

spectra

Generated by Doxygen 1.8.17

1 Data Type Index	1
1.1 Data Types List	1
2 File Index	3
2.1 File List	3
3 Data Type Documentation	5
3.1 Input Struct Reference	5
3.1.1 Detailed Description	5
3.1.2 Field Documentation	6
3.1.2.1 aw_file	6
3.1.2.2 eigvals_file	6
3.1.2.3 eigvecs_file	6
3.1.2.4 fw_file	6
3.1.2.5 gamma_file	6
3.1.2.6 gi_files	6
3.1.2.7 lambda_file	6
3.1.2.8 mu_file	7
3.1.2.9 N	7
3.1.2.10 pop_file	7
3.1.2.11 T	7
3.1.2.12 tau	7
3.2 ode_params Struct Reference	7
3.2.1 Field Documentation	8
3.2.1.1 chiw	8
3.2.1.2 kij	8
3.2.1.3 N	8
3.2.1.4 rates	8
3.2.1.5 Tij	8
3.3 Parameters Struct Reference	8
3.3.1 Field Documentation	9
3.3.1.1 aw_file	9
3.3.1.2 cn	9
3.3.1.3 cw	9
3.3.1.4 fw_file	10
3.3.1.5 g0	10
3.3.1.6 g1	10
3.3.1.7 g2	10
3.3.1.8 gsw	10
3.3.1.9 gt_file	10
3.3.1.10 l0	10
3.3.1.11 l1	10
3.3.1.12 l2	11
3.3.1.13 lambda_file	11

3.3.1.14 ligand	11
3.3.1.15 nu	11
3.3.1.16 offset	11
3.3.1.17 offset_file	11
3.3.1.18 s0	11
3.3.1.19 s1	11
3.3.1.20 s2	12
3.3.1.21 T	12
3.3.1.22 ti	12
3.3.1.23 w1	12
3.3.1.24 w2	12
3.4 Protocol Struct Reference	12
3.4.1 Field Documentation	12
3.4.1.1 ns	13
3.4.1.2 T	13
3.5 pulse Struct Reference	13
3.5.1 Field Documentation	13
3.5.1.1 centre	13
3.5.1.2 type	13
3.5.1.3 width	13
4 File Documentation	15
4.1 couplings/coupling_calc.f08 File Reference	15
4.1.1 Function/Subroutine Documentation	15
4.1.1.1 coupling_calc()	15
4.1.1.2 get_file_length()	15
4.1.1.3 j_calc()	15
4.1.1.4 mu_calc()	16
4.2 lineshape/src/functions.c File Reference	16
4.2.1 Macro Definition Documentation	16
4.2.1.1 M_PI	16
4.2.2 Function Documentation	16
4.2.2.1 At()	17
4.2.2.2 c_n()	17
4.2.2.3 cw_big()	17
4.2.2.4 cw_car()	17
4.2.2.5 cw_chl()	17
4.2.2.6 cw_odo()	17
4.2.2.7 Ft()	18
4.2.2.8 reorg_int()	18
4.2.2.9 trig_im()	18
4.2.2.10 trig_re()	18
4.3 lineshape/src/functions.h File Reference	18

4.3.1 Function Documentation	19
4.3.1.1 At()	19
4.3.1.2 c_n()	19
4.3.1.3 cw_big()	19
4.3.1.4 cw_car()	19
4.3.1.5 cw_chl()	19
4.3.1.6 cw_odo()	20
4.3.1.7 Ft()	20
4.3.1.8 reorg_int()	20
4.3.1.9 trig_im()	20
4.3.1.10 trig_re()	20
4.4 lineshape/src/parameters.c File Reference	20
4.4.1 Function Documentation	21
4.4.1.1 get_parameters()	21
4.4.1.2 get_protocol()	21
4.5 lineshape/src/parameters.h File Reference	21
4.5.1 Macro Definition Documentation	22
4.5.1.1 CMS	22
4.5.2 Function Documentation	22
4.5.2.1 fortran_wrapper()	22
4.5.2.2 get_parameters()	22
4.5.2.3 get_protocol()	22
4.6 lineshape/src/specDens.c File Reference	22
4.6.1 Function Documentation	23
4.6.1.1 main()	23
4.7 spectra/src/fluorescence.c File Reference	23
4.7.1 Function Documentation	23
4.7.1.1 array_to_gsl_matrix()	23
4.7.1.2 bcs()	23
4.7.1.3 jacobian()	24
4.7.1.4 odefunc()	24
4.7.1.5 rate_calc()	24
4.7.1.6 relaxation_rates()	24
4.7.1.7 transfer_matrix()	24
4.7.1.8 trapezoid()	25
4.8 spectra/src/fluorescence.h File Reference	25
4.8.1 Function Documentation	25
4.8.1.1 array_to_gsl_matrix()	25
4.8.1.2 bcs()	26
4.8.1.3 final_matrix()	26
4.8.1.4 jacobian()	26
4.8.1.5 odefunc()	26
4.8.1.6 rate_calc()	26

