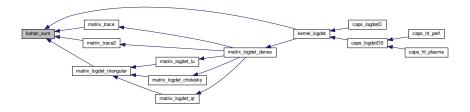
in	input	array
in	Ν	length of array

5.15 misc.c File Reference 175

Returns

sum sum of array elements

Here is the caller graph for this function:



5.15.2.2 logadd()

Add two numbers given by their logarithms.

Both numbers are assumed to be nonnegative.

Parameters

in	log⊷	number
	_a	
in	log⊷	number
	b	

Returns

$$\text{log_sum} \log \left[\exp(\log_a) + \exp\left(\log_b\right) \right]$$

Here is the caller graph for this function:



5.15.2.3 logadd_ms()

Add N numbers given by their logarithms.

The logarithm and the sign of the N numbers are given by list. The numbers of elements of list must be N, the sign of the result will be stored in sign.

Parameters

in	list	list of numbers given by logarithm and sign
in	Ν	number of elements of list
out	sign	sign of the result

Returns

logsum log(sum_i list_i)

5.15.2.4 sqrtpm1()

```
double sqrtpm1 ( double x )
```

Compute $\sqrt{1+x}-1$.

If x is small, $\sqrt{1+x}\approx 1$ and a loss of significance occurs when calculating $\sqrt{1+x}-1$.

For this reason we compute

$$\sqrt{1+x} - 1 = \frac{x}{\sqrt{1+x} + 1}$$

to avoid a loss of significance if \boldsymbol{x} is small.

Parameters



Return values

sqrt(1+x)-1

Here is the caller graph for this function:

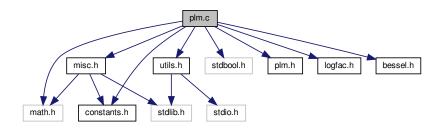


5.16 plm.c File Reference

computation of Legendre and associated Legendre polynomials

```
#include <math.h>
#include <stdbool.h>
#include "constants.h"
#include "plm.h"
#include "logfac.h"
#include "misc.h"
#include "bessel.h"
#include "utils.h"
```

Include dependency graph for plm.c:



Functions

• double InPIm (int I, int m, double x)

Associated Legendre polynomials for argument x > 1.

• double InPIm_upwards (int I, int m, double x)

Associated Legendre polynomials using upwards recurrence relation.

• static double _PI1 (int I, double x, double sinhxi)

Compute Legendre polynomial $\log P_l(x)$ for large x.

- static double _fn (int n, double hn[13])
- static double _PI2 (int I, double x)

Compute Legendre polynomial $\log P_l(x)$ for small x.

• static double _PI3 (int I, double x)

Compute Legendre polynomial $\log P_l(x)$ using recurrence relation.

• double InPI (int I, double x)

Compute Legendre polynomial $\log P_l(x)$.

• double PIm_continued_fraction (const long I, const long m, const double x) $\textit{Calculate fraction } P_l^{m-1}(x)/P_l^m(x).$

• double InPlm downwards (int I, int m, double x)

Compute associated Legendre polynomials using downwards recurrence relation.

• double dlnPlm (int I, int m, double x, double *d2lnPlm)

Compute 1st and 2nd logarithmic derivative of associated Legendre polynomial.

5.16.1 Detailed Description

computation of Legendre and associated Legendre polynomials

Author

Michael Hartmann caps@speicherleck.de

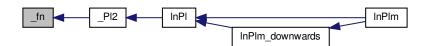
Date

January, 2019

5.16.2 Function Documentation

```
5.16.2.1 _fn()
```

see equations (3.27)-(3.31) Here is the caller graph for this function:



5.16.2.2 _PI1()

Compute Legendre polynomial $\log P_l(x)$ for large x.

Evaluation of $\log P_l(x)$ for $x \geq 1$ using an asymptotic expansion provided that

$$(l+1)\sqrt{(x+1)(x-1)} \ge 25.$$

 $\mathcal{O}(1)$ computation of Legendre polynomials and Gauss-Legendre nodes and weights for parallel computing, section 3.2.

See InPl.

Parameters

in	1	degree
in	X	argument
in	sinhxi	$\sinh \xi = \sqrt{(x+1)(x-1)}$

Return values

$$\left| \begin{array}{c|c} \textit{log} \leftarrow & \log P_l(x) \\ \textit{PI} & \end{array} \right|$$

Here is the caller graph for this function:



5.16.2.3 _PI2()

```
static double _P12 (  \mbox{int $l$,} \\ \mbox{double $x$ ) [static]}
```

Compute Legendre polynomial $\log P_l(x)$ for small x.

Evaluation of $\log P_l(x)$ for ≥ 1 using an asymptotic expansion provided that

$$(l+1)\sqrt{(x+1)(x-1)} < 25.$$

 $\mathcal{O}(1)$ computation of Legendre polynomials and Gauss-Legendre nodes and weights for parallel computing, section 3.3.

See InPl.

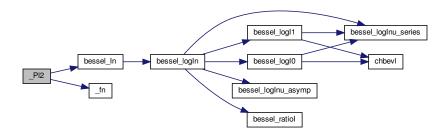
Parameters

in	1	
in	X	

Return values

log⊷	$\log P_l(x)$
PI	

Here is the call graph for this function:



Here is the caller graph for this function:



5.16.2.4 _PI3()

```
static double _P13 (  \mbox{int } 1, \\ \mbox{double } x \;) \;\; [static]
```

Compute Legendre polynomial $\log P_l(x)$ using recurrence relation.

Evaluation of $\log P_l(x)$ for $x \ge 1$ using the recurrence relation

$$(n+1)P_{n+1}(x) = (2n+1)xP_n(x) - nP_{n-1}(x).$$

See InPl.

Parameters

in	1	order
in	Χ	argument

Return values

log⊷	$\log P_l(x)$
PI	

Here is the caller graph for this function:



5.16.2.5 dlnPlm()

```
double dlnPlm (
    int 1,
    int m,
    double x,
    double * d2lnPlm )
```

Compute 1st and 2nd logarithmic derivative of associated Legendre polynomial.

Compute $\frac{\mathrm{d}}{\mathrm{d}x}\log P_l^m(x)$ and $\frac{\mathrm{d}^2}{\mathrm{d}x^2}\log P_l^m(x).$

If d2lnPlm is NULL, the 2nd logarithmic derivative will not be computed.

Parameters

in	1	degree
in	m	order
in	X	argument
out	d2lnPlm	2nd logarithmic derivative of $P_l^m(x)$

Generated by Doxygen

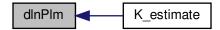
Return values

```
dlnPlm | first logarithmic derivative of P_l^m(x)
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.16.2.6 InPI()

```
double lnPl ( \inf \ 1, \operatorname{double} \ x \ )
```

Compute Legendre polynomial $\log P_l(x)$.

Evaluation of $\log P_l(x)$ for $x \ge 1$.

For l<100 a recurrence relation is used (see _PI3), otherwise asymptotic expansions are used (see _PI1 and _PI2).

The function returns $\log P_l(x)$.

Reference:

• Bogaert, Michiels, Fostier, O(1) Computation of Legendre Polynomials and Gauss-Legendre Nodes and Weights for Parallel Computing, SIAM J. Sci. Comput. 3, 34 (2012)

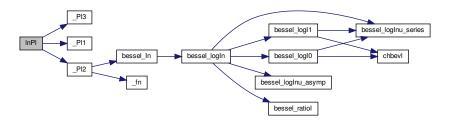
Parameters

in	I	degree
in	X	argument

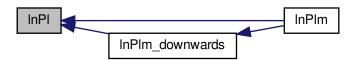
Return values

log⊷	$\log P_l(x)$
PI	

Here is the call graph for this function:



Here is the caller graph for this function:



5.16.2.7 InPIm()

Associated Legendre polynomials for argument x>1.

This function calculates associated Legendre functions for $m\geq 0$ and x>1.

The associated Legendre polynomials for x > 1 are defined as follows (see references)

$$P_l^m(x) = (x^2 - 1)^{m/2} \frac{\mathrm{d}}{\mathrm{d}x^m} P_l(x).$$

Note that in contrast to the common choice in physics, we omit the Condon-Shortly phase $(-1)^m$, and interchange the factors x^2 and 1 in the first bracket after the equal sign. With this definition the associated Legendre polynomials are real and positive functions.

For $l-m \leq 200$ we use an upwards recurrence relation in m, see InPIm_upwards, otherwise we use a downwards recurrence relation in m, see InPIm_downwards .

References:

- DLMF, §14.7.11, http://dlmf.nist.gov/14.7#E11
- · Zhang, Jin, Computation of Special Functions, 1996

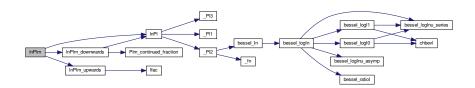
Parameters

in	1	degree
in	m	order
in	X	argument

Return values

logPlm	$\log P_l^m(x)$
--------	-----------------

Here is the call graph for this function:



5.16.2.8 InPIm_downwards()

