

Dichroism Calculations

DichroCalc

Benjamin M. Bulheller

Research Group
Jonathan D. Hirst
University of Nottingham
Nottingham
United Kingdom

Research Group
Shaul Mukamel
University of California
Irvine
USA

Contents

1	Introduction	3
1.1	General Information and Conventions	3
1.2	Files and Folder Structure	4
1.2.1	Folder Structure of Stand-Alone Version	4
1.2.2	Folder Structure of Spectron Version	4
1.2.3	Source Files	5
1.3	Units	5
2	Compilation	6
2.1	Linear Algebra Extension NewMat	6
2.2	Compilation Parameters	7
3	Performing a Calculation	7
3.1	Scripts and Perl Libraries	7
3.2	Parameters and Output Files	8
3.3	Workflow	8
3.4	Input File	9
3.4.1	Creation	9
3.4.2	Syntax	12
3.5	Matrix Method Parameters	14
3.6	Generating the Spectra	17
3.6.1	Adding Bandshapes	17
3.6.2	Plotting	23
4	Course of a Calculation	24
4.1	Initializing the Object	24
4.2	Reading the Input (<code>readinput.cpp</code>)	25
4.3	Fitting the Parameters (<code>fitparameters.cpp</code>)	26

4.4	Setting up the Hamiltonian Matrix (<code>matrix.cpp</code>)	27
4.5	The Dichroism Calculation (<code>dichroism.cpp</code>)	29
5	Object Structure and Variables	30
5.1	Input Data	31
5.2	Parameter Set Data	32
5.3	Processed and Calculated Data	33
5.4	Results	35
5.5	Debugging Output	36
5.6	Error Handling	37
6	Bibliography	39

1 Introduction

1.1 General Information and Conventions

All variables that are global within the DichroCalc object are named with the prefix `DC_` to indicate their global scope. The data read from the input and parameter files are stored in nested data structures instead of multiple variables. The advantage of this is that such structures combine a lot of information in an organized fashion as opposed to data scattered over several variables. A structure can be passed on to routines as a whole and it is also available to functions outside the object. For example, `DC_ParSets` contains all parsed parameter set data, `DC_System` all information about the processed system, and `DC_Results` all results of the calculation.

The data structures and classes used to build them are shown in detail in section 5. For each class `X` that is defined in the class `Dichro` (`dichrocalc.h`), an individual function `OutputXClass` was created that prints the entire variable based on the class to the `.dbg` file. Dedicated output functions are important and generally recommended when defining large structures as it is otherwise not possible to output them quickly, e.g. during debugging. Mind that if any changes are done to the class definitions, these have to be reflected in the respective output function to avoid breaking it. The output routines generally print to the debug file because the structures often contain a large amount of data. Therefore, the `-debug` option is required when such functions are used. Another advantage of using output routines for classes is that it removes the clutter of printing the content from the routines filling and processing the data structures.

The main header file containing all class and function definitions is `dichrocalc.h` with the main class being `class Dichro`. It is also used as central point for including header files, that is all inclusions required by other parts of the program are collected there. Therefore, `dichrocalc.h` is the only header file included by other program files. It is also the interface to the outside and is the only file needed to be included by programs using DichroCalc (apart from the library which has to be linked during compilation).

In general, strings and vectors are used exclusively, disfavoring C-style strings and arrays, respectively. The only exception are variables involved in linear algebra calculations, for which the NewMat types are used and which are only accessed using the respective NewMat functions. The access to array/vector elements starts counting at 0 and, to stay consistent, for NewMat variable types the `.element(row,col)` accessor function is used because it also starts counting at 0, while other access methods in NewMat 11 start at 1. Elements in vectors are accessed using the `.at(row).at(col)` function because of its built-in range check (which the access using `[row][col]` does not feature). In order to avoid errors, there is also no ‘conversion’ done for output to the debug file, that is the first transition on the third group is referred to as *transition 0 of state 0 on group 2*.

The standard type for floating point variables is `double` throughout the program. When `float` was used instead, the results showed too large deviations, and to avoid numerous casting commands, `double` was chosen in general.

The web interface of DichroCalc¹ offers the same possibilities for CD and LD as the command line version (in combination with the `bandshape` script). To perform an orientation search while calculating the LD, the online version has to be used, which will return a PDF report for multiple orientations of the PDB.

1.2 Files and Folder Structure

The folder structure differs slightly between the stand-alone version, which was designed to be fully contained in one directory tree, and the version used in Spectron. The source files are identical, the differences are the makefiles, which expect the required library of NewMat in different locations, and some additional folders with information required by the Spectron makefile.

1.2.1 Folder Structure of Stand-Alone Version

```
dichrocalc/
|— include/      all DichroCalc header files and the compiled library libdichrocalc.a
|— lib/          third-party header files and libraries (currently only NewMat)
|— obj/          the object files compiled from the .cpp files
|— src/          .cpp source code files of DichroCalc
|
|— dichrocalc    the compiled binary for stand-alone use
└─ makefile      the makefile to build the binary and library
```

1.2.2 Folder Structure of Spectron Version

```
dichrocalc/
|— dichrocalc/
|   |— dep/      files declaring the dependencies of each source file
|   |— doc/      the DichroCalc documentation
|   |— include/  all DichroCalc header files
|   |— obj/      the object files compiled from the .cpp files
|   |— src/      .cpp source code files of DichroCalc
|   |
|   └─ makefile  the Spectron-based makefile to build the binary and library
|
|— bin/          output directory for the stand-alone binary
|— include/      third-party header files (NewMat)
|— lib/          the DichroCalc and NewMat libraries
|— newmat/       NewMat source files to compile its library
|
|— configure     prepares the compilation, creates make.top
|— make.extralibs include-file to compile the NewMat library
|— makefile      include-file for Spectron-makefile to build the project
|— make.top.in   include-file template with general compile settings
└─ make.top      created from make.top.in by the configure script
```

1.2.3 Source Files

- **dichrocalc.h**
Main DichroCalc header file, contains all definitions of the object class. This is included in all files of DichroCalc to declare the `Dichro::` class with its “global” variables, class definitions and functions.
- **dichrocalc.cpp**
This is merely a front-end using the DichroCalc routines. The binary of it is the stand-alone version of DichroCalc.
- **iolibrary.cpp** and **iolibrary.h**
A collection of general routines (e.g. for handling strings or printing structures) that are used by all files of DichroCalc.
- **readinginput.cpp**
Routines for reading the input file (`.inp`) with the information on the system, and all parameter files (`.par`) specified in it. The parameter files are read from a directory given as parameter to the program.
The file also holds the class constructor and destructor.
- **fitparameters.cpp**
Holds routines to fit the parameters to the chromophores of the system. For each chromophore a new instance of the respective parameter set is created and the monopoles and dipole moments rotated and translated to be fitted to the chromophore’s coordinates.
- **matrix.cpp**
Routines for setting up the Hamiltonian matrix.
- **dichroism.cpp**
The functions to calculate circular and linear dichroism.

1.3 Units

Input units

- atom and moment coordinates: Ångstrom
- monopole charges: 10^{-10} esu
Electrostatic Unit (esu) = $\text{cm}^{\frac{3}{2}}\text{g}^{\frac{1}{2}}\text{s}^{-1} = 3.356 \cdot 10^{-10} \text{ C (= As)}$
- electric transition dipole moments: Debye
- magnetic transition dipole moments: Bohr Magnetons
- energy: cm^{-1}

2 Compilation

2.1 Linear Algebra Extension NewMat

Directory (stand-alone)	lib/newmat.source
Directory (Spectron)	newmat/newmat11
Version	11 β
Website	http://www.robertnz.net/
Libraries required	—
Libraries created	lib/libnewmat.a

For DichroCalc itself only NewMat is needed for the diagonalization of the Hamiltonian matrix. The source directory of the stand-alone version contains the original tarball, the NewMat 11 documentation and folders with compiled versions for Mac OS (`source.darwin`) and Linux (`source.linux`), respectively. The script `copycompiled` can be used after the files were recompiled, it copies the library for the needed architecture to `lib`.

Several makefiles are provided for different compilers (`nm_*.mak`), if the required compiler is not preconfigured, the utility `genmake` can be downloaded from the website to create a suitable makefile. Under Mac OS and using `g++` the library can be compiled with

```
make -f nm_gnu.mak
```

The final library is `libnewmat.a` and needs to be linked as `newmat` during the compilation of programs using the NewMat functions. If the file `libnewmat.a` is at `./lib`, the command to compile a programs is

```
g++ program.cpp -o program -L./lib -l newmat
```

Mind that there must not be a space after `-L`. The required header files (in `dichrocalc/lib/`) are

- `newmat.h`
- `include.h`
- `myexcept.h`
- `newmatap.h`
- `newmatio.h`

The final release of v10 was tried, however, this suffered from linker errors that could not be resolved. Therefore, v11 β was chosen and worked without problems on both Mac OS and Linux.

The routines to reset the source directory for recompilation are `clean`, `clobber`, and `distclean`, given in increasing purging power. They are defined in `Makefile.in`, which needs to be given explicitly to `make` since the inclusion in the actual makefile is not enough:

```
make -f Makefile.in distclean
```

2.2 Compilation Parameters

A `makefile` is provided to compile the program. It was tested using `gcc/g++` version 4.0.1 under Mac OS 10.5 and version 3.3.1 under SuSE Linux 9.0. The single source files do not need any compiler switches, the crucial step is linking the NewMat library and header files when compiling the final binary. For the stand-alone version the required include commands are

```
{object files} -I./lib -L./lib -I./include -L./include \
               -lnewmat -ldichrocalc -lm
```

CAUTION: Using `g++` on Linux the sequence of the parameters is *crucial*. The object files (`.o`) have to be given before the linker parameters, that is the directories to include and libraries to use. Otherwise the error messages will be about **undefined references** because the library is not linked during compilation. Several pages with error messages such as

```
obj/iolibrary.o(.text+0x7c89): In function 'PrintMatrix(Matrix*)':
: undefined reference to 'Matrix::element(int, int)'
```

are due to linker problems and point to the NewMat library `libnewmat.a`. This also occurs, for example, if the library was built for the wrong platform. In `dichrocalc/lib/newmat.source` (stand-alone) and `newmat/newmat11` (Spectron), the source files can be found to recompile the library. For the stand-alone version, the Linux and Mac OS libraries are precompiled and can be copied using the respective command:

```
./copycompiled linux
./copycompiled darwin
```

3 Performing a Calculation

3.1 Scripts and Perl Libraries

The following Perl scripts are needed and require the listed libraries:

Scripts:	Libraries:	
<code>dcinput</code>	<code>DebugPrint.pm</code>	<code>Error.pm</code>
<code>bandshape</code>	<code>GetBaseName.pm</code>	<code>GetParameters.pm</code>
<code>plotspectrum</code>	<code>ParsePDB.pm</code>	<code>ReadDichroCalc.pm</code>
	<code>ReadSpectrum.pm</code>	<code>VectorMath</code>

Perl scripts search the Perl library path for libraries. The above scripts additionally include the folder `~/bin/perl/lib` and the directory of the actual script location. That is, all scripts can simply be kept in the same directory as the libraries. To store them separately on compute nodes without root access, the `~/bin/perl/lib` directory can be used.