4.8.1.7 relaxation_rates()	27
4.8.1.8 transfer_matrix()	27
4.8.1.9 trapezoid()	27
4.9 spectra/src/input.c File Reference	27
4.9.1 Function Documentation	27
4.9.1.1 read()	28
4.9.1.2 read_eigvecs()	28
4.9.1.3 read_gi()	28
4.9.1.4 read_input_file()	28
4.9.1.5 read_mu()	29
4.10 spectra/src/input.h File Reference	29
4.10.1 Macro Definition Documentation	30
4.10.1.1 E0	30
4.10.1.2 EC	30
4.10.1.3 JPERINVC	30
4.10.1.4 KCOUL	30
4.10.1.5 MAX_PIGMENT_NUMBER	30
4.10.1.6 TOCM1	30
4.10.1.7 TOFS	30
4.10.2 Function Documentation	31
4.10.2.1 read()	31
4.10.2.2 read_eigvecs()	31
4.10.2.3 read_gi()	31
4.10.2.4 read_input_file()	32
4.10.2.5 read_mu()	32
4.11 spectra/src/spectra.c File Reference	32
4.11.1 Function Documentation	32
4.11.1.1 main()	33
4.12 spectra/src/steady_state.c File Reference	33
4.12.1 Function Documentation	33
4.12.1.1 guess()	33
4.12.1.2 incident()	33
4.12.1.3 pop_converge()	34
4.12.1.4 pop_steady_df()	34
4.12.1.5 pop_steady_f()	34
4.12.1.6 pop_steady_fdf()	34
4.13 spectra/src/steady_state.h File Reference	34
4.13.1 Typedef Documentation	35
4.13.1.1 pulse	35
4.13.1.2 pulse_type	35
4.13.1.3 ss_init	35
4.13.2 Enumeration Type Documentation	35
4.13.2.1 pulse_type	35

4.13.2.2 ss_init	36
4.13.3 Function Documentation	36
4.13.3.1 guess()	36
4.13.3.2 incident()	36
4.13.3.3 pop_converge()	37
4.13.3.4 pop_steady_df()	37
4.13.3.5 pop_steady_f()	37
4.13.3.6 pop_steady_fdf()	37
Index	39

Chapter 1

Data Type Index

1.1 Data Types List

Here are the data types with brief descriptions:

Input	
Input struct for important parameters and filenames where the exciton-basis data is to be read in from	5
ode_params	7
Parameters	8
Protocol	12
pulse	13

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

couplings/coupling_calc.f08	15
lineshape/src/functions.c	16
lineshape/src/functions.h	18
lineshape/src/parameters.c	20
lineshape/src/parameters.h	21
lineshape/src/specDens.c	22
spectra/src/fluorescence.c	23
spectra/src/fluorescence.h	25
spectra/src/input.c	27
spectra/src/input.h	29
spectra/src/spectra.c	32
spectra/src/steady_state.c	33
spectra/src/steady_state.h	34

Chapter 3

Data Type Documentation

3.1 Input Struct Reference

Input struct for important parameters and filenames where the exciton-basis data is to be read in from.

```
#include <input.h>
```

Data Fields

- unsigned int **N**
Number of pigments/excitons.
- double **T**
Temperature.
- unsigned int **tau**
number of steps in $g(t)$ arrays ($\equiv fs$)
- char **eigvecs_file** [200]
- char **eigvals_file** [200]
- char **mu_file** [200]
- char **lambda_file** [200]
- char **gamma_file** [200]
- char **aw_file** [200]
- char **fw_file** [200]
- char **pop_file** [200]
- char * **gi_files** []

3.1.1 Detailed Description

Input struct for important parameters and filenames where the exciton-basis data is to be read in from.

This struct mainly holds a list of filenames which have been output by the fortran code, listing where all the data is for the eigenstates of our Hamiltonian (the exciton basis data). Also holds a few parameters that aren't specific to any particular pigment or exciton.

3.1.2 Field Documentation

3.1.2.1 aw_file

```
char Input::aw_file[200]
```

3.1.2.2 eigvals_file

```
char Input::eigvals_file[200]
```

3.1.2.3 eigvecs_file

```
char Input::eigvecs_file[200]
```

3.1.2.4 fw_file

```
char Input::fw_file[200]
```

3.1.2.5 gamma_file

```
char Input::gamma_file[200]
```

3.1.2.6 gi_files

```
char* Input::gi_files[]
```

3.1.2.7 lambda_file

```
char Input::lambda_file[200]
```

3.1.2.8 mu_file

```
char Input::mu_file[200]
```

3.1.2.9 N

```
unsigned int Input::N
```

Number of pigments/excitons.

3.1.2.10 pop_file

```
char Input::pop_file[200]
```

3.1.2.11 T

```
double Input::T
```

Temperature.

3.1.2.12 tau

```
unsigned int Input::tau
```

number of steps in g(t) arrays (\equiv fs)

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

- [spectra/src/input.h](#)

3.2 ode_params Struct Reference

```
#include <fluorescence.h>
```

Data Fields

- unsigned int [N](#)
- double ** [kij](#)
- double ** [Tij](#)
- double * [rates](#)
- double * [chiw](#)

3.2.1 Field Documentation

3.2.1.1 `chiw`

```
double* ode_params::chiw
```

3.2.1.2 `kij`

```
double** ode_params::kij
```

3.2.1.3 `N`

```
unsigned int ode_params::N
```

3.2.1.4 `rates`

```
double* ode_params::rates
```

3.2.1.5 `Tij`

```
double** ode_params::Tij
```

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

- [spectra/src/fluorescence.h](#)

3.3 Parameters Struct Reference

```
#include <parameters.h>
```


Data Fields

- double `s0`
- double `s1`
- double `s2`
- double `g0`
- double `g1`
- double `g2`
- double `l0`
- double `l1`
- double `l2`
- double `offset`
- double `w1`
- double `w2`
- double `ti`
- double `T`
- double `nu`
- double(* `cw`)(double, void *)
- double(* `cn`)(double, void *)
- double `gsw` [3][48]
- int `ligand`
- char `aw_file` [200]
- char `gt_file` [200]
- char `fw_file` [200]
- char `lambda_file` [200]
- char `offset_file` [200]