```
double lnPlm\_downwards ( int \ l, int \ m, double \ x \ )
```

Compute associated Legendre polynomials using downwards recurrence relation.

First, the fraction $P_l^m(x)/P_l^{m-1}(x)$ is computed using PIm_continued_fraction. Then the downwards recurrence relation http://dlmf.nist.gov/14.10.E6 is used from $P_l^m(x)$ to $P_l^0(x)$. Together with $P_l(x)$ (see InPI) one can compute $P_l^m(x)$.

This routine is efficient if $l \gg m$.

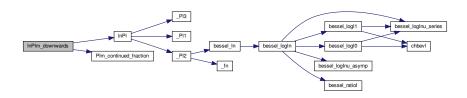
Parameters

in	1	degree
in	m	order
in	X	argument

Return values

$ logPlm log P_l^m(x)$

Here is the call graph for this function:



Here is the caller graph for this function:



5.16.2.9 InPlm_upwards()

Associated Legendre polynomials using upwards recurrence relation.

The values of $P_l^m(x)$ is computed using the recurrence relation http://dlmf.nist.gov/14.10.E3 in upwards direction starting from

$$P_m^m(x) = \frac{(2m)!}{2^m m!} (x^2 - 1)^{m/2}$$

(http://dlmf.nist.gov/14.7.E15).

Parameters

in	1	degree
in	m	order
in	X	argument

Return values

logPlm	$\log P_l^m(x)$
- 3	· 0 (.)

Here is the call graph for this function:



Here is the caller graph for this function:



5.16.2.10 Plm_continued_fraction()

Calculate fraction $P_l^{m-1}(x)/P_l^m(x)$.

The fraction is computed using a continued fraction, see http://dlmf.nist.gov/14.14.E1.

To evaluate the continued fraction, we use http://dlmf.nist.gov/1.12#E5 and $http://dlmf. \leftarrow nist.gov/1.12\#E6$.

See also Numerical Recipes in C, chapter 5.2, Evaluation of Continued Fractions.

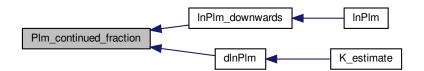
Parameters

in	1	degree
in	m	order
in	X	argument

Return values

ratio	$P_l^{m-1}(x)/P_l^m(x)$
-------	-------------------------

Here is the caller graph for this function:

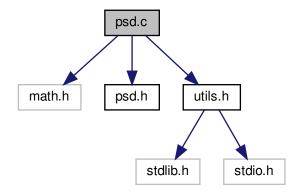


5.17 psd.c File Reference

expansion coefficients and poles for Pade spectrum decomposition

```
#include <math.h>
#include "psd.h"
#include "utils.h"
```

Include dependency graph for psd.c:



Functions

• int dstemr_ (char *jobz, char *range, int *n, double *d__, double *e, double *vl, double *vu, int *il, int *iu, int *m, double *w, double *z__, int *ldz, int *nzc, int *isuppz, int *tryrac, double *work, int *lwork, int *iwork, int *liwork, int *info)

• static double <u>eta</u> (int N, double z)

Compute expansion coefficients.

• int psd (int N, double xi[N], double eta[N])

Compute poles ξ_i and expansion coefficients η_i for PSD.

5.17.1 Detailed Description

expansion coefficients and poles for Pade spectrum decomposition

Author

Michael Hartmann caps@speicherleck.de

Date

December, 2018

5.17.2 Function Documentation

```
5.17.2.1 _eta()
```

```
static double _eta (  \qquad \qquad \text{int $N$,} \\  \qquad \qquad \text{double $z$ ) [static]}
```

Compute expansion coefficients.

Compute expansion coefficient η_i according to the paragraph around equations (12) and (13). See psd.

Parameters

in	Ν	order
in	Z	$z = -\xi_j^2$



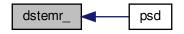
Here is the caller graph for this function:



5.17.2.2 dstemr_()

```
int dstemr_ (
             char * jobz,
             char * range,
             int * n,
             double * d__,
             double * e,
             double *vl,
             double * vu,
             int * il,
             int * iu,
             int * m,
             double * w,
             double * z_{\_},
             int * ldz,
             int * nzc,
             int * isuppz,
             int * tryrac,
             double * work,
             int * lwork,
             int * iwork,
             int * liwork,
             int * info)
```

prototype for LAPACK routine Here is the caller graph for this function:



5.17.2.3 psd()

Compute poles ξ_i and expansion coefficients η_i for PSD.

This function computes the poles ξ_j (at imaginary frequency) and the expansion coefficients η_j for the Pade spectrum decomposition of order N, see reference [1]. The poles are stored in the array xi, the coefficients are stored in the array eta.

References:

• Hu, Xu, Yan, J. Chem. Phys. 133, 101106 (2010)

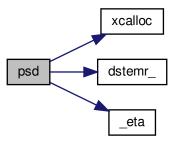
Parameters

in	Ν	order
out	хi	poles
out	eta	expansion coefficients

Return values

success	0 if successful
---------	-----------------

Here is the call graph for this function:



5.18 utils.c File Reference

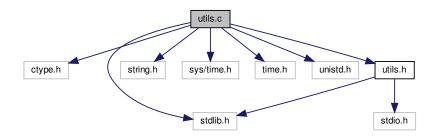
wrappers for malloc, calloc realloc, and a few more useful functions

```
#include <ctype.h>
#include <stdlib.h>
```

5.18 utils.c File Reference 191

```
#include <string.h>
#include <sys/time.h>
#include <time.h>
#include <unistd.h>
#include "utils.h"
```

Include dependency graph for utils.c:



Functions

void * xmalloc (size_t size)

Wrapper for malloc.

void * xcalloc (size_t nmemb, size_t size)

Wrapper for calloc.

void * xrealloc (void *p, size_t size)

Wrapper for realloc.

• double now (void)

Seconds since 01/01/1970.

void time_as_string (char *s, size_t len)

Write time into string.

void disable_buffering (void)

Disable buffering to stderr and stdout.

• void strrep (char *s, const char a, const char b)

Replace character by different character in string.

void strim (char *str)

Remove whitespace at beginng and end of string.

5.18.1 Detailed Description

wrappers for malloc, calloc realloc, and a few more useful functions

Author

Michael Hartmann caps@speicherleck.de

Date

January, 2018

5.18.2 Function Documentation

5.18.2.1 disable_buffering()

Disable buffering to stderr and stdout.

```
5.18.2.2 now()
```

```
double now ( void )
```

Seconds since 01/01/1970.

This function returns the seconds since 1st Jan 1970 in μs precision.

Return values

```
time seconds since 1st Jan 1970
```

5.18.2.3 strim()

```
void strim ( {\tt char} \ * \ str \ )
```

Remove whitespace at beginng and end of string.

If str is NULL the function doesn't do anything. Otherwise, trailing whitespace and whitespace at the beginning of the string are removed.

Parameters

```
str string
```

5.18.2.4 strrep()

```
void strrep ( {\rm char} \, * \, s,
```

5.18 utils.c File Reference

```
const char a, const char b)
```

Replace character by different character in string.

Replace occurence of a by b in the string s.

Parameters

in,out	s	string, terminated by \0
in	а	character to replace
in	b	substitute

5.18.2.5 time_as_string()

Write time into string.

Write current time in a human readable format into string s. The output is similar to "Aug 30 2018 14:37:35".

Parameters

s	string
len	maximum length of array s

5.18.2.6 xcalloc()

Wrapper for calloc.

This function is a wrapper for calloc. If calloc fails TERMINATE is called.

Parameters

nmemb	number of elements
size	size of each element

ptr	pointer to memory
-----	-------------------

Here is the caller graph for this function:



5.18.2.7 xmalloc()

Wrapper for malloc.

This function is a wrapper for malloc. If malloc fails TERMINATE is called.

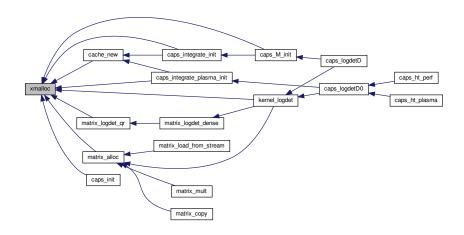
Parameters

size	size of bytes to allocate
------	---------------------------

Return values