3.2 Parameters and Output Files

The binary (compiled from `dichrocalc.cpp`) is only a front-end for the library with the only purpose of feeding parameters given via the command line into the constructor. Therefore, the same features are available for use via the library (in Spectron) or the binary. The following parameters can be given via the command line (the respective options to construct the object are covered in Section 4.1):

Usage: `dichrocalc [options]`

<code>-i , --input inputfile</code>	filename of the input file to process (mandatory)
<code>-p , --params</code>	directory with the parameter files (*.par)
<code>-v , --verbose</code>	verbose output
<code>-d , --debug</code>	set level of debug output to .dbg file (0-5)
<code>--vec</code>	create .vec file (for absorbance/LD)
<code>--pol</code>	create .pol file (transition polarizations)
<code>--mat</code>	create .mat file (matrix, eigenvectors, eigenvalues)
<code>-h , --help, -?</code>	usage output

If no additional files (such as `.dbg`, `.vec`, etc.) are requested, the only file that is produced by default is the CD line spectrum (`.cdl`). If no parameter directory is given then `~/bin/params` is used by default. At the beginning of the constructor there are several variables that define these defaults:

```
// Global configuration parameters
DC_PrintCdl      = true;           // whether to print the .cdl file
DC_PrintXyzFiles = false;          // files with atom coordinates
DC_ParamsDefault = "/bin/params";  // $HOME is added
```

These settings allow to prevent the creation of the `.cdl` file (e.g. if the library is used via Spectron) and to change the default directory for the parameter files. The switch `DC_PrintXyzFiles` triggers the creation of one file containing the original atom coordinates read from the PDB and the fitted coordinates of the parameter sets. The quickest way to compare the match of both coordinate sets is to run `gnuplot` and issue a command such as

```
splot "coord-group.xyz", "coord-parset.xyz"
```

3.3 Workflow

To calculate spectra, a PDB file of the protein is required for the 3D structure, that is the coordinates of the chromophores. The script `dcinput` creates the input file for DichroCalc (`.inp`) from the `.pdb`. The file contains all parameters for the calculation as well as the structural information of the protein (i.e. which atoms make up the chromophores), including all atom coordinates. DichroCalc requires only this file to run and produces, among several other optional output files, the line spectrum (gas phase spectrum). This is either `.cdl` for CD, or `.vec` for absorbance and LD. The script `bandshape` adds bandshapes to the line spectra and `plotspectrum` is used to create postscript files of the results.

Hence, for using the stand-alone version of DichroCalc for a CD calculation the basic steps are:

```
dcinput file.pdb          (=> file.inp)
dichrocalc -i file.inp    (=> file.cdl and file.vec)
bandshape file            (=> file.cd, file.ld, file.ab)
plotspectrum file.cd      (=> file.cd.ps)
```

For use in Spectron, in addition to the .inp file the respective Spectron input file is required. Please see the Spectron documentation for information on it.

3.4 Input File

3.4.1 Creation

The input file is created from a PDB file using the script `dcinput`. For backbone-only calculations (parameters by Hirst or Woody), the script can be used on its own. For charge-transfer and/or side chain chromophores, the available parameters to use have to be defined in the file `chromophores.dat`, which is covered below.

Usage: `dcinput file1 [file2] [...] [options]`

<code>file</code>	a .pdb file, the extension can be omitted
<code>-v</code>	verbose mode
<code>-o</code>	base name of the output file (only if a single file is processed)
<code>-pdb</code>	to output the renumbered pdb file (.inp.pdb) with the numbers referred to in the inp
<code>-log</code>	to output the log file (.inp.log) with error messages and warnings
<code>-dbg</code>	prints out debug information like the CT chromophore assignments (angles)
<code>-nma</code>	use NMA4FIT2 for the backbone chromophores (default)
<code>-wdy</code>	use NMA99WDY for the backbone chromophores
<code>-urea</code>	use urea chromophores instead of peptide chromophores for the backbone
<code>-p SET</code>	use the given parameter set for the backbone chromophores
<code>-ct</code>	include charge-transfer transitions
<code>-sc</code>	include aromatic side chain transitions of PHE, TYR and TRP
<code>-phe</code>	include phenylalanine side chain group
<code>-tyr</code>	include tyrosine side chain group
<code>-trp</code>	include tryptophan side chain group
<code>-scn</code>	include non-aromatic side chain transitions of ASP, GLU, ASN and GLN
<code>-asp</code>	include the aspartic acid carboxyl group
<code>-glu</code>	include the glutamic acid carboxyl group
<code>-asn</code>	include asparagine side chain peptide group
<code>-gln</code>	include glutamine side chain peptide group
<code>-nb</code>	include parameters for all nucleic bases
<code>-ade</code>	include adenine
<code>-gua</code>	include guanine
<code>-cyt</code>	include cytosine
<code>-thy</code>	include thymine
<code>-ura</code>	include uracil

-nap include the naphthalenediimide chromophore
 -lnk include the naphthalenediimide carboxy group (the linker)
 -trt include the complete trityl group
 -trtsel include specific phenyl rings of the trityl group, e.g. 12 or 13

 -cyc look for peptide bonds between non-successive residues
 (currently LYS-ASP with the atoms CG OD1 NZ supported)
 -term include the terminal carboxyl group
 -nobb do not include the backbone chromophores

 -bbt define the number of backbone transitions (1-4, default is 2)
 -ctt define the number of charge-transfer transitions (1-4, default is 4)
 -sct define the number of side chain transitions (1-4, default is 4)
 -nbt define the number of nucleic bases transitions (1 or 2, default is 2)

 -x x range of the spectrum, default is 150 to 250
 -i interval of the wavelength, default is 1

 -trans x To force a certain number of transitions for each parameter set
 -bbtrans x Choose a particular transition for the backbone (do not confuse
 with -bbt!).
 1 = n-pi*, 2 = pi-pi*, 3 = pinb-pi*, 4 = n'-pi*
 This also sets the number of backbone transitions to 1.
 -cttrans x Choose a particular transition for the charge-transfer transitions.
 1 = n1-pi*2, 2 = n2-pi*1, 3 = pinb1-pi*2, 4 = pinb2-pi*1
 For this you need a cut-down and sorted version of the monopole file!.
 -chrom To define a list of chromophores to be used from chromophores.dat.
 The local chromophore has to be given, too. That way it is possible
 to use only specific CT chromophores without having the same assigned
 to all groups (as it would be the case if only one chromophore was
 given in chromophores.dat).
 -nochrom To define a list of chromophores NOT to use from chromophores.dat.
 They WILL show up in the inp, but will NOT be assigned.

If a chromophores.dat is found in the current directory, it will be read, otherwise
 it is tried to read ~/bin/chromophores.dat instead. To include side chains, charge-
 transfer or nucleic bases, the respective command line parameters MUST be given,
 otherwise the lines are ignored. The local chromophore from chromophores.dat can
 be overridden via -nma, -wdy, etc.

Internally, `dcinput` uses the PDB parser `ParsePDB.pm`¹ to process each given file. To prepare the file for calculation, the following tasks are carried out:

- All hetero atoms (HETATM) and atoms stated as ANISOU, SIGATM and SIGUIJ are removed. Only the peptide bonds and side chains are taken into account during the calculation, hence the unneeded information would just make problems in case the maximum atom number is reached.
- Inserted residues are removed. Often alternative residues are given in PDB files (numbered for example as residues 21 and 21A) to show that the protein exists with two different amino acids at a certain position. Only superimposed residues are affected by this and the inserted one, designated by a letter in the residue number, is removed. If both have a letter (like 21A and 21B), the second one is deleted.
- Alternative atom locations are removed. If some atoms showed a positional deviation during the elucidation via the X-ray diffraction or NMR measurements, these alternative atom locations are sometimes included in the PDB. The parser detects the method used to indicate those alternatives and keeps only the first one.
- All chains, residues and atoms are renumbered to ensure a correct number scheme. The atom number will then match the line number and `dcinput` will write the actual atom number into the `.inp` file, while DichroCalc later on interprets this via the array index as the respective line number in the `$COORDINATES` section. Therefore, a correct numbering is crucial.

The processed PDB file can be written out to `.inp.pdb` by using the `-pdb` option and a detailed log file can be requested with the `-log` option. The logfile first of all lists all problems and warnings of the PDB parser, followed by all issues of `dcinput` itself. All atom assignments are reported, e.g. “4 5 7” for a peptide group, which refers to the atom numbers (in `.inp.pdb`) as well as line numbers (`$COORDINATES`) of the C, O and N atom of the respective peptide bond. This allows to verify all assigned groups.