3.3.1 Field Documentation

3.3.1.1 `aw_file`

```
char Parameters::aw_file[200]
```

3.3.1.2 `cn`

```
double(* Parameters::cn) (double, void *)
```

3.3.1.3 `cw`

```
double(* Parameters::cw) (double, void *)
```

3.3.1.4 fw_file

```
char Parameters::fw_file[200]
```

3.3.1.5 g0

```
double Parameters::g0
```

3.3.1.6 g1

```
double Parameters::g1
```

3.3.1.7 g2

```
double Parameters::g2
```

3.3.1.8 gsw

```
double Parameters::gsw[3][48]
```

3.3.1.9 gt_file

```
char Parameters::gt_file[200]
```

3.3.1.10 l0

```
double Parameters::l0
```

3.3.1.11 l1

```
double Parameters::l1
```

3.3.1.12 l2

```
double Parameters::l2
```

3.3.1.13 lambda_file

```
char Parameters::lambda_file[200]
```

3.3.1.14 ligand

```
int Parameters::ligand
```

3.3.1.15 nu

```
double Parameters::nu
```

3.3.1.16 offset

```
double Parameters::offset
```

3.3.1.17 offset_file

```
char Parameters::offset_file[200]
```

3.3.1.18 s0

```
double Parameters::s0
```

3.3.1.19 s1

```
double Parameters::s1
```

3.3.1.20 s2

```
double Parameters::s2
```

3.3.1.21 T

```
double Parameters::T
```

3.3.1.22 ti

```
double Parameters::ti
```

3.3.1.23 w1

```
double Parameters::w1
```

3.3.1.24 w2

```
double Parameters::w2
```

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

- [lineshape/src/parameters.h](#)

3.4 Protocol Struct Reference

```
#include <parameters.h>
```

Data Fields

- long unsigned int [ns](#)
- double [T](#)

3.4.1 Field Documentation

3.4.1.1 ns

```
long unsigned int Protocol::ns
```

3.4.1.2 T

```
double Protocol::T
```

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

- [lineshape/src/parameters.h](#)

3.5 pulse Struct Reference

```
#include <steady_state.h>
```

Data Fields

- [pulse_type](#) type
- double [centre](#)
- double [width](#)

3.5.1 Field Documentation

3.5.1.1 centre

```
double pulse::centre
```

3.5.1.2 type

```
pulse\_type pulse::type
```

3.5.1.3 width

```
double pulse::width
```

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

- [spectra/src/steady_state.h](#)

Chapter 4

File Documentation

4.1 couplings/coupling_calc.f08 File Reference

Functions/Subroutines

- program `coupling_calc`
- integer function `get_file_length` (buffer)
- real(sp) function `j_calc` (p1, p2, len1, len2)
- real(sp) function, dimension(3) `mu_calc` (p, len, D)

4.1.1 Function/Subroutine Documentation

4.1.1.1 `coupling_calc()`

```
program coupling_calc
```

4.1.1.2 `get_file_length()`

```
integer function coupling_calc::get_file_length (
    character(len=*), intent(in) buffer )
```

4.1.1.3 `j_calc()`

```
real(sp) function coupling_calc::j_calc (
    real(sp), dimension(4, len1) p1,
    real(sp), dimension(4, len2) p2,
    integer, intent(in) len1,
    integer, intent(in) len2 )
```

4.1.1.4 mu_calc()

```
real(sp) function, dimension(3) coupling_calc::mu_calc (
    real(sp), dimension(4, len) p,
    integer, intent(in) len,
    real(sp), intent(in) D )
```

4.2 lineshape/src/functions.c File Reference

```
#include <math.h>
#include <complex.h>
#include "functions.h"
```

Macros

- `#define M_PI 3.1415926535897932384626433832795L`

Functions

- double `cw_chl` (double w, void *params)
- double `cw_car` (double w, void *params)
- double `cw_odo` (double w, void *params)
- double `cw_big` (double w, void *params)
- double `c_n` (double w, void *params)
- double `trig_re` (double w, void *params)
- double `trig_im` (double w, void *params)
- double `reorg_int` (double w, void *params)
- double complex `At` (double w0, double re, double im, double t, double gamma)
- double complex `Ft` (double w0, double re, double im, double reorg, double t, double gamma)

4.2.1 Macro Definition Documentation

4.2.1.1 M_PI

```
#define M_PI 3.1415926535897932384626433832795L
```

4.2.2 Function Documentation

4.2.2.1 At()

```
double complex At (
    double w0,
    double re,
    double im,
    double t,
    double gamma )
```

4.2.2.2 c_n()

```
double c_n (
    double w,
    void * params )
```

4.2.2.3 cw_big()

```
double cw_big (
    double w,
    void * params )
```

4.2.2.4 cw_car()

```
double cw_car (
    double w,
    void * params )
```

4.2.2.5 cw_chl()

```
double cw_chl (
    double w,
    void * params )
```

4.2.2.6 cw_odo()

```
double cw_odo (
    double w,
    void * params )
```

4.2.2.7 Ft()

```
double complex Ft (
    double w0,
    double re,
    double im,
    double reorg,
    double t,
    double gamma )
```

4.2.2.8 reorg_int()

```
double reorg_int (
    double w,
    void * params )
```

4.2.2.9 trig_im()

```
double trig_im (
    double w,
    void * params )
```

4.2.2.10 trig_re()

```
double trig_re (
    double w,
    void * params )
```

4.3 lineshape/src/functions.h File Reference

```
#include <complex.h>
#include <stdio.h>
#include "parameters.h"
```

Functions

- double [cw_chl](#) (double *w*, void **params*)
- double [cw_car](#) (double *w*, void **params*)
- double [cw_odo](#) (double *w*, void **params*)
- double [cw_big](#) (double *w*, void **params*)
- double [c_n](#) (double *w*, void **params*)
- double [trig_re](#) (double *w*, void **params*)
- double [trig_im](#) (double *w*, void **params*)
- double [reorg_int](#) (double *w*, void **params*)
- double complex [At](#) (double *w0*, double *re*, double *im*, double *t*, double *gamma*)
- double complex [Ft](#) (double *w0*, double *re*, double *im*, double *reorg*, double *t*, double *gamma*)