```
ptr pointer to memory
```

Here is the caller graph for this function:



5.18 utils.c File Reference 195

5.18.2.8 xrealloc()

```
void* xrealloc ( \label{eq:condition} \mbox{void} * p, \\ \mbox{size\_t } size \mbox{)}
```

Wrapper for realloc.

This function is a wrapper for realloc. If realloc fails TERMINATE is called.

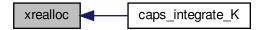
Parameters

р	ptr to old memory
size	size

Return values

newptr	pointer to new memory
--------	-----------------------

Here is the caller graph for this function:



Index

_Pl1	bessel_K0
plm.c, 178	bessel.c, 26
_Pl2	bessel_K1
plm.c, 179	bessel.c, 26
_Pl3	bessel_Kn
plm.c, 180	bessel.c, 27
_eta	bessel_logI0
psd.c, 188	bessel.c, 28
_fn	bessel_logI1
plm.c, 178	bessel.c, 29
_parse	bessel_logIn
material.c, 154	bessel.c, 30
	bessel_logIn_half
argparse, 7	bessel.c, 31
argparse_option, 8	bessel_logInKn_half
	bessel.c, 32
bessel.c, 21	bessel loglnu asymp
bessel_I0, 23	bessel.c, 33
bessel_I1, 24	bessel_logInu_series
bessel_In, 25	bessel.c, 34
bessel_K0, 26	bessel_logK0
bessel_K1, 26	bessel.c, 35
bessel_Kn, 27	bessel_logK1
bessel_logI0, 28	-
bessel_logl1, 29	bessel.c, 36
bessel_logIn, 30	bessel_logKn
bessel_logIn_half, 31	bessel.c, 37
bessel_logInKn_half, 32	bessel_logKn_half
bessel_logInu_asymp, 33	bessel.c, 38
bessel_logInu_series, 34	bessel_logKn_recursive
bessel_logK0, 35	bessel.c, 39
bessel_logK1, 36	bessel_logKnu_asymp
bessel_logKn, 37	bessel.c, 40
bessel_logKn_half, 38	bessel_ratiol
bessel_logKn_recursive, 39	bessel.c, 41
bessel_logKnu_asymp, 40	buf, 9
bessel_ratiol, 41	buffer, 9
chbevl, 41	capacity, 9
I0_coeffs, 42	size, 9
I1_coeffs, 43	buf.h
K0_coeffsA, 44	buf_capacity, 60
K0_coeffsB, 44	buf_clear, 60
K1_coeffsA, 44	buf_free, 61
K1_coeffsB, 45	buf_grow, 61
bessel_I0	buf_pop, 61
bessel.c, 23	buf_push, 61
bessel_I1	buf_size, 61
bessel.c, 24	buf_trunc, 62
bessel_In	buf_capacity
bessel.c, 25	buf.h, 60
•	•

buf_clear	L, 11
buf.h, 60	LbyR, 11
buf_free	ldim, 11
buf.h, 61	R, 11
buf_grow	y, 12
buf.h, 61	caps_M_elem
buf_pop	libcaps.c, 137
buf.h, 61	libcaps.h, 87
buf_push	caps_M_free
buf.h, 61	libcaps.c, 139
buf_size	libcaps.h, 88
buf.h, 61	caps_M_init
buf_trunc	libcaps.c, 139
buf.h, 62	libcaps.h, 89
buffer	caps_M_t, 12
buf, 9	caps_build
CARS CACHE FLEMS	libcaps.c, 119
CAPS_CACHE_ELEMS	libcaps.h, 69
libcaps.h, 68	caps_epsilonm1_drude
CAPS_EPSREL	libcaps.c, 119
libcaps.h, 68	libcaps.h, 69
CAPS_FACTOR_LDIM	caps_epsilonm1_perf
libcaps.h, 68	libcaps.c, 120
CAPS_MINIMUM_LDIM	libcaps.h, 70
libcaps.h, 68	caps_epsilonm1_plate
CAPS_c	libcaps.c, 120
constants.h, 63	libcaps.h, 71
CAPS_hbar	caps_epsilonm1_sphere
constants.h, 63	libcaps.c, 121
CAPS_hbar_eV	libcaps.h, 71
constants.h, 63	caps_estimate_lminmax
CAPS_kB	libcaps.c, 122
constants.h, 63	libcaps.h, 72
COMPILER	caps_free
utils.h, 98	libcaps.c, 122
cache.c, 46	libcaps.h, 73
cache_free, 47	caps_fresnel
cache_insert, 47	libcaps.c, 124
cache_lookup, 48	libcaps.h, 73
cache_new, 48	caps_get_detalg
cache_entry_t, 9	libcaps.c, 124
cache_free	libcaps.h, 74
cache.c, 47	caps_get_epsrel
cache_insert	libcaps.c, 125
cache.c, 47	libcaps.h, 74
cache_lookup	
cache.c, 48	caps_get_ldim libcaps.c, 125
cache_new	•
cache.c, 48	libcaps.h, 74
cache_t, 10	caps_ht_drude
calL	libcaps.c, 125
caps, 11	libcaps.h, 75
material_t, 18	caps_ht_perf
capacity	libcaps.c, 126
buf, 9	libcaps.h, 76
caps, 10	caps_ht_plasma
calL, 11	libcaps.c, 127
detalg, 11	libcaps.h, 77
epsrel, 11	caps_info

libcaps.c., 128		
caps_init libcaps.h, 94 caps_integrate A libcaps.c, 144 caps_integrate A libcaps.c, 144 integration.c, 106 caps_integrate.B caps_integrate C libcaps.c, 145 caps_integrate D libcaps.c, 146 integration.c, 106 libcaps.c, 146 caps_integrate D libcaps.c, 146 integration.c, 107 caps_set ldim caps_integrate plantegrate, free integration.c, 108 libcaps.c, 146 caps_integrate init integration.c, 109 caps_integrate init integration.c, 110 caps_integrate_plate, free integration.c, 111 caps_integrate_plasma integration.c, 110 caps_integrate_plasma integration.c, 111 caps_integrate_plasma integration.c, 112 caps_integrate_plasma caps_integrate_plasma_init integration.c, 113 caps_integrate_plasma_init integration.c, 114 caps_integrate_plasma_init integration.c, 114 days_kernel_MO_EE libcaps.c, 133 ibcaps.c, 134 libcaps.c, 133 ibcaps.c, 134 libcaps.c, 133 ibcaps.c, 134 libcaps.c, 136 ilicaps.c, 136 libcaps.c, 135	libcaps.c, 128	caps_set_epsilonm1
libcaps.c, 129 caps_set_epsilonm1 plate libcaps.c, 144 libcaps.h, 94 libcaps.h, 94 libcaps.c, 144 libcaps.h, 94 libcaps.h, 94 libcaps.h, 94 libcaps.h, 95 caps_integrate. B libcaps.h, 95 libcaps.h, 95 caps_integrate. C caps_integrate. C libcaps.h, 95 libcaps.h, 96 libcaps.h, 97 libcaps.h, 98 libcaps.h, 98 libcaps.h, 99 libcaps.h, 90	libcaps.h, 78	libcaps.c, 144
libcaps.h, 78	caps_init	libcaps.h, 94
caps_integrate_A libcaps.h, 94 integration.c, 104 caps_set_epsilonn1_sphere caps_integrate_B libcaps.c, 145 integration.c, 106 libcaps.c, 146 caps_integrate_C caps_set_epsrel integration.c, 106 libcaps.c, 146 caps_integrate_D libcaps.c, 146 integration.c, 107 caps_set_ldim caps_integrate_free libcaps.c, 146 integration.c, 108 libcaps.h, 96 caps_integrate_lfree libcaps.h, 96 caps_integrate_lfree libcaps.h, 96 caps_integrate_lfree libcaps.h, 88 caps_integrate_int caps_integrate_lfree integration.c, 110 caps_integrate_plasma caps_integrate_plasma cAPS_c, 63 caps_integrate_plasma_free caps_integrate_plasma_free integration.c, 113 caps_integrate_lM0_EE libcaps.c, 133 M_LOGP, 63 caps_kernel_M0_EE M_PI, 64 MAX, 64 MIDcaps.h, 80 caps_kernel_M0_IMD caps_integrate_lama libcaps.c, 130 caps_integrate_lama <td< td=""><td>libcaps.c, 129</td><td></td></td<>	libcaps.c, 129	