`dcinput` will at first try to read the file `chromophores.dat` in the current directory. If it is not found there, `~/bin/chromophores.dat` will be read instead. The file looks similar to this:

# Name	Type	Trans	Phi	Psi	
NMA4FIT2	LOC	2	0.0	0.0	# peptide bond
CT01009A	CTR	4	180.0	180.0	# charge-transfer
CTBE009A	CTR	4	-135.0	135.0	# charge-transfer
CT-BE09A	CTR	4	135.0	-135.0	# charge-transfer
CT05009A	CTR	4	-120.0	120.0	# charge-transfer
CT-0509A	CTR	4	120.0	-120.0	# charge-transfer
CT02009B	CTR	4	-120.0	180.0	# charge-transfer
CT-0209B	CTR	4	120.0	180.0	# charge-transfer
CT08009A	CTR	4	-120.0	60.0	# charge-transfer
CT-0809A	CTR	4	120.0	-60.0	# charge-transfer
CT03009A	CTR	4	-60.0	180.0	# charge-transfer
CT-0309A	CTR	4	60.0	180.0	# charge-transfer
CT31009A	CTR	4	-74.0	-4.0	# charge-transfer
CT-3109A	CTR	4	74.0	4.0	# charge-transfer
CTBT009B	CTR	4	-62.0	-41.0	# charge-transfer
CT-BT09B	CTR	4	62.0	41.0	# charge-transfer

CTAL009A	CTR	4	-48.0	-57.0	# charge-transfer
CT-AL09A	CTR	4	48.0	57.0	# charge-transfer
CT15009A	CTR	4	-60.0	-60.0	# charge-transfer
CT-1509A	CTR	4	60.0	60.0	# charge-transfer
PHECASI3	PHE	4	0.0	0.0	# phenylalanine
TYRCASY4	TYR	4	0.0	0.0	# tyrosine
TRPCASH1	TRP	4	0.0	0.0	# tryptophan
ASNHIRST	ASN	4	0.0	0.0	# asparagine
GLNHIRST	GLN	4	0.0	0.0	# glutamine
ASP_CNVE	ASP	2	0.0	0.0	# aspartic acid
GLU_CNVE	GLU	2	0.0	0.0	# glutamic acid

Lines starting with a hash symbol can be used to add comments as shown in the first line. Each other line represents one parameter set from the respective `.par` file. The first column states the name, which is given in the latter file as, for example, `-NMA4FIT2-`. The second column defines the type of the set as local (backbone, LOC), charge-transfer (CTR), or side chain chromophore (PHE, TYR, TRP, etc.). Using the parameters `-nma` (the default), `-wdy` or `-urea` it is possible to override the local chromophore given in `chromophores.dat` and use another frequent one instead.

The third column denotes the number of transitions available for the set. The last two columns define the ϕ and ψ angles for charge-transfer chromophores, and `dcinput` uses those values to determine the best-fitting CT chromophore for any given dipeptide group. Especially for the number of transitions, `dcinput` offers many options to override settings in `chromophores.dat`.

Mind that the `chromophores.dat` contains all parameter sets which *can* be chosen from the available parameter files (the syntax of these is covered in section 3.5). However, only the backbone chromophore is included by default, if it is not overridden by an option such as `-nobb`. To use the charge-transfer or side chain parameters, the switches `-ct` or `-sc` have to be given, respectively. Usually all needed configurations concerning which chromophores to take can be done via the `dcinput` command line. The parameters of `dcinput` allow a complete definition which parameters are to be used or omitted from the `chromophores.dat` (e.g. charge-transfer, side chains and how many transitions) and how many transitions are to be used for each chromophore type. Many of these functions have been implemented to analyze the effect of the parameter sets and their transitions in detail on the SP175 protein set.²

The parameter sets for the calculation are stored in text files with the extension `.par`. They are kept in one directory, whose location is given to DichroCalc by a parameter. Their syntax is covered in the following section.

3.4.2 Syntax

The input file (`.inp`) is formatted in blocks titled with keywords in *capital* letters. `$KEYWORD` marks the beginning of a block, `$END` the last line of it. Empty lines and comments (beginning with `#`) are ignored and may be used to group or annotate the data. For indentation and separation of columns, blanks and tabs can be used.

The `$CONFIGURATION` block holds general information about the protein and some parameters to control DichroCalc. The individual variables of the class that stores the values internally are in the letter case as given below, however, the keywords are compared case-insensitively when the file is parsed.

```

$CONFIGURATION
  BBTrans      -1      To use a specific backbone transition
  CTTrans      -1      To use a specific charge-transfer transition
  Factor        4      Scaling factor for intensity (number of residues)
  MinWL        150     The minimum wavelength to plot
  MaxWL        250     The maximum wavelength to plot
$END

```

The values **BBTrans** and **CTTrans** are used to select a specific backbone or charge-transfer transition. If set to -1 (default) all transitions are used. A value of 0 would use only the first transition, 1 only the second transition, etc. If this feature is used, only a single transition can be included from the respective set, that is it is not possible to include only the 2nd and 3rd transition of a parameter set. Important for the use of **BBTrans** is that the backbone parameter set has to be the first one (index 0). Also note that the transition sequence in the charge-transfer parameter sets differs and the ground state transitions have to be sorted identically before including, for example, only the $\pi_b \rightarrow \pi^*$ transition of all sets.

MinWL, **MaxWL** and **Factor** are only required for the post-processing of the spectra on the web interface of DichroCalc. For the actual calculation they are not needed. If no specific transitions are required, the whole **\$CONFIGURATION** section can be omitted for the use with Spectron or from the command line.

The **\$PARAMETERS** block lists all parameter set names that are to be used in the calculation. The names have to match case-sensitively the respective **.par** filename in the directory with the parameter sets. Each name is followed by the number of transitions to be considered for it. The parameter sets are referred to in the **\$CHROMOPHORES** block by their index in this list and, for convenience, this index is given in last column.

```

$PARAMETERS
#  name      trans  index
  NMA4FIT2   2      # 0      This is a peptide bond
  CT01009A   4      # 1      The planar charge-transfer chromophore
  CTBT009A   4      # 2      The  $\alpha$ -helical charge-transfer chromophore
  CTBE009A   4      # 3      The  $\beta$ -strand charge-transfer chromophore
$END

```

Each chromophore (e.g. peptide bond, charge-transfer group, aromatic side chain, etc.) is represented by a certain number of atoms. To these atoms the parameters are fitted later on. In the **\$CHROMOPHORES** block the type of the group on each line is defined in the first column by the index of the respective parameter set in the **\$PARAMETERS** block (second column in the latter). The following columns specify the atoms defining the chromophore, these numbers are the indices of the atom in the **\$COORDINATES** block. A peptide group, for example, is defined by the *C*, *O*, and *N* atoms.

```

$CHROMOPHORES
#  type      atoms
#  peptide chromophores
  0          2      3      4
  0          6      7      8
  0          10     11     12
#  charge-transfer chromophores
  2          2      3      4      6      7      8
  2          6      7      8      10     11     12
$END

```

In the above block, the first chromophore is of type 0, which points to NMA4FIT2³ in the \$PARAMETERS section, the parameter set of a peptide bond. The chromophore is represented by atoms number 2, 3, and 4 in the \$COORDINATES section below.

The atom coordinates in the \$COORDINATES block are read from the PDB file, that is they are *xyz* coordinates in Ångstrom.

\$COORDINATES

#	x	y	z	#	atom label
-1.322	0.857	35.960	#	0 N	
-1.393	1.972	35.040	#	1 CA	
-0.359	1.932	33.910	#	2 C	
-0.671	2.170	32.740	#	3 O	
0.886	1.532	34.230	#	4 N	
1.994	1.430	33.300	#	5 CA	
1.789	0.408	32.170	#	6 C	
2.064	0.673	31.000	#	7 O	
1.207	-0.760	32.500	#	8 N	
0.931	-1.845	31.580	#	9 CA	
-0.055	-1.494	30.460	#	10 C	
0.158	-1.813	29.290	#	11 O	
-1.118	-0.739	30.800	#	12 N	
-2.153	-0.307	29.880	#	13 CA	
-1.663	0.607	28.750	#	14 C	
-2.017	0.438	27.580	#	15 O	

etc...

\$END

3.5 Matrix Method Parameters

For every chromophore type to be considered in the calculation an individual parameter set is required. Each set needs to be in a separate file with the same name (case-sensitive) and the extension `.par`. DichroCalc takes a directory name as parameter, so usually all available parameter set files are saved in a special folder whose name is passed on to DichroCalc. The available parameters have to be defined in the file `chromophores.dat`, which is read by the `dcinput` script while the input file for DichroCalc is created (see section 3.4.1).

A parameter set for matrix method calculations first of all requires a set of reference coordinates. For NMA4FIT2,³ the parameters of a peptide bond, these are the atoms with the PDB labels C, O, and N. A transition is defined by its transition energy, electric and magnetic transition dipole moment, and a set of monopole charges. The monopoles are placed near or at the atom positions of the parameter set (Figures 1 and 2). Charge-transfer parameters contain two peptide groups next to each other, both with the same monopole configuration as shown in Figure 1. Note that in NMA4FIT2 the hydrogen atom is not actually contained as a reference atom; only the monopoles have been fitted to the position where the hydrogen atom was in the calculations.

The first line of a parameter set file must contain the name of it, identical to the filename. The second states the number of atoms to be read in. Each atom coordinate (Ångstrom) is followed by a weighting factor, a hash symbol (#), and the PDB label of the atom:

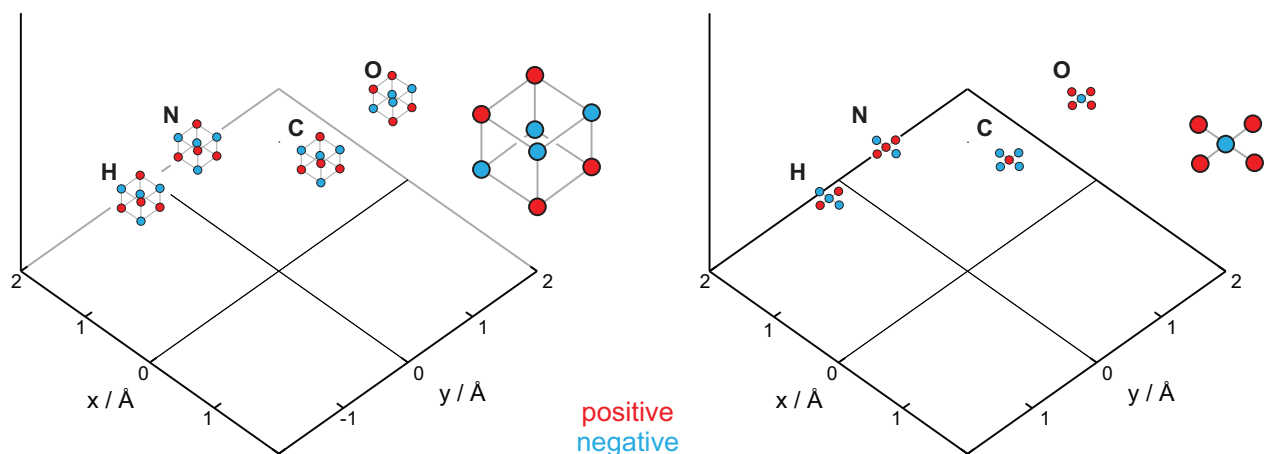


Figure 1: Monopole positions of the *ab initio* amide parameter set, NMA4FIT2,³ for the $n \rightarrow \pi^*$ transition (left panel) and the $\pi \rightarrow \pi^*$ transition (right panel). Negative charges are represented by blue circles and positive charges by red circles. The distance of the monopoles to the atoms is about 0.1 Å in the left panel and either 0 or 0.05 Å in the right panel.