4.3.1 Function Documentation

4.3.1.1 At()

```
double complex At (
    double w0,
    double re,
    double im,
    double t,
    double gamma )
```

4.3.1.2 c_n()

```
double c_n (
    double w,
    void * params )
```

4.3.1.3 cw_big()

```
double cw_big (
    double w,
    void * params )
```

4.3.1.4 cw_car()

```
double cw_car (
    double w,
    void * params )
```

4.3.1.5 cw_chl()

```
double cw_chl (
    double w,
    void * params )
```

4.3.1.6 cw_odo()

```
double cw_odo (
    double w,
    void * params )
```

4.3.1.7 Ft()

```
double complex Ft (
    double w0,
    double re,
    double im,
    double reorg,
    double t,
    double gamma )
```

4.3.1.8 reorg_int()

```
double reorg_int (
    double w,
    void * params )
```

4.3.1.9 trig_im()

```
double trig_im (
    double w,
    void * params )
```

4.3.1.10 trig_re()

```
double trig_re (
    double w,
    void * params )
```

4.4 lineshape/src/parameters.c File Reference

```
#include "parameters.h"
```

Functions

- Protocol `get_protocol` (char *filename)
- Parameters `get_parameters` (char *filename)

4.4.1 Function Documentation

4.4.1.1 `get_parameters()`

```
Parameters get_parameters (  
    char * filename )
```

4.4.1.2 `get_protocol()`

```
Protocol get_protocol (  
    char * filename )
```

4.5 lineshape/src/parameters.h File Reference

```
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <complex.h>  
#include <math.h>
```

Data Structures

- struct `Protocol`
- struct `Parameters`

Macros

- `#define CMS 299792458` /* speed of light _C_ in _M_etres per _S_econd */

Functions

- Parameters `get_parameters` (char *filename)
- Protocol `get_protocol` (char *filename)
- Parameters * `fortran_wrapper` (int ligand)

4.5.1 Macro Definition Documentation

4.5.1.1 CMS

```
#define CMS 299792458 /* speed of light _C_ in _M_etres per _S_econd */
```

4.5.2 Function Documentation

4.5.2.1 fortran_wrapper()

Parameters `* fortran_wrapper (`
 `int ligand)`

4.5.2.2 get_parameters()

Parameters `get_parameters (`
 `char * filename)`

4.5.2.3 get_protocol()

Protocol `get_protocol (`
 `char * filename)`

4.6 lineshape/src/specDens.c File Reference

```
#include <time.h>
#include <stdio.h>
#include <math.h>
#include <gsl/gsl_integration.h>
#include <gsl/gsl_errno.h>
#include <fftw3.h>
#include "functions.h"
#include "parameters.h"
```

Functions

- int `main` (int argc, char **argv)

4.6.1 Function Documentation

4.6.1.1 main()

```
int main (
    int argc,
    char ** argv )
```

4.7 spectra/src/fluorescence.c File Reference

```
#include "fluorescence.h"
```

Functions

- `gsl_matrix * array_to_gsl_matrix` (unsigned int n1, unsigned int n2, double **mat)
- `double ** rate_calc` (unsigned int N, double **eig, double **wij, `Parameters` *p)
- `double * relaxation_rates` (unsigned int N, double *gamma)
- `double ** transfer_matrix` (unsigned int N, double *relax, double **kij)
- `int jacobian` (double t, const double y[], double *dfdy, double dfdt[], void *params)
- `int odefunc` (double x, const double *y, double *f, void *params)
- `double * bcs` (unsigned const int N, const double *eigvals, const double T)
- `double trapezoid` (double *f, unsigned int n)

4.7.1 Function Documentation

4.7.1.1 array_to_gsl_matrix()

```
gsl_matrix* array_to_gsl_matrix (
    unsigned int n1,
    unsigned int n2,
    double ** mat )
```

4.7.1.2 bcs()

```
double* bcs (
    unsigned const int N,
    const double * eigvals,
    const double T )
```

4.7.1.3 jacobian()

```
int jacobian (
    double t,
    const double y[],
    double * dfdy,
    double dfdt[],
    void * params )
```

4.7.1.4 odefunc()

```
int odefunc (
    double x,
    const double * y,
    double * f,
    void * params )
```

4.7.1.5 rate_calc()

```
double** rate_calc (
    unsigned int N,
    double ** eig,
    double ** wij,
    Parameters * p )
```

4.7.1.6 relaxation_rates()

```
double* relaxation_rates (
    unsigned int N,
    double * gamma )
```

4.7.1.7 transfer_matrix()

```
double** transfer_matrix (
    unsigned int N,
    double * relax,
    double ** kij )
```


4.7.1.8 trapezoid()

```
double trapezoid (
    double * f,
    unsigned int n )
```

4.8 spectra/src/fluorescence.h File Reference

```
#include <gsl/gsl_integration.h>
#include <gsl/gsl_errno.h>
#include <gsl/gsl_matrix.h>
#include <gsl/gsl_odeiv2.h>
#include "input.h"
#include "../lineshape/src/parameters.h"
```

Data Structures

- struct [ode_params](#)

Functions

- [gsl_matrix * array_to_gsl_matrix](#) (unsigned int n1, unsigned int n2, double **mat)
- [double ** rate_calc](#) (unsigned int N, double **eig, double **wij, [Parameters](#) *p)
- [double * relaxation_rates](#) (unsigned int N, double *gamma)
- [double ** transfer_matrix](#) (unsigned int N, double *relax, double **kij)
- [double ** final_matrix](#) (unsigned int N, double *relax, double **Tij)
- [int odefunc](#) (double x, const double *y, double *f, void *params)
- [int jacobian](#) (double t, const double y[], double *dfdy, double dfdt[], void *params)
- [double * bcs](#) (unsigned const int N, const double *eigvals, const double T)
- [double trapezoid](#) (double *f, unsigned int n)

4.8.1 Function Documentation

4.8.1.1 array_to_gsl_matrix()

```
gsl_matrix* array_to_gsl_matrix (
    unsigned int n1,
    unsigned int n2,
    double ** mat )
```