integration.c, 104 caps integrate. B integration.c, 106 caps_integrate. C integration.c, 106 caps_integrate. C integration.c, 107 caps integrate. D integration.c, 107 caps_integrate. To caps_integrate. D integration.c, 107 caps_integrate free integration.c, 108 caps_integrate. I integration.c, 109 caps_integrate. I integration.c, 109 caps_integrate. I integration.c, 110 caps_integrate. I integration.c, 110 caps_integrate. I integration.c, 111 caps_integrate. I integration.c, 111 caps_integrate. A integration.c, 111 caps_integrate. plasma integration.c, 112 caps_integrate. plasma integration.c, 113 caps_integrate. plasma integration.c, 114 caps_integrate. plasma integration.c, 114 caps_integrate. plasma integration.c, 114 caps_kernel_M0_EE ilbcaps.c, 131 ilbcaps.c, 130 ilbcaps.c, 132 ilbcaps.c, 132 ilbcaps.c, 133 ilbcaps.c, 133 ilbcaps.c, 134 ilbcaps.c, 135 ilbcaps.c, 136 ilbcaps.c, 135 ilbcaps.c, 136 ilbcaps.c, 135 ilbcaps.c, 135 ilbcaps.c, 136 ilbcaps.c, 136 ilbcaps.c, 137 ilbcaps.c, 136 ilbcaps	libcaps.h, 78	libcaps.c, 144
caps_integrate_B libcaps.c, 145 integration.c, 106 libcaps.h, 95 caps_integrate_C caps_set_epsrel integration.c, 106 libcaps.c, 146 caps_integrate_D libcaps.h, 96 integration.c, 107 caps_set_ldim caps_integrate_free libcaps.h, 96 integration.c, 108 caps_taps.h, 96 caps_integrate_l libcaps.h, 96 caps_integrate_l libcaps.h, 96 caps_integrate_l caps_task_t, 14 integration.c, 109 libcaps.h, 96 caps_integrate_laint caps_task_t, 14 integration.c, 110 caps_task_t, 14 chev integration.c, 111 constants.h caps_integrate_plasma cAPS_c, 63 cAPS_c, 63 cAPS_c, 63 cAPS_bhar eV, 63 cAPS_bhar, 63 caps_integrate_plasma_free cAPS_bhar, 63 integration.c, 114 m_LOGP, 63 daps_integrate_plasma_init m_LOGP, 63 libcaps.h, 80 m_PI, 64 caps_kernel_MO_MM_bloaps.h m_API, 64 caps_kernel_MO_MM_bloaps.h	caps_integrate_A	libcaps.h, 94
integration.c, 106 caps_integrate_C caps_integrate_D integration.c, 107 caps_integrate_D integration.c, 108 caps_integrate_free integration.c, 108 caps_integrate_I	integration.c, 104	
caps_integrate_C caps_set_epsrel integration.c, 106 libcaps.c, 146 caps_integrate_D libcaps.c, 146 integration.c, 107 caps_set_ldim caps_integrate_free libcaps.c, 146 integration.c, 108 libcaps.h, 96 caps_integrate_li caps_tellibraps.h, 96 caps_integrate_init caps_tellibraps.h, 88 caps_integrate_init caps_tellibraps.h, 88 caps_integrate_init caps_tellibraps.h, 88 caps_integrate_init constants.h caps_integrate_plasma constants.h caps_integrate_plasma caps_tellibraps.h integration.c, 111 constants.h caps_integrate_plasma_free caps_integrate_plasma_free integration.c, 113 caps_has_e.83 caps_integrate_plasma_init constants.h integration.c, 114 caps_integrate_plasma_init mLOG2, 63 integration.c, 114 desp_kernel_MO_EE MAN, 64 libcaps.h, 80 desp_kernel_MO_MM_plasma pow_2, 64 libcaps.h, 81 cotal caps_kernel_MO_MM desp_kernel_MO_MM	caps_integrate_B	libcaps.c, 145
integration.c, 106 caps integrate_D	integration.c, 106	•
caps_integrate_D libcaps.h, 96 integration.c, 107 caps_set_Idim caps_integrate_free libcaps.c, 146 integration.c, 108 libcaps.h, 96 caps_integrate_I caps_t integration.c, 109 libcaps.h, 68 caps_integrate_init caps_task_t, 14 integration.c, 110 chbevl caps_integrate_plasma cAPS_c, 63 integration.c, 111 constants.h caps_integrate_plasma cAPS_bar, 63 caps_integrate_plasma cAPS_bar, 63 caps_integrate_plasma free cAPS_hbar, 63 integration.c, 112 cAPS_hbar, 63 caps_integrate_plasma_intere cAPS_hbar, 63 integration.c, 113 M_LOG2, 63 daps_integrate_plasma_intere m_LOGPI, 63 libcaps.c, 131 M_LOG2P, 63 M_PI, 64 MAX, 64 MIN, 64 MP, 64 MAX, 64 MIN, 64 SGN, 64 SGN, 64 sibcaps.c, 132 sign_t, 64 libcaps.c, 136 caps_mit_b libcaps.c, 136 mat	caps_integrate_C	. — — .
integration.c, 107 caps_integrate_free integration.c, 108 caps_integrate_free integration.c, 109 caps_integrate_li integration.c, 109 caps_integrate_lii integration.c, 110 caps_integrate_lii integration.c, 110 caps_integrate_lii caps_integrate_plasma integration.c, 111 caps_integrate_plasma integration.c, 112 caps_integrate_plasma integration.c, 113 caps_integrate_plasma integration.c, 113 caps_integrate_plasma_free integration.c, 114 caps_integrate_plasma_init integration.c, 114 caps_kernel_M0_EE ilibcaps.c, 131 ilibcaps.h, 80 caps_kernel_M0_MM_plasma ilibcaps.c, 132 ilibcaps.h, 81 caps_kernel_M0_MM ilibcaps.c, 132 ilibcaps.h, 81 caps_kernel_M0 ilibcaps.c, 130 ilibcaps.h, 83 caps_logdetD0 ilibcaps.c, 136 ili		•
caps_integrate_free integration.c, 108 libcaps.c, 146 libcaps.h, 96 caps_integrate_l integration.c, 109 caps_integrate_l libcaps.h, 68 caps_integrate_init integration.c, 110 caps_lask_t, 14 caps_integrate_K integration.c, 111 caps_integrate_lasma caps_integrate_plasma integration.c, 112 caps_integrate_plasma_fee caps_integrate_plasma_intere integration.c, 113 caps_bar_eV, 63 caps_integrate_plasma_init integration.c, 114 m_LOG2, 63 caps_integrate_plasma_init integration.c, 113 m_LOG2, 63 caps_kernel_M0_EE M_PI, 64 libcaps.c, 131 M_LOG2, 63 libcaps.h, 80 M_PI, 64 caps_kernel_M0_MM_plasma pow_2, 64 libcaps.h, 82 caps_integrate_lasma_init integration.c, 113 caps_kernel_M0_MM cot2 caps_kernel_M0_MM_plasma pow_2, 64 libcaps.h, 82 caps_integrate_lasma_init integration.c, 113 caps_kernel_M0_MM fcqs.c, 56 caps_kernel_M0_MM fcqs.c, 56 libcaps.h, 81 caps_integrate_lasma_init integration.c, 113 caps_kernel_M0_MM fcqs.c, 56 caps_integrate_lasma_init integration.		•