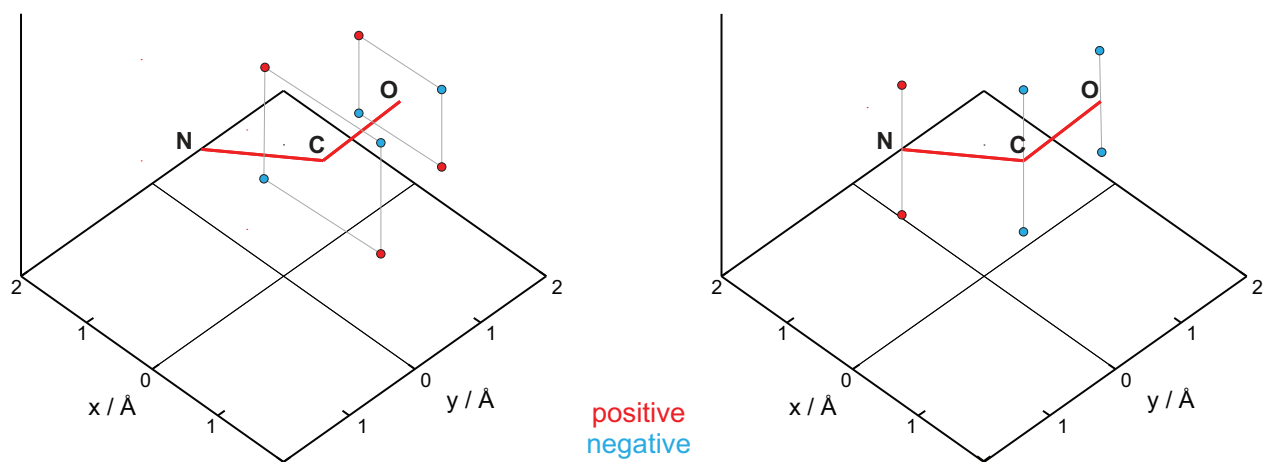


Figure 2: Monopole positions of the semiempirical amide parameter set, NMA99WDY,⁶ for the $n \rightarrow \pi^*$ transition (left panel) and the $\pi \rightarrow \pi^*$ transition (right panel). Negative charges are represented by blue circles and positive charges by red circles. The distance of the monopoles to the atoms is between 0.8 and 1.2 Å.

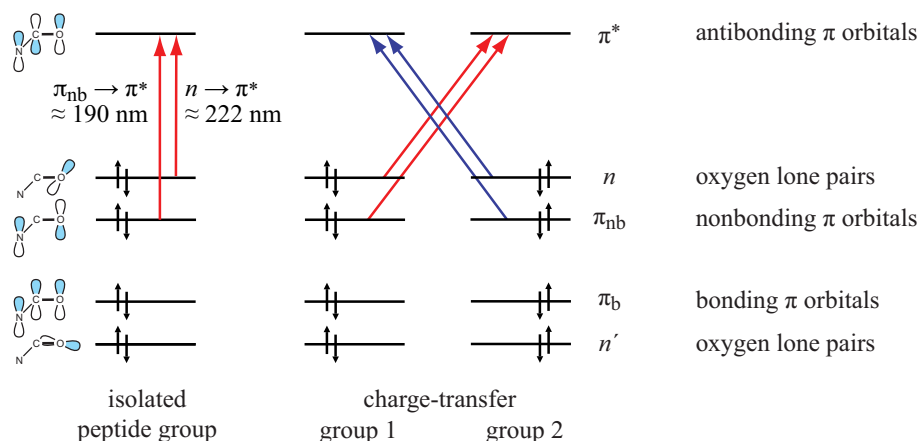


Figure 3: Parametrized transition for the peptide group (left) and charge-transfer between two neighboring peptide groups (right).

```

-NMA4FIT2-
3
1.2270      0.0000      0.0000      1.0000  # C
0.0000      0.0000      0.0000      1.0000  # O
1.9405      1.1917      0.0000      1.0000  # N

```

All transitions from the ground state (state 0) to the excited state of the chromophore follow after the line `&TRANSITION 1->...`. The block labelled `&TRANSITION 2->...` lists interactions of the first transition with the others (one less than in the first block). Figure 4 shows the sequence of the transitions a parameter set.

Each transition block first contains the number of monopoles and the transition energy in wavenumbers. The coordinates of the electric transition dipole moments are given for the moment in Debye. There is also a scale factor, which is a remnant of previous versions and not implemented in DichroCalc. The magnetic transition dipole moment is given in Bohr magnetons, μ_B . Finally, the coordinates and charges of the monopoles follow.

```

32      45455.0
0.0000      0.0000      0.2260      1.000  # Electr. mom., scale-fac
-0.8653     -0.1281      0.0000          # Magnet. mom.
#      x      y      z      q
1.18279936  -0.05639013  0.05291770  -400.30447238
1.18279936  -0.05639013  -0.05291770   400.30447238
1.28839029  -0.04920080  0.05291770   430.73015155
...

```

For protein dichroism calculations the most important parameter set is NMA4FIT2, which was derived from *ab initio* calculations.³ It contains four ground state excitations (Fig. 3):

$$n \rightarrow \pi^*, \quad \pi \rightarrow \pi^*, \quad \pi_b \rightarrow \pi^* \quad \& \quad n' \rightarrow \pi^*.$$

The two transitions that are used by default for the peptide group are the $n \rightarrow \pi^*$ and $\pi \rightarrow \pi^*$ transitions. The four transitions of charge-transfer chromophores.^{2,4,5}

The charge-transfer chromophores (-CT-) each consist of two peptide groups, parametrizing the CT between them. There are 19 CT parameter sets with different conformations (different ϕ/ψ angles between the monomers).^{2,4,5} The first four transitions are four local transitions, two for each monomer:

$$n_1 \rightarrow \pi_1^*, \quad n_2 \rightarrow \pi_2^*, \quad \pi_{nb1} \rightarrow \pi_1^* \quad \& \quad \pi_{nb2} \rightarrow \pi_2^*.$$

Transitions 5–8 are the actual charge-transfer transitions. They are also excitations from the n and π (nonbonding) orbital, but into the π^* orbital of the neighboring peptide group (Figure 3). In the parameter sets, they are not necessarily in the following order!

$$n_1 \rightarrow \pi_2^*, \quad n_2 \rightarrow \pi_1^*, \quad \pi_{nb1} \rightarrow \pi_2^* \quad \& \quad \pi_{nb2} \rightarrow \pi_1^*.$$

Table 2 lists the data of the charge-transfer chromophores. These are required to identify a transition in the parameter set because the order of the transitions is not equal in all sets. By comparing the transition energies given in Table 2 the sequence in the set can be easily determined. Very often, there are already comments in the parameter set, for example

```
64          61659.    # 162.2 nm, Energy 1->6, n2 => Pi*1
```










but if this is not the case, Table 2 is required.

3.6 Generating the Spectra

3.6.1 Adding Bandshapes

For a CD spectrum, the script **bandshape** reads the line spectrum (.cd1), adds Gaussian band shapes and saves the band spectrum to .cd. The plots of an example spectrum before and after the convolution are shown below. The bandwidth 12.5 nm was found to match the experimental spectra best and is used by default.^{8,9} Apart from Gaussian band shapes (default), it can also use Lorentzian and approximate Lorentzian curves. The spectrum is then scaled by a factor which is the number of residues in the protein. If a PDB file with the same base name is found, the number is determined from it, otherwise the scaling factor can be given via the command line option **-s**. This scaling factor is the value stated in the \$CONFIGURATION block of the input file.

Common problems are spectra containing negative wavelengths. This can happen if two residues overlap too much and the Coulomb interaction goes through the roof. There are a couple of cases in CD190⁹ and after lots of attempts the affected residues have been found and removed but without major changes to the spectrum, so generally this problem can be ignored if it is only affecting two or three residues. If negative wavelengths are encountered, **bandshape** stops processing the file and issues an error to check this. With the **-force** option, the spectrum is created nevertheless and **bandshape** issues a warning that negative wavelengths have been found and how many.

State 0	State 1	State 2	State 3
Transition 0			
Transition 1			
Transition 2			
Transition 3			




































State 0	State 1	State 2	State 3	State 4	State 5	State 6	State 7
Local trans. 0							
Local trans. 1							
Local trans. 2							
Local trans. 3							
CT trans. 0							
CT trans. 1							
CT trans. 2							
CT trans. 3							

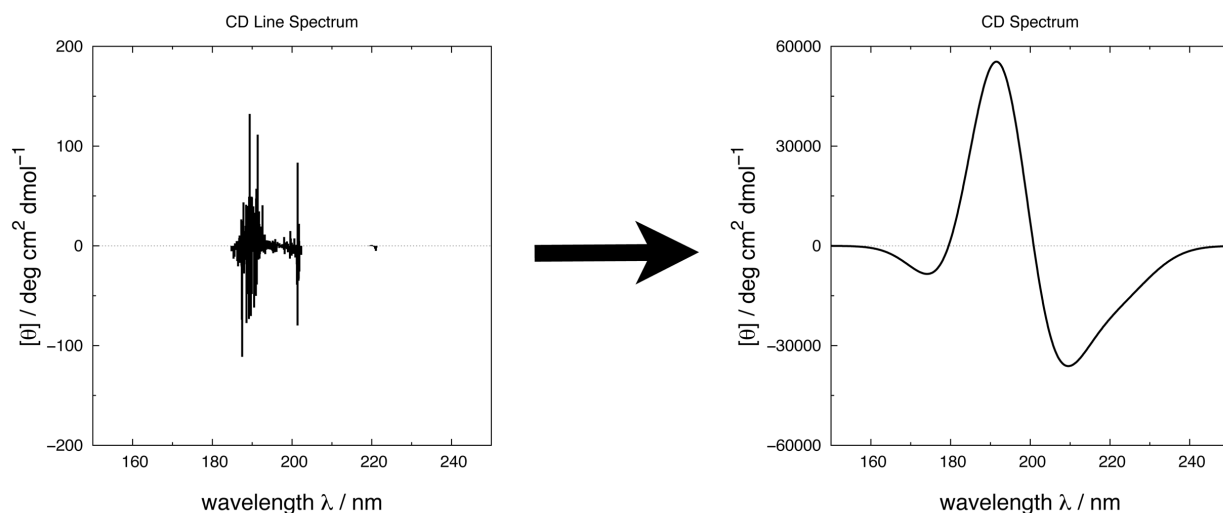
Figure 4: Order of transitions in the states present in a parameter set for a standard chromophore with four transitions (top) and a charge-transfer chromophore (bottom). State 0 contains excitation from the ground state, higher states the interactions between transitions as depicted by the brackets.