4.8.1.2 bcs()

```
double* bcs (
    unsigned const int N,
    const double * eigvals,
    const double T )
```

4.8.1.3 final_matrix()

```
double** final_matrix (
    unsigned int N,
    double * relax,
    double ** Tij )
```

4.8.1.4 jacobian()

```
int jacobian (
    double t,
    const double y[],
    double * dfdy,
    double dfdt[],
    void * params )
```

4.8.1.5 odefunc()

```
int odefunc (
    double x,
    const double * y,
    double * f,
    void * params )
```

4.8.1.6 rate_calc()

```
double** rate_calc (
    unsigned int N,
    double ** eig,
    double ** wij,
    Parameters * p )
```

4.8.1.7 relaxation_rates()

```
double* relaxation_rates (
    unsigned int N,
    double * gamma )
```

4.8.1.8 transfer_matrix()

```
double** transfer_matrix (
    unsigned int N,
    double * relax,
    double ** kij )
```

4.8.1.9 trapezoid()

```
double trapezoid (
    double * f,
    unsigned int n )
```

4.9 spectra/src/input.c File Reference

```
#include <string.h>
#include "input.h"
```

Functions

- `Input * read_input_file` (char *filename)
Reads in the output from the fortran code, returns `Input` struct.
- `double * read` (char *input_file, unsigned int N)
Reads a file with one double per line and returns it as an array.
- `double ** read_mu` (char *input_file, unsigned int N)
Reads transition dipole moments from file and returns array.
- `double ** read_eigvecs` (char *input_file, unsigned int N)
Reads in $N \times N$ eigenvector matrix and returns as an array.
- `double complex ** read_gi` (char *input_files[], unsigned int N, unsigned int tau)
Read in the set of line-broadening functions, return as 2d array.

4.9.1 Function Documentation

4.9.1.1 read()

```
double* read (
    char * input_file,
    unsigned int N )
```

Reads a file with one double per line and returns it as an array.

Note that this and every other function in here returns a pointer to a malloc'd object, so they all need to be freed somewhere!

4.9.1.2 read_eigvecs()

```
double** read_eigvecs (
    char * input_file,
    unsigned int N )
```

Reads in $N \times N$ eigenvector matrix and returns as an array.

It occurs to me as I write these docs that I could do away with separate functions just by taking two ints n and m and reading in an $n \times m$ array from those. Would only require a check that both are greater than 1.

Other than that same deal, reads $N \times N$ floats, outputs $N \times N$ array.

4.9.1.3 read_gi()

```
double complex** read_gi (
    char * input_files[],
    unsigned int N,
    unsigned int tau )
```

Read in the set of line-broadening functions, return as 2d array.

The $g_i(t)$ functions are the mixed line-broadening functions output by the fortran code, consisting of τ complex elements each.

Honestly it's been a while since I wrote these functions and I can't remember now why this one uses `fscanf` instead of the `fgets` etc. from the other functions - maybe I could edit the others to use `fscanf` instead and save myself some duplication of effort. Either way, it generates $gi[i][j]$, where $i \in 0 \rightarrow N - 1$, $j \in 0 \rightarrow \tau - 1$.

4.9.1.4 read_input_file()

```
Input* read_input_file (
    char * filename )
```

Reads in the output from the fortran code, returns `Input` struct.

This is not very portable and to some extent is duplicating effort made in the `lineshape` code (`lineshape/src/parameters.c`) - there's probably a way of fixing that / a library I could use instead but whatever.

Note that the number of `gi_files` isn't known till runtime because neither is N , the number of pigments; luckily C99 allows us to leave the last member of a struct with variable size and then malloc it later, which is what I do here.

`fgets()` reads in the newline at the end of the line as well; to stop that newline from being passed to `fopen()` later I use the `line[strcspn()] = 0` trick to set the newline to a null char.

`strndup` is the last remaining gnu99 extension I use in the whole repo; I will eventually get around to fixing that but I'm lazy :)

NB: check how to free those malloc'd struct members, I can't remember if there's some fancy stuff you have to do or not.

4.9.1.5 read_mu()

```
double** read_mu (
    char * input_file,
    unsigned int N )
```

Reads transition dipole moments from file and returns array.

This one (and the others below) are extremely hacky - the numbers are hardcoded (19, 20, 22) based on the output format of the fortran code. It disgusts me a little bit every time I remember writing it. Not much going on other than that - reads 3 doubles a line, N lines, returns a pointer to the resulting array.

4.10 spectra/src/input.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <string.h>
#include "../lineshape/src/parameters.h"
#include "../lineshape/src/functions.h"
```

Data Structures

- struct **Input**

Input struct for important parameters and filenames where the exciton-basis data is to be read in from.

Macros

- #define **MAX_PIGMENT_NUMBER** 200
- #define **EC** 1.6022E-19
- #define **E0** 8.854E-12
- #define **JPERINVC** 1.986E-23
- #define **KCOUL** (1. / (4. * **M_PI** * **E0** * 0.5))
- #define **TOFS** (2. * **M_PI** * **CMS** * 100. * 1E-15)
- #define **TOCM1** ((1.295E3 * 8.988E9 * 0.5))

Functions

- **Input** * **read_input_file** (char *filename)
Reads in the output from the fortran code, returns Input struct.
- double * **read** (char *input_file, unsigned int N)
Reads a file with one double per line and returns it as an array.
- double ** **read_mu** (char *input_file, unsigned int N)
Reads transition dipole moments from file and returns array.
- double ** **read_eigvecs** (char *input_file, unsigned int N)
Reads in N x N eigenvector matrix and returns as an array.
- double complex ** **read_gi** (char *input_files[], unsigned int N, unsigned int tau)
Read in the set of line-broadening functions, return as 2d array.

4.10.1 Macro Definition Documentation

4.10.1.1 E0