integration.c, 108 caps_integrate_I integration.c, 109 caps_integrate_init integrate_init integrate_init integrate_init integration.c, 110 caps_integrate_K integrate_plasma integration.c, 111 caps_integrate_plasma integration.c, 112 caps_integrate_plasma integration.c, 113 caps_integrate_plasma_int integration.c, 113 caps_integrate_plasma_int integration.c, 114 caps_kenel_MO_EE libcaps.c, 131 libcaps.c, 136 libcaps.c, 130 libcaps.c, 136	_	• — —
caps_integrate_I integration.c., 109 ilbcaps.h, 68 caps_integrate_init caps_task_t, 14 integration.c, 110 caps_task_t, 14 caps_integrate_K chbevI integration.c, 111 constants.h caps_integrate_plasma CAPS_c, 63 integration.c, 112 CAPS_hbar, 63 caps_integrate_plasma_free CAPS_hbar_eV, 63 integration.c, 113 M_LOG2, 63 caps_integrate_plasma_init M_LOGPI, 63 integration.c, 114 M_PI, 64 caps_kernel_M0_EE MAX, 64 libcaps.c, 131 MIN, 64 libcaps.c, 133 SGN, 64 libcaps.c, 133 SGN, 64 libcaps.c, 132 fcqs.c, 56 libcaps.n, 81 cot2 caps_kernel_M0_MM detalg libcaps.c, 130 caps_indeprate_indepr	• - • -	•
integration.c, 109 caps_integrate_init integration.c, 110 caps_integrate_K integration.c, 111 caps_integrate_K integration.c, 111 caps_integrate_blasma integration.c, 112 caps_integrate_plasma integration.c, 112 caps_integrate_plasma integration.c, 112 caps_integrate_plasma_free integration.c, 113 caps_integrate_plasma_free integration.c, 113 caps_integrate_plasma_init integration.c, 114 caps_kernel_M0_EE ilibcaps.c, 131 ilibcaps.c, 131 ilibcaps.h, 80 caps_kernel_M0_MM_plasma ilibcaps.h, 82 caps_kernel_M0_MM_plasma ilibcaps.h, 81 caps_kernel_M0 ilibcaps.c, 132 ilibcaps.h, 81 caps_kernel_M0 ilibcaps.c, 132 ilibcaps.c, 130 ilibcaps.c, 130 ilibcaps.c, 130 ilibcaps.c, 134 ilibcaps.c, 136 ilibcaps.c, 136 ilibcaps.c, 136 ilibcaps.c, 136 ilibcaps.c, 136 ilibcaps.c, 136 ilibcaps.c, 135 ilibcaps.c, 136 ilibcap	-	•
caps_integrate_init integration.c, 110 caps_task_t, 14 chbevl caps_integrate_K integration.c, 111 constants.h caps_integrate_plasma integration.c, 112 constants.h caps_integrate_plasma integration.c, 113 cAPS_c, 63 caps_integrate_plasma_free integration.c, 114 cAPS_hbar, 63 caps_integrate_plasma_init integration.c, 114 M_LOG2, 63 caps_kernel_M0_EE M_PI, 64 libcaps.c, 131 MMX, 64 libcaps.c, 130 MIN, 64 libcaps.c, 133 SGN, 64 libcaps.n, 82 cquadpack/include/quadpack.h, 49 caps_kernel_M0_MM fcqs.c, 56 libcaps.n, 81 cquadpack/include/quadpack.h, 49 caps_kernel_M detalg caps_lnLambda matrix_t, 20 libcaps.c, 130 caps_integrate_plasma_init libcaps.c, 134 dim libcaps.c, 135 matrix_t, 20 dim2 matrix_t, 20 dim2ps_cla		
Chbev Chbev Chbev Chbev Chbev Chbev Chbev Caps_integrate_K integration.c, 111 Caps_integrate_plasma CAPS_c, 63 CAPS_hbar, 63 CAPS_hbar, 63 CAPS_hbar, 63 CAPS_hbar_eV, 63 CAPS_hbar_eV, 63 CAPS_kB, 64 CAPS_kB, 63 CAPS_kB, 64 CAPS_kB, 63 CAPS_kB, 63 CAPS_kB, 64 CAPS_kB, 63 CAPS_kB, 64 CAPS_kB, 64 CAPS_kB, 64 CAPS_kB, 63 CAPS_kB, 64	•	•
Caps_integrate_K	• - • -	• — —
Caps_integrate_plasma		********
CAPS_c, 63		,
CAPS_hbar, 63	_	
caps_integrate plasma_free CAPS_hbar_eV, 63 caps_integrate plasma_init M_LOG2, 63 integration.c, 114 M_LOGPI, 63 caps_kernel_M0_EE M_PI, 64 libcaps.c, 131 MAX, 64 libcaps.b, 80 MAX, 64 caps_kernel_M0_MM_plasma pow_2, 64 libcaps.c, 133 SGN, 64 libcaps.b, 82 sign_t, 64 caps_kernel_M0_MM cot2 fcqs.c, 56 cquadpack/include/quadpack.h, 49 libcaps.h, 81 caps_ln_ln_ln_lde/quadpack.h, 49 caps_kernel_M detalg libcaps.h, 79 detalg caps_ln_ln_ambda matrix_t, 20 libcaps.h, 83 matrix_t, 20 caps_logdetD0 disable_buffering libcaps.h, 85 utils.h, 99 caps_logdetD dlnPlm libcaps.h, 84 dagg caps_mie_perf dagg libcaps.c, 140 dipage libcaps.c, 142 quadpack.h, 52 dibcaps.h, 90 quadpack.h, 52 caps_mie_perf daggs		= '
integration.c, 113 caps_integrate_plasma_init integration.c, 114 caps_kernel_M0_EE libcaps.c, 131 libcaps.h, 80 caps_kernel_M0_MM_plasma libcaps.c, 133 libcaps.h, 82 caps_kernel_M0_MM libcaps.c, 132 libcaps.h, 81 caps_kernel_M libcaps.c, 130 libcaps.h, 81 caps_kernel_M libcaps.c, 130 libcaps.h, 79 caps_lnLambda libcaps.c, 134 libcaps.h, 83 caps_logdetD0 libcaps.c, 136 libcaps.h, 85 caps_logdetD libcaps.c, 135 libcaps.h, 84 caps_mie libcaps.c, 140 libcaps.c, 142 libcaps.h, 92 caps_mpi_t, 13 caps_logdetQ caps_mie libcaps.c, 143 libcaps.c, 148 libcaps.c, 1	_	
caps_integrate_plasma_init integration.c, 114 M_LOG2, 63 caps_kernel_M0_EE M_PI, 64 libcaps.c, 131 MIN, 64 libcaps.c, 80 MIN, 64 caps_kernel_M0_MM_plasma pow_2, 64 libcaps.c, 133 sgn_t, 64 libcaps.h, 82 cot2 caps_kernel_M0_MM fcqs.c, 56 libcaps.h, 81 cquadpack/include/quadpack.h, 49 caps_kernel_M detalg libcaps.c, 130 caps, 11 libcaps.h, 79 dim dibcaps.c, 134 dim2 libcaps.c, 134 dim2 libcaps.c, 136 utils.c, 192 libcaps.h, 85 utils.n, 99 caps_logdetD dlnPlm libcaps.c, 135 plm.c, 181 libcaps.c, 140 dqage libcaps.c, 140 dqagi libcaps.c, 142 quadpack.h, 52 libcaps.c, 142 quadpack.h, 52 libcaps.mpj.t, 13 quadpack.h, 53 caps_mpj.t, 13 epsmj.t, 13 caps_mpj.t, 13 epsm1		
Caps_Integration.c, 114	_	- ·
mitegrations, 14, 14 claps_kernel_M0_EE		-