Parameter set	Chromophore	Transition	Excitation/State	Wavelength / nm	Energy / cm ⁻¹
NMA4FIT2 ³	peptide bond	0	$n \rightarrow \pi^*$	220.0	45455
		1	$\pi \rightarrow \pi^*$	193.0	51813
		2	$\pi_b \rightarrow \pi^*$	129.4	77263
		3	$n' \rightarrow \pi^*$	122.6	81537
NMA99WDY ⁶	peptide bond	0	$n \rightarrow \pi^*$	220.0	45455
		1	$\pi \rightarrow \pi^*$	190.0	52632
PHECASI3 ⁷	phenylalanine	0	1L_b	263.1	38005
		1	1L_a	208.5	47953
		2	1B_a	191.1	52322
		3	1B_b	190.4	52532
TYRCASY4 ⁷	tyrosine	0	1L_b	274.0	36492
		1	1L_a	216.4	46205
		2	1B_a	196.3	50932
		3	1B_b	192.4	51963
TRPCASH1 ⁷	tryptophan	0	1L_b	282.5	35396
		1	1L_a	262.8	38053
		2	1B_b	204.3	48951
		3	1B_a	196.0	51032
ASNHIRST	arginine	0	$n \rightarrow \pi^*$	220.0	45455
		1	$\pi \rightarrow \pi^*$	193.0	51813
		2	$\pi_b \rightarrow \pi^*$	129.4	77263
		3	$n' \rightarrow \pi^*$	122.6	81537
GLNHIRST	glutamine	0	$n \rightarrow \pi^*$	220.0	45455
		1	$\pi \rightarrow \pi^*$	193.0	51813
		2	$\pi_b \rightarrow \pi^*$	129.4	77263
		3	$n' \rightarrow \pi^*$	122.6	81537

Table 1: Data of matrix method parameter sets for the peptide backbone and side chains.

Table 2: Wavelengths and oscillator strengths (f) of the charge-transfer chromophores. A mirror image of each geometry was created additionally to construct parameters, for example, for the left-handed helical and sheet regions of the Ramachandran plot.^{2,4,5}

$\Phi/^\circ$	$\Psi/^\circ$	$n_1 \rightarrow \pi_2^*$		$n_2 \rightarrow \pi_1^*$		$\pi_{nb1} \rightarrow \pi_2^*$		$\pi_{nb2} \rightarrow \pi_1^*$	
		λ/nm	f	λ/nm	f	λ/nm	f	λ/nm	f
strand structures									
180	180	155.0	0.000	132.6	0.042	171.0	0.040	131.8	0.042
−135	135	154.0	0.001	133.5	0.045	174.4	0.168	122.9	0.006
−120	120	155.7	0.009	135.2	0.111	173.6	0.202	126.4	0.008
hybrid structures									
−120	180	153.2	0.029	142.1	0.056	167.0	0.015	138.5	0.047
−120	60	166.7	0.215	134.9	0.021	159.9	0.056	135.2	0.005
−60	180	156.1	0.004	135.9	0.023	170.9	0.051	136.4	0.058
helical structures									
−74	−4	138.2	0.022	162.9	0.091	153.1	0.002	145.9	0.010
−62	−41	163.5	0.121	159.2	0.068	155.3	0.011	156.1	0.042
−48	−57	156.9	0.123	162.1	0.013	164.4	0.014	152.7	0.015
−60	−60	170.7	0.162	154.5	0.051	163.1	0.020	145.1	0.013



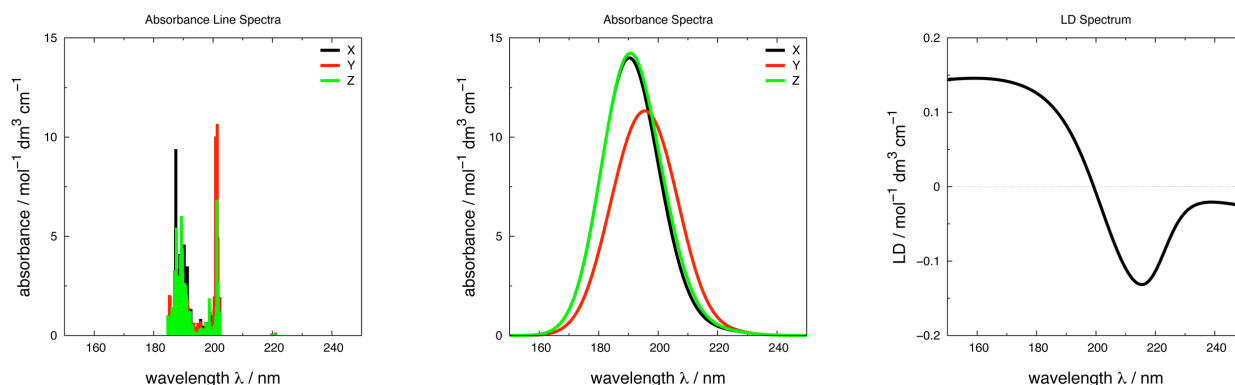
An LD spectrum is generated from a `.vec` file, which lists the polarization vectors (x , y and z direction of the electric transition dipole moment) at each wavelength, for example like this:

220.624106765373568	-0.003161481316737	0.010501954693544	0.013719453780057
220.598672652223826	-0.005832340820421	0.000287929023263	-0.040574242107183
220.570380334759164	-0.004506393382331	0.015418699095573	0.006213239718095
220.537504034016962	-0.007808463093552	0.000615882128750	-0.031612650742340

In the first step, the absorbance line spectra are created by taking the wavelengths and the respective vector component to produce the files named `.x.abl`, `.y.abl` and `.z.abl` (left in the plot below). The line spectra are then convoluted to yield the absorbance spectra `.x.ab`, `.y.ab` and `.z.ab` (middle). z is defined as the parallel direction with respect to the flow direction in a Couette flow cell and, therefore, x and y are perpendicular directions. The actual perpendicular absorbance is considered as a rotational average over x and y :

$$A_{\parallel} = A_z \quad A_{\perp} = \frac{1}{2}(A_x + A_y)$$

Both the parallel and perpendicular absorbance spectra are created, `.para.ab` (the same as `.z.ab` but created for consistency) and `.perp.ab`. From those spectra, the LD spectrum `.ld` (right) is calculated using the chosen method (LD, dichroic ratio or reduced LD).



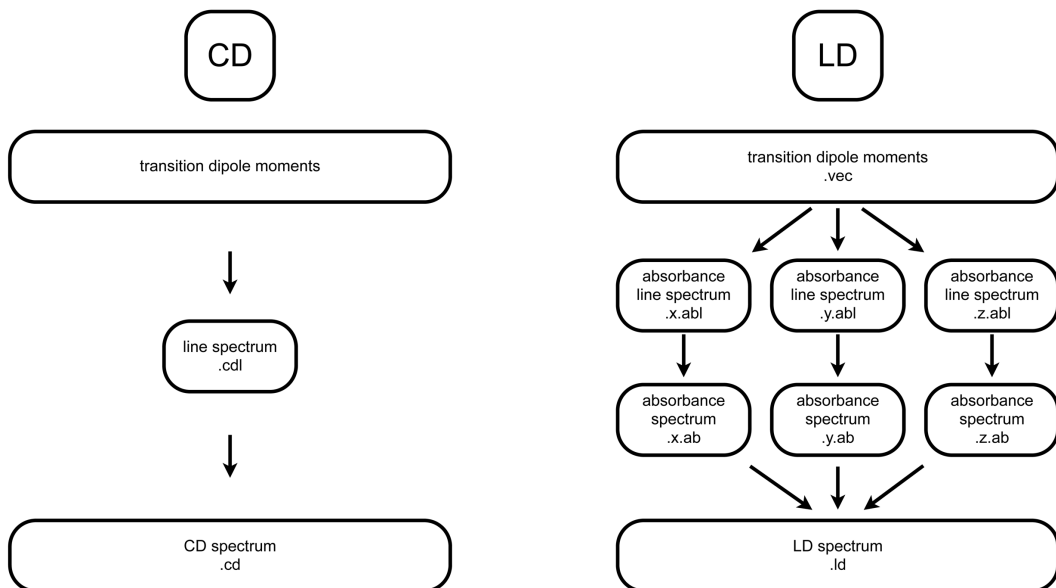


Figure 5: Workflow to create CD, absorbance and LD spectra.

For the calculation of the LD, three different methods can be used. Plain *LD* is the standard, however, the dichroic ratio, *DR*, is a common standard for linear dichroism measurements. The reduced LD, *LD^r*, is obtained by dividing by the absolute absorbance.

$$LD = A_{\parallel} - A_{\perp} = \mu_z^2 - \frac{1}{2}(\mu_x^2 + \mu_y^2)$$

$$LD^r = \frac{A_{\parallel} - A_{\perp}}{A} = \frac{\mu_z^2 - \frac{1}{2}(\mu_x^2 + \mu_y^2)}{\frac{1}{3}(\mu_x^2 + \mu_y^2 + \mu_z^2)}$$

$$DR = \frac{A_{\parallel} - A_{\perp}}{A_{\parallel} + A_{\perp}} = \frac{\mu_z^2 - \frac{1}{2}(\mu_x^2 + \mu_y^2)}{\mu_z^2 + \frac{1}{2}(\mu_x^2 + \mu_y^2)}$$

For consistency, **bandshape** can also handle **.abl** files and convolutes them to create the respective **.ab** spectrum. However, the LD spectrum can only be calculated from the **.vec** file and not from the single absorbance line spectra.

If the **PrintPol** parameter is set true when constructing the object (or the **-pol** option given via the command line), DichroCalc also creates the **.pol** file. This contains much more information than the **.vec**, for example, the polarizations for each single transition type, including the polarization tensors. The algorithm to calculate the LD has been developed for the two default transitions of peptides. Although it has now been extended to “accept” more than two transitions, this should be used with care and possibly checked for validity. The information in the **.pol** file is for sure invalid if charge-transfer chromophores or side chains are included because the groups contain different transition sequences and types. If only peptide bonds are considered, the first transition is always $n \rightarrow \pi^*$, the second always $\pi \rightarrow \pi^*$ and it makes sense to calculate the polarization contributions for all first and all second transitions. However, in charge-transfer and side-chain chromophores the transitions are of a completely different kind.