```
#define E0 8.854E-12
```

4.10.1.2 EC

```
#define EC 1.6022E-19
```

4.10.1.3 JPERINVC

```
#define JPERINVC 1.986E-23
```

4.10.1.4 KCOUL

```
#define KCOUL (1. / (4. * M_PI * E0 * 0.5))
```

4.10.1.5 MAX_PIGMENT_NUMBER

```
#define MAX_PIGMENT_NUMBER 200
```

4.10.1.6 TOCM1

```
#define TOCM1 ((1.295E3 * 8.988E9 * 0.5))
```

4.10.1.7 TOFS

```
#define TOFS (2. * M_PI * CMS * 100. * 1E-15)
```

4.10.2 Function Documentation

4.10.2.1 read()

```
double* read (
    char * input_file,
    unsigned int N )
```

Reads a file with one double per line and returns it as an array.

Note that this and every other function in here returns a pointer to a malloc'd object, so they all need to be freed somewhere!

4.10.2.2 read_eigvecs()

```
double** read_eigvecs (
    char * input_file,
    unsigned int N )
```

Reads in $N \times N$ eigenvector matrix and returns as an array.

It occurs to me as I write these docs that I could do away with separate functions just by taking two ints n and m and reading in an $n \times m$ array from those. Would only require a check that both are greater than 1.

Other than that same deal, reads $N \times N$ floats, outputs $N \times N$ array.

4.10.2.3 read_gi()

```
double complex** read_gi (
    char * input_files[],
    unsigned int N,
    unsigned int tau )
```

Read in the set of line-broadening functions, return as 2d array.

The $g_i(t)$ functions are the mixed line-broadening functions output by the fortran code, consisting of τ complex elements each.

Honestly it's been a while since I wrote these functions and I can't remember now why this one uses `fscanf` instead of the `fgets` etc. from the other functions - maybe I could edit the others to use `fscanf` instead and save myself some duplication of effort. Either way, it generates $gi[i][j]$, where $i \in 0 \rightarrow N - 1$, $j \in 0 \rightarrow \tau - 1$.

4.10.2.4 read_input_file()

```
Input* read_input_file (
    char * filename )
```

Reads in the output from the fortran code, returns `Input` struct.

This is not very portable and to some extent is duplicating effort made in the lineshape code ([lineshape/src/parameters.c](#)) - there's probably a way of fixing that / a library I could use instead but whatever.

Note that the number of `gi_files` isn't known till runtime because neither is `N`, the number of pigments; luckily C99 allows us to leave the last member of a struct with variable size and then malloc it later, which is what I do here.

`fgets()` reads in the newline at the end of the line as well; to stop that newline from being passed to `fopen()` later I use the `line[strlen()] = 0` trick to set the newline to a null char.

`strndup` is the last remaining gnu99 extension I use in the whole repo; I will eventually get around to fixing that but I'm lazy :)

NB: check how to free those malloc'd struct members, I can't remember if there's some fancy stuff you have to do or not.

4.10.2.5 read_mu()

```
double** read_mu (
    char * input_file,
    unsigned int N )
```

Reads transition dipole moments from file and returns array.

This one (and the others below) are extremely hacky - the numbers are hardcoded (19, 20, 22) based on the output format of the fortran code. It disgusts me a little bit every time I remember writing it. Not much going on other than that - reads 3 doubles a line, `N` lines, returns a pointer to the resulting array.

4.11 spectra/src/spectra.c File Reference

```
#include "input.h"
#include "steady_state.h"
#include <fftw3.h>
#include <stdio.h>
#include <gsl/gsl_eigen.h>
```

Functions

- `int main (int argc, char **argv)`

4.11.1 Function Documentation

4.11.1.1 main()

```
int main (
    int argc,
    char ** argv )
```

4.12 spectra/src/steady_state.c File Reference

```
#include "steady_state.h"
```

Functions

- unsigned short int `pop_converge` (double *y, double *yprev, unsigned int N, double thresh)
- gsl_vector * `guess` (const `ss_init` p, const double *boltz, const double *musq, unsigned const int max, unsigned const int N)
- double * `incident` (`pulse` p, unsigned int tau)
- int `pop_steady_f` (const gsl_vector *x, void *params, gsl_vector *f)
- int `pop_steady_df` (const gsl_vector *x, void *params, gsl_matrix *J)
- int `pop_steady_fdf` (const gsl_vector *x, void *params, gsl_vector *f, gsl_matrix *J)

4.12.1 Function Documentation

4.12.1.1 guess()

```
gsl_vector* guess (
    const ss_init p,
    const double * boltz,
    const double * musq,
    unsigned const int max,
    unsigned const int N )
```

4.12.1.2 incident()

```
double* incident (
    pulse p,
    unsigned int tau )
```

4.12.1.3 pop_converge()

```
unsigned short int pop_converge (  
    double * y,  
    double * yprev,  
    unsigned int N,  
    double thresh )
```

4.12.1.4 pop_steady_df()

```
int pop_steady_df (  
    const gsl_vector * x,  
    void * params,  
    gsl_matrix * J )
```

4.12.1.5 pop_steady_f()

```
int pop_steady_f (  
    const gsl_vector * x,  
    void * params,  
    gsl_vector * f )
```

4.12.1.6 pop_steady_fdf()

```
int pop_steady_fdf (  
    const gsl_vector * x,  
    void * params,  
    gsl_vector * f,  
    gsl_matrix * J )
```

4.13 spectra/src/steady_state.h File Reference