Caps_kernie_mu_EL MAX, 64 libcaps.c, 131 MIN, 64 caps_kernel_M0_MM_plasma pow_2, 64 libcaps.c, 133 sign_t, 64 caps_kernel_M0_MM plasma pow_2, 64 libcaps.c, 133 sign_t, 64 caps_kernel_M0_MM cot2 caps_kernel_M0_MM libcaps.c, 132 cquadpack/include/quadpack.h, 49 caps_kernel_M detalg caps, 11 libcaps.c, 130 caps, 11 libcaps.h, 79 dim matrix_t, 20 disaps_ln_Lambda matrix_t, 20 libcaps.h, 83 matrix_t, 20 libcaps.h, 85 utils.c, 192 utils.c, 192 utils.h, 99 caps_logdetD dlnPlm libcaps.c, 135 plm.c, 181 dagge quadpack.h, 51 libcaps.h, 84 dqage caps_mie quadpack.h, 51 libcaps.c, 140 dqagi libcaps.c, 142 quadpack.h, 52 dags quadpack.h, 52 dags quadpack.h, 53 disaps_set_detalg libcaps.c, 143 epsm1	_	
Ilibcaps.h, 80	• – – –	- :
caps_kernel_M0_MM_plasma libcaps.c, 133 libcaps.h, 82 caps_kernel_M0_MM libcaps.c, 132 libcaps.h, 81 caps_kernel_M libcaps.c, 130 libcaps.h, 81 caps_kernel_M libcaps.c, 130 libcaps.h, 79 caps_lnLambda libcaps.c, 134 libcaps.h, 83 caps_logdetD0 libcaps.c, 136 libcaps.h, 85 caps_logdetD libcaps.c, 135 libcaps.h, 84 caps_mie libcaps.c, 140 libcaps.c, 140 libcaps.c, 142 libcaps.h, 92 caps_mpi_t, 13 caps_set_detalg libcaps.c, 143 libcaps.c, 144 libcaps.c, 143 libcaps.c, 143 libcaps.c, 144 libcaps.c, 145 libcaps.c, 145 libcaps.c, 146 libcaps.c, 148 libcaps.c, 1	•	
SGN, 64 sign_t, 64 sign_t	·	
libcaps.h, 82 caps_kernel_M0_MM libcaps.c, 132 libcaps.h, 81 caps_kernel_M libcaps.c, 130 libcaps.h, 79 caps_InLambda libcaps.c, 134 libcaps.h, 83 caps_logdetD0 libcaps.h, 85 caps_logdetD libcaps.c, 135 libcaps.h, 84 caps_mie libcaps.c, 140 libcaps.c, 140 libcaps.c, 142 libcaps.h, 92 caps_mie, 13 caps_logdetD caps_mie libcaps.c, 142 libcaps.h, 92 caps_mpi_t, 13 caps_set_detalg libcaps.c, 143 sign_t, 64 cot2 fcqs.c, 56 cquadpack/include/quadpack.h, 49 detalg caps, 11 dim matrix_t, 20 dim2 matrix_t, 20 disable_buffering utils.c, 192 utils.h, 99 dlnPlm plm.c, 181 dqage quadpack.h, 51 dqagi quadpack.h, 51 dqagi quadpack.h, 52 dqags quadpack.h, 52 detemr_ psd.c, 189	. – – – –	• —
caps_kernel_M0_MM libcaps.c, 132 libcaps.h, 81 caps_kernel_M libcaps.c, 130 libcaps.h, 79 caps_InLambda libcaps.c, 134 libcaps.h, 83 caps_logdetD0 libcaps.c, 136 libcaps.h, 85 caps_logdetD libcaps.c, 135 libcaps.h, 84 caps_mie libcaps.c, 140 libcaps.c, 140 libcaps.c, 142 libcaps.h, 92 caps_mie_leff libcaps.c, 142 libcaps.c, 143 caps_sernel_M caps_mie libcaps.c, 142 libcaps.c, 143 caps_ser_detalg libcaps.c, 143 caps_ser_detalg libcaps.c, 143 caps_ser_detalg libcaps.c, 143 caps_mie libcaps.c, 143 caps_mie_leff libcaps.c, 142 libcaps.h, 92 caps_mpi_t, 13 caps_ser_detalg libcaps.c, 143 caps_mie libcaps.c, 143 caps_mie_leff libcaps.c, 142 libcaps.h, 92 caps_mpi_t, 13 caps_ser_detalg libcaps.c, 143 caps_mie libcaps.c, 143 caps_mie_leff libcaps.c, 142 libcaps.h, 92 caps_mpi_t, 13 caps_ser_detalg libcaps.c, 143 caps_mie_lepenf libcaps.c, 143 caps_mie_left libcaps.c, 142 libcaps.h, 92 caps_mpi_t, 13 caps_ser_detalg libcaps.c, 143 caps_mie_lepenf libcaps.c, 144 libcaps.c, 145 caps_mpi_t, 13 caps_ser_detalg libcaps.c, 143 caps_mie_lepenf libcaps.c, 143 caps_mpi_t, 13 caps_ser_detalg libcaps.c, 143 caps_mpi_t, 13 caps_ser_detalg libcaps.c, 143 caps_mpi_t, 13 caps_mpi_t,	•	•
fcqs.c, 56 cquadpack/include/quadpack.h, 49 libcaps.h, 81 detalg caps, 11 dim matrix_t, 20 dim2 matrix_t, 20 disable_buffering utils.c, 192 utils.h, 99 dlnPlm libcaps.c, 135 libcaps.h, 84 dage quadpack.h, 51 dage quadpack.h, 51 dage quadpack.h, 52 dags quadpack.h, 52 dags quadpack.h, 52 dags quadpack.h, 53 dags_set_detalg libcaps.c, 143 epsm1 epsm2 epsm1 epsm1 epsm2 epsm1 epsm2 epsm1 epsm2 epsm1 epsm2 epsm2 epsm3	·	- -
libcaps.h, 81 caps_kernel_M libcaps.c, 130 libcaps.h, 79 caps_lnLambda libcaps.c, 134 libcaps.h, 83 caps_logdetD0 libcaps.c, 135 libcaps.h, 85 caps_logdetD libcaps.c, 135 libcaps.h, 84 caps_mie libcaps.c, 140 libcaps.c, 140 libcaps.c, 142 libcaps.h, 92 caps_mie_perf libcaps.c, 143 caps_set_detalg libcaps.c, 143 caps_mie cquadpack/include/quadpack.h, 49 detalg caps, 11 dim matrix_t, 20 disable_buffering utils.c, 192 utils.h, 99 dlinPlm plm.c, 181 dqage quadpack.h, 51 dqage quadpack.h, 51 dqage quadpack.h, 52 dags quadpack.h, 52 deterr_ psd.c, 189	• – – –	
caps_kernel_M	•	•
libcaps.c, 130 libcaps.h, 79 caps_InLambda libcaps.c, 134 libcaps.h, 83 caps_logdetD0 libcaps.c, 136 libcaps.h, 85 caps_logdetD libcaps.c, 135 libcaps.h, 84 caps_mie libcaps.c, 140 libcaps.h, 90 caps_mie_perf libcaps.c, 142 libcaps.h, 92 caps_mpi_t, 13 caps_set_detalg libcaps.c, 143 libcaps.c, 143 libcaps.c, 143 libcaps.d, 92 caps_mpi_t, 13 caps_set_detalg libcaps.c, 143 limatrix_t, 20 dim2 matrix_t, 20 matrix_t, 20 matrix_t, 20 dim2 matrix_t, 20 matrix_t, 2	•	
libcaps.h, 79 caps_InLambda	. – –	detalg
caps_InLambda matrix_t, 20 libcaps.c, 134 dim2 libcaps.h, 83 matrix_t, 20 caps_logdetD0 disable_buffering libcaps.c, 136 utils.c, 192 libcaps.h, 85 utils.h, 99 caps_logdetD dlnPlm libcaps.c, 135 plm.c, 181 libcaps.h, 84 dqage caps_mie quadpack.h, 51 libcaps.c, 140 dqagi libcaps.h, 90 quadpack.h, 52 caps_mie_perf dqags libcaps.c, 142 quadpack.h, 53 libcaps.h, 92 dstemr_ caps_mpi_t, 13 psd.c, 189 caps_set_detalg libcaps.c, 143	•	caps, 11
libcaps.c, 134 libcaps.h, 83 caps_logdetD0 libcaps.c, 136 libcaps.h, 85 caps_logdetD libcaps.c, 135 libcaps.h, 84 caps_mie libcaps.c, 140 libcaps.h, 90 caps_mie_perf libcaps.c, 142 libcaps.h, 92 caps_mpi_t, 13 caps_set_detalg libcaps.c, 143 libcaps.c, 144 libcaps.c, 145 libcaps.c, 145 libcaps.c, 145 libcaps.c, 145 libcaps.c, 146 libca	•	dim