Figure 5 shows the steps taken to create a CD and an LD spectrum, respectively, starting with the line spectrum files.

3.6.2 Plotting

The `plotspectrum` script is technically a front-end for gnuplot and produces an input file for the latter based on the given command line parameters. It is possible to write out the gnuplot command file as starting point and alter it to suit special needs. All parameters necessary to configure the plot can be given via the command line and the script can also be told to produce a bash script with the used options so that the style of the plot can be changed later or the command line itself be re-used. Due to the vast amount of options for the script, there is an extensive manual for it. In the following, only some particularities for using gnuplot for 3D graphs are covered.

2D spectra are produced by default and, therefore, an absorption spectrum does not require special parameters. Usually the x - and y range are defined a bit different from the ranges chosen by gnuplot:

```
plotspectrum sig-abs -x 42000 60000 -y 0 10000
```

Internally, the `plot` command of gnuplot is used to produce the spectrum. To plot 3D, gnuplot needs to use the `splot` command and this can be triggered by giving the identically named option to the script. Since the output files of Spectron contain many columns of which only a few are to be plotted, gnuplot has to be instructed so. The `-using 4:3:5` option uses columns 4, 3, and 5 for x -, y -, and z -axis. For 3D plots that show surfaces, a PM3D color map improves the clarity of the plot, particularly if a 3D spectrum is viewed down the z -axis. A color surface map can be enabled with the option `-pm3d`.

To be able to generate a PM3D surface, gnuplot requires grid data, that is the input data has to be divided into blocks with the same number of lines and in which one of the three coordinates is constant (whether this is x , y , or z does not matter). This can be done by `plotspectrum` with the `grid` option. In the following examples, the third column is used to create valid gnuplot grid data:

```
plotspectrum sig-kI -x -41900 -59900 -y 41900 59900 \  
-using 4:3:5 -pm3d -contour 10 -grid 3 -splot
```

```
plotspectrum sig-kI -x -41900 -59900 -y 41900 59900 \  
-using 4:3:6 -pm3d -contour 10 -grid 3 -splot
```


4 Course of a Calculation

The routines are executed in a linear fashion during a calculation. At first the input file and matrix method parameters are parsed and the latter fitted to the chromophore coordinates of the protein. With the monopoles and dipole moments in place the Hamiltonian matrix is constructed, diagonalized, and the eigenvalues and eigenvectors used to calculate the dipole moments of the interacting system. These allow the actual dichroism calculation and the line spectrum is printed as result. The following explains the working of DichroCalc in detail.

4.1 Initializing the Object

A program using the `Dichro::` class only needs to include the `dichrocalc.h` header file (apart from being linked with the `libdichrocalc.a` library). The prototype of the object is

```
Dichro::Dichro ( string InFile, string Params, bool Verbose, int Debug,
                bool PrintVec, bool PrintPol, bool PrintMat )
```

The following parameters may or have to be given to the constructor. Only the input file is mandatory, all other parameters are optional.

- **InFile** (mandatory)
The input file (`inp`) with information on the system.
- **Params**
A directory with the files containing the parameter sets. The filenames have to be identical with the individual names given in the input file and a `.par` extension. Only the parameter sets specified in the input file will actually be read in. If this parameter is omitted then `~/bin/params` is used. The default folder can be changed in the constructor.
- **Verbose**
If true, this instructs DichroCalc to output information along the way.
- **Debug**
An integer value between 0 and 5. If greater than 0, DichroCalc writes debug information such as the read input data and intermediate results to the `.dbg` file. The higher this value, the more information is printed. For values 4 and 5, in addition to the debug file the `.fit` file is created with information on the parameter fitting.
- **PrintVec**
Creates the `.vec` file containing the polarization vectors for each transition. The script `bandshape` uses this file to calculate the absorption and LD spectra.
- **PrintPol**
Creates the `.pol` file with the polarizations broken down into the contributions of the single transitions.
- **PrintMat**
Creates the `.mat` file containing the Hamiltonian matrix, eigenvectors, and eigenvalues.

The constructor directly performs the calculation, the results can be retrieved from the object right after creating it. A minimum program using the library may look like this:

```
<-- standard header files -->

#include "dichrocalc.h"

int main ( void )
{
    Dichro *DichroCalc;
    DichroCalc = new Dichro ( "file.inp", "parameters/" );

    cout << setw(15) << DichroCalc->DC_Results.Hamiltonian << "\n\n";
    cout << setw(15) << DichroCalc->DC_Results.Eigenvectors << "\n\n";
    cout << setw(15) << DichroCalc->DC_Results.Eigenvalues << "\n\n";

    return 0;
}
```

The course of the calculation is laid out in the constructor in `readinput.cpp`.

4.2 Reading the Input (`readinput.cpp`)

The Input File: The function `ReadInput` parses the `.inp` file and fills the information into the `DC_Input` structure (Sec. 5.1). The file extension is added if it was omitted. Possible errors may be unknown options in the `$CONFIGURATION` section, or more or less than the expected number of columns in any of the other blocks.

A general syntax check of the input is performed by `CheckInputData`. This includes tests whether the chromophore types used in the `$CHROMOPHORES` block are actually defined under `$PARAMETERS` and if all atoms used for the chromophores are present under `$COORDINATES`.

The parameter sets: If no errors have been found, the parameter sets defined in the input file are read by `ReadParameters`. The function creates a separate structure for each set and collects these in the vector `DC_ParSets` (Sec. 5.2). The sets are read in the same sequence as given in the input file and the index in the `DC_ParSets` vector is, therefore, the index used under `CHROMOPHORES` to assign the parameters to the groups in the protein. Every parameter set is read in completely, regardless of the number of transitions considered according to the input file. This was implemented like this to keep the parsing routine as general as possible.

After reading the atoms, a reference vector for the chromophore is calculated. This is a calculation similar to the center of mass determination, adding up the coordinates of all atoms (taking their individual weighting factors into account). This position vector is required for the fitting of the parameters to the chromophores.

4.3 Fitting the Parameters (fitparameters.cpp)

In `FitParameters`, for each chromophores defined in the `$CHROMOPHORES` section a copy of the respective parameter set is created and its dipole moments and monopoles moved to the position of the chromophore. A Singular Value Decomposition routine (SVD) is used to fit the parameters. If the debug option is greater than 4, all intermediate steps of the fitting process are printed to the `.fit` file.

As preparation for the SVD, two matrices are created from the parameter set atoms and the PDB atoms, with the x,y,z coordinates as columns. The parameter set atoms are located at the origin in the first place. In order to perform the SVD, the atoms of the chromophore are moved to the origin as well and, therefore, a position vector is determined for the group and subtracted from each atom. The two matrices are the input for the SVD that is performed by the function `RotationMatrix`. A rotation matrix is determined that leads to the best fit when it is applied to parameter set coordinates such as the dipole moments and monopoles. The atoms and monopoles are moved to the original position by adding the position vector after rotation.

For a Matrix M , an SVD is defined as

$$M = U \cdot W \cdot V^T \quad .$$

The matrices U , W , and V are determined for the matrix of the parameter set atoms. W is a diagonal matrix, its elements are the singular values. The inverse of W , W^I , is calculated by taking the reciprocals of all elements. If the system is planar the W matrix of the decomposition shows a 0 in one column and the calculation the reciprocal values would, therefore, lead to a division by zero. If this is encountered, an additional point off the plane is added to both matrices by calculating the cross product of the vectors between the first three atoms. With W^I , the pseudo inverse of the parameter set matrix, P_p , can then be calculated as

$$P_p = V \cdot W^I \cdot U^T \quad ,$$

and it follows the non-unitary rotation matrix, R for the matrix of the chromophore group atoms, M_g :

$$R = M_g \cdot P_p^T \quad .$$

A rotation with the non-unitary matrix allows some stretching of the rotated system and this changes the monopole distances and dipole moment lengths (calculations showed a slight worsening of the spectra). Therefore, the determination of the unitary rotation matrix is crucial. A second SVD is performed for the non-unitary rotation matrix, R :

$$R = U \cdot W \cdot V^T$$

and the unitary rotation matrix, R^{uni} , calculated using

$$R^{uni} = U \cdot V^T$$

It is in this routine where the transitions to use in the calculations are selected from the fully parsed parameter set. That is, if two transitions are requested for the `NMA4FIT2` set of the peptide bond in the input file, `FitParameters` will only fit the first two of the available four transitions.

In the `ReadParameters`-function, the parameter set is read in with the transitions divided up into states. The excitations from the ground state are combined in state 0, and interactions between states are combined in higher states (Fig. 4, page 18). The last state contains the permanent moments. The division into separate states would be cumbersome during the interaction calculation and, therefore, the required transitions including the ones in higher states are added one after another to the `DC_System.Trans` vector. Only permanent moments are stored separately in `DC_System.Perm`. To keep track of the origin of the transitions in the `DC_System.Trans` vector, the variable `Origin` stores the parameter set, state and transition it is originating from. This is printed to the debug file, for example:

Transition 4 (Group 0)

Parameter set: NMA4FIT2 - State 1 - Transition 0

This shows that the 5th transition of the first group is the first transition, taken from the second state of the NMA4FIT2 parameter set.

4.4 Setting up the Hamiltonian Matrix (`matrix.cpp`)

The order of the states along the diagonal is exactly as given in the input file. First all transitions of group 0, then all transitions of group 1, and so on and so forth. Mind that the groups are sorted by parameter set type, that is first peptide groups, then charge-transfer groups, and the side chains. This means that the peptide bond, side chain group and charge-transfer contribution of a tyrosine residue will end up separated in the matrix, even though the transitions are localized on the same group. This has no influence on the result.

For example, for a dipeptide and two transitions per group, the important sections in the input file would be

```
$PARAMETERS
#   name      trans.   #   type
   NMA4FIT2     2      #    1
$END

$CHROMOPHORES
#   type      atoms
   1          3      4      5
   1          7      8      9
$END
```

The Hamiltonian matrix constructed for these groups has the form⁹

$$\hat{H} = \begin{pmatrix} E_{1n\pi^*} & V_{1n\pi^*; 1\pi\pi^*} & V_{1n\pi^*; 2n\pi^*} & V_{1n\pi^*; 2\pi\pi^*} \\ V_{1n\pi^*; 1\pi\pi^*} & E_{1\pi\pi^*} & V_{1n\pi^*; 2\pi\pi^*} & V_{1\pi\pi^*; 2\pi\pi^*} \\ V_{1n\pi^*; 2n\pi^*} & V_{1\pi\pi^*; 2n\pi^*} & E_{2n\pi^*} & V_{2n\pi^*; 2\pi\pi^*} \\ V_{1n\pi^*; 2\pi\pi^*} & V_{1\pi\pi^*; 2\pi\pi^*} & V_{2n\pi^*; 2\pi\pi^*} & E_{2\pi\pi^*} \end{pmatrix}. \quad (1)$$

V_{ii} are interactions between states on the same group and V_{ij} are interactions between different groups, respectively. Interactions are determined by calculating the Coulomb-Coulomb interactions

of all monopoles of the involved transitions. For interactions on **different groups** these are only the respective ground state excitations, for example, the $\pi \rightarrow \pi^*$ transitions on groups 0 and 1. For interactions on the **same group** this is the transition of the respective group with the permanent moments on *all* other groups.