```
#include "fluorescence.h"  
#include <gsl/gsl_multirroots.h>  
#include <gsl/gsl_errno.h>  
#include <gsl/gsl_matrix.h>  
#include <gsl/gsl_odeiv2.h>  
#include <gsl/gsl_blas.h>  
#include <gsl/gsl_math.h>
```

Data Structures

- struct [pulse](#)

Typedefs

- typedef enum `pulse_type` `pulse_type`
- typedef struct `pulse` `pulse`
- typedef enum `ss_init` `ss_init`

Enumerations

- enum `pulse_type` { `FLAT` = 0, `LORENTZIAN` = 1, `GAUSSIAN` = 2, `DELTA` = 3 }
- enum `ss_init` { `BOLTZ` = 0, `BOLTZ_MUSQ` = 1, `CONST` = 2, `MAX` = 3 }

Functions

- unsigned short int `pop_converge` (double *y, double *yprev, unsigned int N, double thresh)
- double * `incident` (`pulse` p, unsigned int tau)
- gsl_vector * `guess` (const `ss_init` p, const double *boltz, const double *musq, unsigned const int max, unsigned const int N)
- int `pop_steady_f` (const gsl_vector *x, void *params, gsl_vector *f)
- int `pop_steady_df` (const gsl_vector *x, void *params, gsl_matrix *J)
- int `pop_steady_fdf` (const gsl_vector *x, void *params, gsl_vector *f, gsl_matrix *J)

4.13.1 Typedef Documentation

4.13.1.1 pulse

```
typedef struct pulse pulse
```

4.13.1.2 pulse_type

```
typedef enum pulse_type pulse_type
```

4.13.1.3 ss_init

```
typedef enum ss_init ss_init
```

4.13.2 Enumeration Type Documentation

4.13.2.1 pulse_type

```
enum pulse_type
```

Enumerator

FLAT	
LORENTZIAN	
GAUSSIAN	
DELTA	

4.13.2.2 ss_init

```
enum ss_init
```

Enumerator

BOLTZ	
BOLTZ_MUSQ	
CONST	
MAX	

4.13.3 Function Documentation

4.13.3.1 guess()

```
gsl_vector* guess (
    const ss_init p,
    const double * boltz,
    const double * musq,
    unsigned const int max,
    unsigned const int N )
```

4.13.3.2 incident()

```
double* incident (
    pulse p,
    unsigned int tau )
```

4.13.3.3 pop_converge()

```
unsigned short int pop_converge (  
    double * y,  
    double * yprev,  
    unsigned int N,  
    double thresh )
```

4.13.3.4 pop_steady_df()

```
int pop_steady_df (  
    const gsl_vector * x,  
    void * params,  
    gsl_matrix * J )
```

4.13.3.5 pop_steady_f()

```
int pop_steady_f (  
    const gsl_vector * x,  
    void * params,  
    gsl_vector * f )
```

4.13.3.6 pop_steady_fdf()