libcaps.h, 83 caps_logdetD0 libcaps.c, 136 libcaps.h, 85 caps_logdetD libcaps.c, 135 libcaps.h, 84 caps_mie libcaps.c, 140 libcaps.h, 90 caps_mie_perf libcaps.c, 142 libcaps.h, 92 caps_mpi_t, 13 caps_set_detalg libcaps.c, 143 matrix_t, 20 disable_buffering utils.c, 192 utils.h, 99 dlnPlm plm.c, 181 dqage quadpack.h, 51 dqagi quadpack.h, 51 dqagi quadpack.h, 52 dags quadpack.h, 52 dags quadpack.h, 53 dstemr_ psd.c, 189	· -	matrix_t, 20
caps_logdetD0 libcaps.c, 136 libcaps.h, 85 caps_logdetD libcaps.c, 135 libcaps.h, 84 caps_mie libcaps.c, 140 libcaps.h, 90 caps_mie_perf libcaps.c, 142 libcaps.h, 92 caps_mpi_t, 13 caps_set_detalg libcaps.c, 143 disable_buffering utils.c, 192 utils.h, 99 dlnPlm plm.c, 181 dqage quadpack.h, 51 dqagi quadpack.h, 52 dqags quadpack.h, 52 dstemr_ psd.c, 189	•	dim2
libcaps.c, 136 libcaps.h, 85 caps_logdetD libcaps.h, 84 caps_mie libcaps.c, 140 libcaps.h, 90 caps_mie_perf libcaps.c, 142 libcaps.h, 92 caps_mpi_t, 13 caps_set_detalg libcaps.c, 143 libcaps.c, 143 libcaps.c, 143 libcaps.c, 148 libcaps.c, 149 libcaps.d, 92 caps_mie_t, 13 caps_set_detalg libcaps.c, 143 lib	•	matrix_t, 20
libcaps.h, 85 caps_logdetD libcaps.c, 135 libcaps.h, 84 caps_mie libcaps.c, 140 libcaps.h, 90 caps_mie_perf libcaps.c, 142 libcaps.h, 92 caps_mpi_t, 13 caps_set_detalg libcaps.c, 143 libcaps.c, 143 utils.h, 99 dlnPlm plm.c, 181 dqage quadpack.h, 51 dqagi quadpack.h, 52 dags quadpack.h, 52 dstemr_ psd.c, 189	· — ·	disable_buffering
caps_logdetD libcaps.c, 135 libcaps.h, 84 caps_mie libcaps.c, 140 libcaps.h, 90 caps_mie_perf libcaps.c, 142 libcaps.c, 142 libcaps.h, 92 caps_mpi_t, 13 caps_set_detalg libcaps.c, 143 dqage quadpack.h, 51 dqagi quadpack.h, 52 dqags quadpack.h, 53 dstemr_ psd.c, 189	•	
libcaps.c, 135 libcaps.h, 84 caps_mie libcaps.c, 140 libcaps.h, 90 caps_mie_perf libcaps.c, 142 libcaps.h, 92 caps_mpi_t, 13 caps_set_detalg libcaps.c, 143 caps_min plm.c, 181 dqage quadpack.h, 51 dqagi quadpack.h, 52 dqags quadpack.h, 53 dstemr_ psd.c, 189	•	
libcaps.h, 84 caps_mie	• - •	
caps_mie quadpack.h, 51 libcaps.c, 140 libcaps.h, 90 quadpack.h, 52 caps_mie_perf dqags libcaps.c, 142 libcaps.h, 92 quadpack.h, 53 libcaps.h, 92 caps_mpi_t, 13 psd.c, 189 caps_set_detalg libcaps.c, 143 epsm1	•	•
libcaps.c, 140 libcaps.h, 90 caps_mie_perf libcaps.c, 142 libcaps.h, 92 caps_mpi_t, 13 caps_set_detalg libcaps.c, 143 dqagi quadpack.h, 52 dqags quadpack.h, 53 dstemr_ psd.c, 189	•	
libcaps.h, 90 quadpack.h, 52 caps_mie_perf dqags libcaps.c, 142 quadpack.h, 53 libcaps.h, 92 dstemr_ caps_mpi_t, 13 psd.c, 189 caps_set_detalg libcaps.c, 143 epsm1	• —	
caps_mie_perf libcaps.c, 142 libcaps.h, 92 caps_mpi_t, 13 caps_set_detalg libcaps.c, 143 dqags quadpack.h, 53 dstemr_ psd.c, 189 epsm1	•	
libcaps.c, 142 quadpack.h, 53 libcaps.h, 92 dstemr_ caps_mpi_t, 13 psd.c, 189 caps_set_detalg libcaps.c, 143 epsm1	•	
libcaps.h, 92 caps_mpi_t, 13 caps_set_detalg libcaps.c, 143 dstemr_ psd.c, 189 epsm1	• — —•	
caps_mpi_t, 13 psd.c, 189 caps_set_detalg libcaps.c, 143 epsm1	·	
caps_set_detalg libcaps.c, 143 epsm1	·	-
libcaps.c, 143 epsm1		psd.c, 189
·		onem1
iibcaps.ii, 30	•	•
	iibcaps.ii, 30	matemai_t, 10

epsrel	caps_integrate_init, 110
caps, 11	caps_integrate_K, 111
	caps_integrate_plasma, 112
fcqs.c, 55	caps_integrate_plasma_free, 113
cot2, 56	caps_integrate_plasma_init, 114
fcqs_finite, 56	K_estimate, 115
fcqs_semiinf, 57	integration_plasma_t, 15
MMIN, 56	integration t, 16
wi_finite, 58	
wi_semiinf, 58	K0_coeffsA
fcqs_finite	bessel.c, 44
fcqs.c, 56	K0_coeffsB
fcqs_semiinf	bessel.c, 44
fcqs.c, 57	K1_coeffsA
filename	bessel.c, 44
material t, 18	K1_coeffsB
	bessel.c, 45
GK_10_21	K estimate
quadpack.h, 50	integration.c, 115
GK_15_31	kahan_sum
quadpack.h, 50	misc.c, 174
GK 20 41	kernel_args_t, 17
quadpack.h, 50	kernel_logdet
GK_25_51	matrix.c, 158
quadpack.h, 50	
GK_30_61	L
quadpack.h, 51	caps, 11
GK_7_15	LbyR
quadpack.h, 51	caps, 11
gamma_high	lda
material_t, 18	matrix t, 20
gamma low	ldim
material_t, 19	caps, 11
material_t, 10	Ifac
hodlr.h	logfac.c, 150
hodir_logdet, 148	Ifac2
hodir_logdet_diagonal, 148	logfac.c, 151
hodir_logdet	libcaps.c, 116
hodir.h, 148	caps_M_elem, 137
hodir_logdet_diagonal	caps_M_free, 139
hodir.h, 148	caps_M_init, 139
nodi.n, 140	caps build, 119
I0 coeffs	caps_epsilonm1_drude, 119
bessel.c, 42	caps_epsilonm1_perf, 120
I1 coeffs	caps_epsilonm1_plate, 120
bessel.c, 43	caps_epsilonm1_sphere, 121
include/buf.h, 59	caps_estimate_lminmax, 122
include/constants.h, 62	caps_free, 122
include/libcaps.h, 65	caps_fresnel, 124
include/utils.h, 97	caps_get_detalg, 124
integrand_plasma_t, 14	,
integrand_t, 14	caps_get_epsrel, 125 caps_get_ldim, 125
integration.c, 103	caps_get_idim, 125 caps_ht_drude, 125
caps_integrate_A, 104	caps_ht_perf, 126
	caps_nt_pen, 126 caps_ht_plasma, 127
caps_integrate_B, 106	• — —•
caps_integrate_C, 106	caps_info, 128
caps_integrate_D, 107	caps_init, 129
caps_integrate_free, 108	caps_kernel_M0_EE, 131
caps_integrate_I, 109	caps_kernel_M0_MM_plasma, 133

	L L MO AMA 400	1.00
	caps_kernel_M0_MM, 132	plm.c, 183
	caps_kernel_M, 130	InPIm_downwards
	caps_InLambda, 134	plm.c, 184
	caps_logdetD0, 136	InPlm_upwards
	caps_logdetD, 135	plm.c, 185
	caps_mie, 140	log_t, 17
	caps_mie_perf, 142	s, 17
	caps_set_detailg, 143	v, 17
	caps_set_epsilonm1, 144	logadd misc.c, 175
	caps_set_epsilonm1_plate, 144	logadd ms
	caps_set_epsilonm1_sphere, 145	misc.c, 175
	caps_set_epsrel, 146	logfac.c, 149
libos	caps_set_ldim, 146	Ifac, 150
IIDCa	ADS CACHE FLEMS 60	Ifac2, 151
	CAPS_CACHE_ELEMS, 68	logi, 152
	CAPS_EPSREL, 68	lookup_lfac, 152
	CAPS_FACTOR_LDIM, 68	lookup logi, 153
	CAPS_MINIMUM_LDIM, 68	logi