If two groups are too close due to a bad fitting of the parameters or a wrong PDB structure, the calculation of the interaction leads to a Coulomb-explosion. This is evident by abnormally-large matrix elements and very high or negative wavelengths. For each group overlap there is usually one or two negative wavelengths at the beginning and the same number of large wavelengths at the end of the line spectrum file (.cdl). The **bandshape** script will flag this up and not create the spectrum in this case. However, it was found that it usually does not affect the spectrum much. If there only a few “bad” transitions, the creation of the band spectrum can be forced using the **-force** option.

The problem of overlapping monopole charges also means that the interactions of two charge-transfer groups sharing one monomer have to be ignored. To catch such cases, all atoms of both groups are compared to check, whether any are shared between the chromophores. The Hamiltonian for a tripeptide, showing only one charge-transfer transition per peptide group for clarity, has the form¹⁰

$$\hat{H} = \begin{pmatrix} E_{Local_1} & V_{E_1; E_2} & V_{E_1; E_3} & V_{E_1; CT_{12}} & V_{E_1; CT_{23}} \\ V_{E_1; E_2} & E_{Local_2} & V_{E_2; E_3} & V_{E_2; CT_{12}} & V_{E_2; CT_{23}} \\ V_{E_1; E_3} & V_{E_2; E_3} & E_{Local_3} & V_{E_3; CT_{12}} & V_{E_3; CT_{23}} \\ V_{E_1; CT_{12}} & V_{E_2; CT_{12}} & V_{E_3; CT_{12}} & CT_{12} & 0 \\ V_{E_1; CT_{23}} & V_{E_2; CT_{23}} & V_{E_3; CT_{23}} & 0 & CT_{23} \end{pmatrix}. \quad (2)$$

Interactions involving charge-transfer groups are **green**. The blocks involving solely local excitations are as defined in Equation 1. The zeros in Equation 2 reflect that charge-transfer chromophores sharing a common peptide group are not allowed to interact. In Equation 2 the first charge-transfer chromophore consists of peptide groups one and two, whereas the latter spans groups two and three.

The matrix is then diagonalized using either the Jacobi or the Householder method, depending on which line of the two is commented. According to the NewMat documentation, the Jacobi method is extremely reliable but much slower than the Householder algorithm.

All results calculated in **HamiltonianMatrix** and the following function, **CD_Calculation**, are collected in the data structure **DC_Results** (see Sec. 5.4, page 35). Notably, the results are accessible twice in the data structure, on a per-transition basis and per-group basis. The former is the way the algorithm works and the interactions are calculated, starting with the first transition of the first group in the first diagonal element. After the diagonalization, the data are copied for each group, including the respective submatrix.

If the object was constructed with the **PrintMat** option then the Hamiltonian matrix, eigenvectors, and eigenvalues are printed to the **.mat** file.

4.5 The Dichroism Calculation (dichroism.cpp)

In the function `CD_Calculation`, first of all the initial magnetic transition dipole moments are *somehow* converted. This involves the electric transition dipole moments, transition energy and the magic number $3.3879 \cdot 10^{-6}$, whose origin has not yet been identified. In the following, the electric and transition dipole moments of the interacting system are calculated, involving the eigenvectors and eigenvalues.

The rotational strength is given by the Rosenfeld equation:

$$R^{0k} = \text{Im}(\langle \psi^0 | \vec{\mu} | \psi^k \rangle \cdot \langle \psi^k | \vec{m} | \psi^0 \rangle) , \quad (3)$$

Therefore, the rotational strength is calculated by multiplying the coordinates of both dipole moments of the transitions. The dipole strength is calculated taking the square of the electric transition dipole moment.

The units of the matrix, eigenvectors and eigenvalues are wavenumbers. The eigenvalues are the transition energies of the interacting system, they are converted to nanometers and printed to the line spectrum file (`.cd1`) with the rotational strength being the *y*-value.

5 Object Structure and Variables

All data read and produced by the program were highly capsulated in nested data structures. This was done in order to make the routines that read the input and parameter files as well as the produced data in general re-usable in other programs. In particular these structures are:

- **DC_Input**
All data read from the input file (`.inp`, the extension may be omitted). Filled by the function `ReadInput` in `readinput.cpp`.
- **DC_ParSets**
Data read from the parameter set files (`.par`). Filled by the function `ReadParameters` in `readinput.cpp`.
- **DC_System**
Combined information from the input and parameter files, that is the complete system with the parameter sets fitted to the protein chromophores. Created by function `FitParameters` in `fitparameters.cpp`.
- **DC_Results**
Calculated results using only the data stored in `DC_System`. Filled by several functions in `matrix.cpp` and `dichroism.cpp`.

The data structures are listed in detail in the following sections. The structures are created by nesting classes using vectors and sub-classes. For each user-defined class there is usually a dedicated output routine that prints it to the debug file. The parent class a particular branch in a tree diagram was created from is given *slanted monospaced* next to the branch.

5.1 Input Data

The data read from the input file (.inp) is collected in the following structure named after the respective section in the input.

DC_Input (<i>Input</i>)	
├─ Configuration (<i>InputConfiguration</i>)	data read from the \$CONFIGURATION block
│ ├─ BBTrans	−1 = all backbone trans., 0–3 = specific BB trans.
│ ├─ CTTrans	−1 = all CT transitions, 0–3 = specific CT trans.
│ ├─ Factor	scale factor for intensities (number of residues)
│ ├─ MinWL	the minimum wavelength
│ └─ MaxWL	the maximum wavelength
├─ Parameters (<i>InputParameters</i>)	data read from the \$PARAMETERS block
│ ├─ Name	names of the parameter sets (also filenames)
│ │ └─ at(<i>Type</i>)	<i>string</i>
│ └─ Trans	number of transitions of the parameter sets
│ │ └─ at(<i>Type</i>)	<i>int</i>
├─ Chromophores (<i>InputChromophores</i>)	data read from the \$CHROMOPHORES block
│ ├─ Type	
│ │ └─ at(<i>Group</i>)	<i>int</i> , index in .Parameters specifying the type
│ └─ Atoms	the indices of the atoms in .Coordinates
│ │ └─ at(<i>Group</i>)	
│ │ │ └─ at(<i>Atom</i>)	<i>int</i>
└─ Coordinates (<i>InputCoordinates</i>)	data read from the \$COORDINATES block
│ ├─ Groups	the <i>xyz</i> coordinates of the atoms
│ │ └─ at(<i>Group</i>)	
│ │ │ └─ at(0 1 2)	<i>double</i>
│ └─ Atoms	atom indices as given in .Chromophores
│ │ └─ at(<i>Group</i>)	<i>int</i>
└─ Labels	the PDB atom labels (e.g. CA)
│ └─ at(<i>Group</i>)	<i>string</i>

Relevant Output Functions

- void Dichro::OutputInputClass (void)
- void Dichro::OutputInputConfigurationClass (void)
- void Dichro::OutputInputParametersClass (void)
- void Dichro::OutputInputChromophoresClass (void)
- void Dichro::OutputInputCoordinatesClass (void)

5.2 Parameter Set Data

For each chromophore (e.g. amide bond, charge-transfer group) a parameter set is read from a file (`.par`). These files are stored in a directory that is given to the program via the `-p/--params` option. All information read from the parameter set files is stored in the following structure.

```

DC_ParSets (ParSet)
├── at(Type) (ParSet)      (the index is the type in .Parameters)
│   ├── Name               string
│   ├── NumberOfAtoms      int
│   ├── NumberOfTransitions int
│   ├── ChargeTransfer      bool
│   ├── Atoms
│   │   ├── at(Atom)
│   │   │   ├── Label      string
│   │   │   ├── Weighting   double
│   │   │   └── Coord.at(0|1|2) double
│   └── Reference.at(0,1,2) double
└── States
    ├── at(State)
    │   ├── at(Trans) (ParSetTrans)
    │   │   ├── Permanent      bool
    │   │   ├── Energy          double (cm-1)
    │   │   ├── Wavelength      double (nm)
    │   │   ├── ScaleFactor     double
    │   │   ├── EDM.at(0|1|2)   double
    │   │   ├── MDM.at(0|1|2)   double
    │   │   ├── NumberOfMonopoles int
    │   │   └── Monopoles
    │   │       ├── at(Mono) (ParSetMonopole)
    │   │       │   ├── Charge    double
    │   │       │   └── Coord.at(0|1|2) double

```

Relevant Output Functions

- `void Dichro::OutputParSetClass (ParSet*)`
- `void Dichro::OutputParSetTransClass (ParSetTrans*, int)`
(the integer is an optional number of the transition)
- `void Dichro::OutputParSetMonopoleClass (ParSetMonopole*)`

5.3 Processed and Calculated Data

DC_System (<i>System</i>)	all information necessary to describe the system
— NumberOfAtoms	<i>int</i> , number of atoms in all parameter sets
— NumberOfGroups	<i>int</i> , number of chromophores
— NumberOfTransitions	<i>int</i> , combined number of transitions
— MatrixDimension	<i>int</i> , same as number of transitions
— Atoms	
— at(<i>Atom</i>)	
— at(0 1 2)	<i>double</i>
— Groups	
— at(<i>Group</i>) (<i>SystemGroup</i>)	
— Atoms	
— at(<i>Atom</i>)	
— at(0 1 2)	<i>double</i>
— AtomIndices	
— at(<i>Atom</i>)	
— at(<i>Index</i>)	<i>int</i>
— Reference	
— at(0 1 2)	<i>double</i> , the reference vector
— NumberOfTransitions	
— ParameterSet	<i>string</i>
— Trans	
— at(<i>Trans</i>) (<i>SystemTransition</i>)	
— Origin	<i>string</i> (parameter set, state & trans.)
— Permanent	<i>bool</i>
— Energy	<i>double</i> (in cm ⁻¹)
— Wavelength	<i>double</i> (in nm)
— NumberOfMonopoles	<i>int</i>
— Monopoles	
— at(<i>Mono</i>) (<i>ParSetMonopole</i>)	
— Charge	<i>double</i>
— Coord.at(0 1 2)	<i>double</i>
— EDM	electric trans. dipole moment
— at(0 1 2)	<i>double</i>
— MDM	magnetic trans. dipole moment
— at(0 1 2)	<i>double</i>
— Perm	
— at(<i>Perm</i>) (<i>SystemTransition</i>)	the permanent moments
:— same structure as <i>Trans</i>	

`DC_System.NumberOfTransitions` is the sum of transitions on all chromophores. This counts only actual transitions, from the ground state to the excited state (e.g. $n \rightarrow \pi^*$ and $\pi \rightarrow \pi^*$), but not higher excitations. It is the same number as `DC_System.MatrixDimension`, it is just stored twice to make some commands in the code easier to read/interpret.