```
int pop_steady_fdf (  
    const gsl_vector * x,  
    void * params,  
    gsl_vector * f,  
    gsl_matrix * J )
```


Index

array_to_gsl_matrix
 fluorescence.c, 23
 fluorescence.h, 25

At
 functions.c, 16
 functions.h, 19

aw_file
 Input, 6
 Parameters, 9

bcs
 fluorescence.c, 23
 fluorescence.h, 25

BOLTZ
 steady_state.h, 36

BOLTZ_MUSQ
 steady_state.h, 36

c_n
 functions.c, 17
 functions.h, 19

centre
 pulse, 13

chiw
 ode_params, 8

CMS
 parameters.h, 22

cn
 Parameters, 9

CONST
 steady_state.h, 36

coupling_calc
 coupling_calc.f08, 15

coupling_calc.f08
 coupling_calc, 15
 get_file_length, 15
 j_calc, 15
 mu_calc, 15

couplings/coupling_calc.f08, 15

cw
 Parameters, 9

cw_big
 functions.c, 17
 functions.h, 19

cw_car
 functions.c, 17
 functions.h, 19

cw_chl
 functions.c, 17
 functions.h, 19

cw_odo
 functions.c, 17
 functions.h, 19

DELTA
 steady_state.h, 36

E0
 input.h, 30

EC
 input.h, 30

eigvals_file
 Input, 6

eigvecs_file
 Input, 6

final_matrix
 fluorescence.h, 26

FLAT
 steady_state.h, 36

fluorescence.c
 array_to_gsl_matrix, 23
 bcs, 23
 jacobian, 23
 odefunc, 24
 rate_calc, 24
 relaxation_rates, 24
 transfer_matrix, 24
 trapezoid, 24

fluorescence.h
 array_to_gsl_matrix, 25
 bcs, 25
 final_matrix, 26
 jacobian, 26
 odefunc, 26
 rate_calc, 26
 relaxation_rates, 26
 transfer_matrix, 27
 trapezoid, 27

fortran_wrapper
 parameters.h, 22

Ft
 functions.c, 17
 functions.h, 20

functions.c
 At, 16
 c_n, 17
 cw_big, 17
 cw_car, 17
 cw_chl, 17

- cw_odo, 17
 - Ft, 17
 - M_PI, 16
 - reorg_int, 18
 - trig_im, 18
 - trig_re, 18
- functions.h
 - At, 19
 - c_n, 19
 - cw_big, 19
 - cw_car, 19
 - cw_chl, 19
 - cw_odo, 19
 - Ft, 20
 - reorg_int, 20
 - trig_im, 20
 - trig_re, 20
- fw_file
 - Input, 6
 - Parameters, 9
- g0
 - Parameters, 10
- g1
 - Parameters, 10
- g2
 - Parameters, 10
- gamma_file
 - Input, 6
- GAUSSIAN
 - steady_state.h, 36
- get_file_length
 - coupling_calc.f08, 15
- get_parameters
 - parameters.c, 21
 - parameters.h, 22
- get_protocol
 - parameters.c, 21
 - parameters.h, 22
- gi_files
 - Input, 6
- gsw
 - Parameters, 10
- gt_file
 - Parameters, 10
- guess
 - steady_state.c, 33
 - steady_state.h, 36
- incident
 - steady_state.c, 33
 - steady_state.h, 36
- Input, 5
 - aw_file, 6
 - eigvals_file, 6
 - eigvecs_file, 6
 - fw_file, 6
 - gamma_file, 6
 - gi_files, 6
 - lambda_file, 6
 - mu_file, 6
 - N, 7
 - pop_file, 7
 - T, 7
 - tau, 7
- input.c
 - read, 27
 - read_eigvecs, 28
 - read_gi, 28
 - read_input_file, 28
 - read_mu, 28
- input.h
 - E0, 30
 - EC, 30
 - JPERINVCN, 30
 - KCOUL, 30
 - MAX_PIGMENT_NUMBER, 30
 - read, 31
 - read_eigvecs, 31
 - read_gi, 31
 - read_input_file, 31
 - read_mu, 32
 - TOCM1, 30
 - TOFS, 30
- j_calc
 - coupling_calc.f08, 15
- jacobian
 - fluorescence.c, 23
 - fluorescence.h, 26
- JPERINVCN
 - input.h, 30
- KCOUL
 - input.h, 30
- kij
 - ode_params, 8
- 10
 - Parameters, 10
- 11
 - Parameters, 10
- 12
 - Parameters, 10
- lambda_file
 - Input, 6
 - Parameters, 11
- ligand
 - Parameters, 11
- lineshape/src/functions.c, 16
- lineshape/src/functions.h, 18
- lineshape/src/parameters.c, 20
- lineshape/src/parameters.h, 21
- lineshape/src/specDens.c, 22
- LORENTZIAN
 - steady_state.h, 36
- M_PI

- functions.c, 16
- main
 - specDens.c, 23
 - spectra.c, 32
- MAX
 - steady_state.h, 36
- MAX_PIGMENT_NUMBER
 - input.h, 30
- mu_calc
 - coupling_calc.f08, 15
- mu_file
 - Input, 6
- N
 - Input, 7
 - ode_params, 8
- ns
 - Protocol, 12
- nu
 - Parameters, 11
- ode_params, 7
 - chiw, 8
 - kij, 8
 - N, 8
 - rates, 8
 - Tij, 8
- odefunc
 - fluorescence.c, 24
 - fluorescence.h, 26
- offset
 - Parameters, 11
- offset_file
 - Parameters, 11
- Parameters, 8
 - aw_file, 9
 - cn, 9
 - cw, 9
 - fw_file, 9
 - g0, 10
 - g1, 10
 - g2, 10
 - gsw, 10
 - gt_file, 10
 - l0, 10
 - l1, 10
 - l2, 10
 - lambda_file, 11
 - ligand, 11
 - nu, 11
 - offset, 11
 - offset_file, 11
 - s0, 11
 - s1, 11
 - s2, 11
 - T, 12
 - ti, 12
 - w1, 12
 - w2, 12
- parameters.c
 - get_parameters, 21
 - get_protocol, 21
- parameters.h
 - CMS, 22
 - fortran_wrapper, 22
 - get_parameters, 22
 - get_protocol, 22
- pop_converge
 - steady_state.c, 33
 - steady_state.h, 36
- pop_file
 - Input, 7
- pop_steady_df
 - steady_state.c, 34
 - steady_state.h, 37
- pop_steady_f
 - steady_state.c, 34
 - steady_state.h, 37
- pop_steady_fdf
 - steady_state.c, 34
 - steady_state.h, 37
- Protocol, 12
 - ns, 12
 - T, 13
- pulse, 13
 - centre, 13
 - steady_state.h, 35
 - type, 13
 - width, 13
- pulse_type
 - steady_state.h, 35
- rate_calc
 - fluorescence.c, 24
 - fluorescence.h, 26
- rates
 - ode_params, 8
- read
 - input.c, 27
 - input.h, 31
- read_eigvecs
 - input.c, 28
 - input.h, 31
- read_gi
 - input.c, 28
 - input.h, 31
- read_input_file
 - input.c, 28
 - input.h, 31
- read_mu
 - input.c, 28
 - input.h, 32
- relaxation_rates
 - fluorescence.c, 24
 - fluorescence.h, 26
- reorg_int
 - functions.c, 18

- functions.h, 20
- s0
 - Parameters, 11
- s1
 - Parameters, 11
- s2
 - Parameters, 11
- specDens.c
 - main, 23
- spectra.c
 - main, 32
- spectra/src/fluorescence.c, 23
- spectra/src/fluorescence.h, 25
- spectra/src/input.c, 27
- spectra/src/input.h, 29
- spectra/src/spectra.c, 32
- spectra/src/steady_state.c, 33
- spectra/src/steady_state.h, 34
- ss_init
 - steady_state.h, 35, 36
- steady_state.c
 - guess, 33
 - incident, 33
 - pop_converge, 33
 - pop_steady_df, 34
 - pop_steady_f, 34
 - pop_steady_fdf, 34
- steady_state.h
 - BOLTZ, 36
 - BOLTZ_MUSQ, 36
 - CONST, 36
 - DELTA, 36
 - FLAT, 36
 - GAUSSIAN, 36
 - guess, 36
 - incident, 36
 - LORENTZIAN, 36
 - MAX, 36
 - pop_converge, 36
 - pop_steady_df, 37
 - pop_steady_f, 37
 - pop_steady_fdf, 37
 - pulse, 35
 - pulse_type, 35
 - ss_init, 35, 36
- T
 - Input, 7
 - Parameters, 12
 - Protocol, 13
- tau
 - Input, 7
- ti
 - Parameters, 12
- Tij
 - ode_params, 8
- TOCM1
 - input.h, 30
- TOFS
 - input.h, 30
- transfer_matrix
 - fluorescence.c, 24
 - fluorescence.h, 27
- trapezoid
 - fluorescence.c, 24
 - fluorescence.h, 27
- trig_im
 - functions.c, 18
 - functions.h, 20
- trig_re
 - functions.c, 18
 - functions.h, 20
- type
 - pulse, 13
- w1
 - Parameters, 12
- w2
 - Parameters, 12
- width
 - pulse, 13