	caps_M_elem, 87	logfac.c, 152
	caps_M_free, 88	lookup Ifac
	caps_M_init, 89	logfac.c, 152
	caps_build, 69	lookup_logi
	caps_epsilonm1_drude, 69	logfac.c, 153
	caps_epsilonm1_perf, 70	109140.0, 100
	caps_epsilonm1_plate, 71	M
	caps_epsilonm1_sphere, 71	matrix_t, 20
	caps_estimate_lminmax, 72	M_LOG2
	caps_free, 73	constants.h, 63
	caps_fresnel, 73	M_LOGPI
	caps_get_detalg, 74	constants.h, 63
	caps_get_epsrel, 74	M_PI
	caps_get_ldim, 74	constants.h, 64
	caps_ht_drude, 75	MAX
	caps_ht_perf, 76	constants.h, 64
	caps_ht_plasma, 77	MIN
	caps_info, 78	constants.h, 64
	caps_init, 78	MMIN
	caps_kernel_M0_EE, 80	fcqs.c, 56
	caps_kernel_M0_MM_plasma, 82	material.c, 153
	caps_kernel_M0_MM, 81	_parse, 154
	caps_kernel_M, 79	material_epsilonm1, 154
	caps_InLambda, 83	material_free, 155
	caps_logdetD0, 85	material_get_extrapolation, 155
	caps_logdetD, 84	material_info, 156
	caps_mie, 90	material_init, 156
	caps_mie_perf, 92	material_epsilonm1
	caps_set_detalg, 93	material.c, 154
	caps_set_epsilonm1, 94	material_free
	caps_set_epsilonm1_plate, 94	material.c, 155
	caps_set_epsilonm1_sphere, 95	material_get_extrapolation
	caps_set_epsrel, 96	material.c, 155
	caps_set_ldim, 96	material_info
	caps_t, 68	material.c, 156
194-1	polarization_t, 69	material_init
	odlr/include/hodlr.h, 147	material.c, 156
InPl		material_t, 18
le Di	plm.c, 182	calL, 18
InPli	III	epsm1, 18

filename, 18	matrix_t, 20
gamma_high, 18	dim, 20
gamma_low, 19	dim2, 20
omegap_high, 19	lda, 20
omegap_low, 19	M, 20
points, 19	matrix_trace
xi, 19	matrix.c, 171
xi_max, 19	matrix_trace2
xi_min, 19	matrix.c, 172
matrix.c, 157	misc.c, 173
kernel_logdet, 158	kahan_sum, 174
matrix_alloc, 159	logadd, 175
matrix_copy, 160	logadd_ms, 175
matrix_free, 161	sqrtpm1, 176
matrix_load_from_file, 162	2011
matrix_load_from_stream, 162	now
matrix_logdet_cholesky, 163	utils.c, 192
matrix_logdet_dense, 164	utils.h, 99
matrix_logdet_lu, 165	omegap_high
matrix_logdet_qr, 166	material_t, 19
matrix_logdet_triangular, 167	omegap_low
matrix_mult, 168	material_t, 19
matrix_norm_frobenius, 169	material_t, 10
matrix_save_to_file, 169	plm.c, 177
matrix_save_to_stream, 170	PI1, 178
matrix_setall, 170	Pl2, 179
matrix_trace, 171	Pl3, 180
matrix_trace2, 172	fn, 178
matrix_alloc	dlnPlm, 181
matrix.c, 159	InPl, 182
matrix_copy	InPlm, 183
matrix.c, 160	InPlm_downwards, 184
matrix_free	InPlm_upwards, 185
matrix.c, 161	Plm_continued_fraction, 186
matrix_load_from_file	Plm_continued_fraction
matrix.c, 162	plm.c, 186
matrix_load_from_stream	points
matrix.c, 162	material_t, 19
matrix_logdet_cholesky	polarization_t
matrix.c, 163	libcaps.h, 69
matrix_logdet_dense	pow_2
matrix.c, 164	constants.h, 64
matrix_logdet_lu	psd
matrix.c, 165 matrix_logdet_qr	
mairix loodel or	psd.c, 189
	psd.c, 189 psd.c, 187
matrix.c, 166	•
matrix.c, 166 matrix_logdet_triangular	psd.c, 187
matrix.c, 166 matrix_logdet_triangular matrix.c, 167	psd.c, 187 _eta, 188
matrix.c, 166 matrix_logdet_triangular matrix.c, 167 matrix_mult	psd.c, 187 _eta, 188 dstemr_, 189 psd, 189
matrix.c, 166 matrix_logdet_triangular matrix.c, 167 matrix_mult matrix.c, 168	psd.c, 187 eta, 188 dstemr, 189 psd, 189 quadpack.h
matrix.c, 166 matrix_logdet_triangular matrix.c, 167 matrix_mult matrix.c, 168 matrix_norm_frobenius	psd.c, 187eta, 188dstemr, 189psd, 189 quadpack.hdqage, 51
matrix.c, 166 matrix_logdet_triangular matrix.c, 167 matrix_mult matrix.c, 168 matrix_norm_frobenius matrix.c, 169	psd.c, 187eta, 188dstemr, 189psd, 189 quadpack.hdqage, 51dqagi, 52
matrix.c, 166 matrix_logdet_triangular matrix.c, 167 matrix_mult matrix.c, 168 matrix_norm_frobenius matrix.c, 169 matrix_save_to_file	psd.c, 187eta, 188dstemr, 189psd, 189 quadpack.hdqage, 51dqagi, 52dqags, 53
matrix.c, 166 matrix_logdet_triangular matrix.c, 167 matrix_mult matrix.c, 168 matrix_norm_frobenius matrix.c, 169 matrix_save_to_file matrix.c, 169	psd.c, 187eta, 188dstemr, 189psd, 189 quadpack.hdqage, 51dqagi, 52dqags, 53GK1021, 50
matrix.c, 166 matrix_logdet_triangular matrix.c, 167 matrix_mult matrix.c, 168 matrix_norm_frobenius matrix.c, 169 matrix_save_to_file matrix_c, 169 matrix_save_to_stream	psd.c, 187eta, 188dstemr, 189psd, 189 quadpack.hdqage, 51dqagi, 52dqags, 53GK1021, 50GK1531, 50
matrix.c, 166 matrix_logdet_triangular matrix.c, 167 matrix_mult matrix.c, 168 matrix_norm_frobenius matrix.c, 169 matrix_save_to_file matrix.c, 169 matrix_save_to_stream matrix.c, 170	psd.c, 187eta, 188dstemr, 189 _ psd, 189 quadpack.hdqage, 51dqagi, 52dqags, 53GK1021, 50GK1531, 50GK2041, 50
matrix.c, 166 matrix_logdet_triangular matrix.c, 167 matrix_mult matrix.c, 168 matrix_norm_frobenius matrix.c, 169 matrix_save_to_file matrix_c, 169 matrix_save_to_stream	psd.c, 187eta, 188dstemr, 189psd, 189 quadpack.hdqage, 51dqagi, 52dqags, 53GK1021, 50GK1531, 50

G	GK_7_15, 51	xcall	
R c	aps, 11	xfree	utils.c, 193 utils.h, 101
s		xi	utils.h, 99
lo SGN	og_t, 17	xi_m	material_t, 19
c sign_t	onstants.h, 64	xi_m	material_t, 19
c size	onstants.h, 64	xmal	material_t, 19
b sqrtpn	ouf, 9 n1	λιτιαι	utils.c, 194 utils.h, 101
m strim	nisc.c, 176	xrea	
	ıtils.c, 192 ıtils.h, 100		utils.h, 102
strrep u	itils.c, 192	у	caps, 12
	tils.h, 100		σαρό, τΖ
u time_a u	IINATE tils.h, 98 as_string tils.c, 193 tils.h, 100		
n s s ti x x x x utils.h C d n s s T ti V x x x x x	lisable_buffering, 192 how, 192 ttrim, 192 ttrrep, 192 me_as_string, 193 calloc, 193 malloc, 194 trealloc, 194		
v Id	og_t, 17		
WARN	N Itils.h, 99		
wi_fini	ite		
wi_ser	cqs.c, 58 miinf cqs.c, 58		