`DC_System.Groups.at(Group).at(Trans).NumberOfTransitions` is determined the same way and holds the number of ground-to-excited state transitions for a particular group.

Relevant Output Functions

- `void Dichro::OutputSystemClass (void)`
- `void Dichro::OutputSystemGroupClass (SystemGroup*, int)`
(the integer is an optional number of the group)
- `void Dichro::OutputSystemTransitionClass (SystemTransition*, int)`
(the integer is an optional number of the transition)

5.4 Results

DC_Results (Results)	all information calculated from DC_System
— NumberOfAtoms	<i>int</i> , number of atoms in all parameter sets
— NumberOfGroups	<i>int</i> , number of chromophores
— NumberOfTransitions	<i>int</i> , combined number of transitions
— MatrixDimension	<i>int</i> , same as number of transitions
— Hamiltonian.element(row,col)	<i>SymmetricMatrix</i>
— Eigenvectors.element(row,col)	<i>Matrix</i>
— Eigenvalues.element(row,row)	<i>DiagonalMatrix</i>
— Groups	
— at(Group) (ResultsGroup)	access via the group
— NumberOfTransitions	<i>int</i>
— ChargeTransfer	<i>bool</i>
— ParameterSet	<i>string</i>
— Energy.at(Trans)	<i>double</i> (in cm ⁻¹)
— Wavelength.at(Trans)	<i>double</i> (in nm)
— DipoleStrength.at(Trans)	<i>double</i>
— RotationalStrength.at(Trans)	<i>double</i>
— OscillatorStrength.at(Trans)	<i>double</i>
— Submatrix.element(row,col)	<i>SymmetricMatrix</i>
— Reference.at(Trans)	Reference vectors of the groups
— at(0 1 2)	<i>double</i>
— EDM.at(Trans)	Electric transition dipole moments
— at(0 1 2)	<i>double</i>
— MDM.at(Trans)	Magnetic transition dipole moments
— at(0 1 2)	<i>double</i>
— PolarizationVector.at(Trans)	Polarization vectors
— at(0 1 2)	<i>double</i>
— Trans (ResultsTrans)	access via the transition number
— GroupSequence.at(Trans)	<i>int</i>
— TransSequence.at(Trans)	<i>int</i>
— ParSetSequence.at(Trans)	<i>string</i>
— Energy.at(Trans)	<i>double</i> (in cm ⁻¹)
— Wavelength.at(Trans)	<i>double</i> (in nm)
— DipoleStrength.at(Trans)	<i>double</i>
— RotationalStrength.at(Trans)	<i>double</i>
— OscillatorStrength.at(Trans)	<i>double</i>
— Reference.at(Trans)	Reference vectors of the groups
— at(0 1 2)	<i>double</i>
— EDM.at(Trans)	Electric transition dipole moments
— at(0 1 2)	<i>double</i>
— MDM.at(Trans)	Magnetic transition dipole moments
— at(0 1 2)	<i>double</i>
— PolarizationVector.at(Trans)	Polarization vectors
— at(0 1 2)	<i>double</i>

Relevant Output Functions

- `void Dichro::OutputResultsClass (void)`
- `void Dichro::OutputResultsGroupClass (void)`
- `void Dichro::OutputResultsTransClass (void)`

NewMat offers two different matrix diagonalization algorithms (section 3.22, “Eigenvalue decomposition” in the NewMat 11 manual). The Jacobi method is extremely reliable but much slower than the second method, the Householder algorithm. In `matrix.cpp` both methods can be selected and it has yet to be tested if one or the other is better suited for the calculations, be it for robustness or speed reasons.

5.5 Debugging Output

- Debug > 0
Data of `DC_Results`
- Debug > 1
Data of `DC_System`
- Debug > 2
 - Fitted groups with rotation matrices
 - Information on the matrix assembly (group interactions)
 - Information on the circular and linear dichroism calculation
- Debug > 3
Data of `DC_Input`
- Debug > 4
Data of `DC_ParSets`
 - The data of all parsed parameter sets are printed. These are the original positions as read from the parameter set files and all transitions available in there (regardless of a smaller transition number used when fitting the parameters to the chromophores based on the input file).
 - All intermediate steps of the parameter fitting to the groups printed (`.fit` file).
 - Hamiltonian matrix, eigenvectors and eigenvalues
 - Monopoles of initial parameter sets and fitted to the chromophores

In the debug file, the names of the variables the data are originating from are used as markers that can be used by scripts reading information from the file. All lines starting with a dollar sign are marker lines for a certain block of information:

\$DC_System.Groups.[0]: Chromophore 0

Parameter set: NMA4FIT2

Atom 0 (Index 2):	-0.3590	1.9320	33.9100
Atom 1 (Index 3):	-0.6710	2.1700	32.7400
Atom 2 (Index 4):	0.8860	1.5320	34.2300

Reference point:	-0.6710	2.1700	32.7400
------------------	---------	--------	---------

Transition 0 (Group 0)

Parameter set: NMA4FIT2 - State 0 - Transition 0

Energy: 45455.00 cm-1

Wavelength: 220.00 nm

Monopoles: 32

EDM:	0.062248	0.215543	0.027246
------	----------	----------	----------

MDM:	-0.344868	0.198079	-0.779092
------	-----------	----------	-----------

Some serious care has been taken to ensure a nice format of the output in the debug file. Just to have it mentioned at least once, even if nobody appreciates it.

In `iolibrary.cpp` there are several functions that are handy during debugging for printing several variable types. `PrintVector` prints vectors of *int*, *double*, *string*, and also vectors of vectors of these types. `PrintMatrix` handles the NewMat types *Matrix*, *Symmetric Matrix*, and *Diagonal Matrix*. For many variable types, including all vectors of vectors handled by `PrintVector`, there is a routine `dp` (*debug print*) whose sole purpose is to print out the given variable and then terminate the program execution. And of course, for every class defined in `dichrocalc.h` an output function is provided in `iolibrary.cpp`. Generally, if you need to print a variable type, there's an `iolibrary` function for that...

5.6 Error Handling

Two variables give information about the exit status of the calculation. `DC_ErrorCode` contains the return code of the last completed function and `DC_Error` the error message, if something went wrong. If `DichroCalc` is used interactively as a library (e.g. within a web interface), it may be important to react depending on the error code.

ReadInput

- 100 Unable to open input file
- 103 Unknown line in input file

ReadInputSection

- 110 Missing \$END in input file
- 112 Unknown option in input file
- 113 Wrong number of columns in \$CONFIGURATION section
- 114 Wrong number of columns in \$PARAMETERS section
- 115 Wrong number of columns in \$CHROMOPHORES section
- 116 Wrong number of columns in \$COORDINATES section

CheckInputData

- 120 Missing parameter in input file
- 123 Missing atoms
- 126 Specific backbone transition selected but number of transitions not 1
- 127 Specific charge-transfer transition selected but number of transitions not 1

ReadParameters

- 130 Error reading directory with parameter files
- 131 Parameter file not found
- 132 Error reading parameter file
- 133 Format error, first line does not match filename of parameter set
- 134 Format error, number of atoms could not be interpreted
- 135 Wrong number of columns in parameter set file
- 137 &TRANSITION label not found in parameter set file

FitParameters

- 140 Number of assigned atoms in parameter set and chromophore does not match
- 143 Matrix dimensions for SVD do not match
- 146 Singular value 0 found

6 Bibliography

The PDFs of cited publications are in the directory `publications` within the documentation's source folder.

References

- [1] B.M. Bulheller and J.D. Hirst, DichroCalc – Circular and Linear Dichroism Online, *Bioinformatics*, **2009**, *25*, 539–540.
- [2] B.M. Bulheller, A.J. Miles, B.A. Wallace, and J.D. Hirst, Charge-Transfer Transitions in the Vacuum-Ultraviolet of Protein Circular Dichroism Spectra, *J. Phys. Chem. B*, **2008**, *112*, 1866–1874.
- [3] N.A. Besley and J.D. Hirst, Theoretical Studies toward Quantitative Protein Circular Dichroism Calculations, *J. Am. Chem. Soc.*, **1999**, *121*, 9636–9644.
- [4] M.T. Oakley and J.D. Hirst, Charge-Transfer Transitions in Protein Circular Dichroism Calculations, *J. Am. Chem. Soc.*, **2006**, *128*, 12414–12415.
- [5] M.T. Oakley, B.M. Bulheller, and J.D. Hirst, First-Principles Calculations of Protein Circular Dichroism in the Far-Ultraviolet and Beyond, *Chirality*, **2006**, *18*, 340–347.
- [6] R.W. Woody and N. Sreerama, Comment on "Improving Protein Circular Dichroism Calculations in the Far-Ultraviolet Through Reparametrizing the Amide Chromophore" [J. Chem. Phys. 109, 782, (1998)], *J. Chem. Phys.*, **1999**, *111*, 2844–2845.
- [7] D.M. Rogers and J.D. Hirst, *Ab Initio* Study of Aromatic Side Chains of Amino Acids in Gas Phase and Solution, *J. Phys. Chem. A*, **2003**, *107*, 11191–11200.
- [8] J.D. Hirst, K. Colella, and A.T.B. Gilbert, Electronic Circular Dichroism of Proteins from First-Principles Calculations, *J. Phys. Chem. B*, **2003**, *107*, 11813–11819.
- [9] B.M. Bulheller, A. Rodger, and J.D. Hirst, Circular and Linear Dichroism of Proteins, *Phys. Chem. Chem. Phys.*, **2007**, *9*, 2020–2035.
- [10] B.M. Bulheller, *Circular and Linear Dichroism of Proteins*, Thesis, University of Nottingham.