Ortlight Documentation

Release 1.2.2

Christian Meisenbichler

May 12, 2015

The Ortlight can analyse various light scattering data. It consists of the `ort` command line tool and the `ortlight` GUI and some additional tools for file conversion and work flow organization.

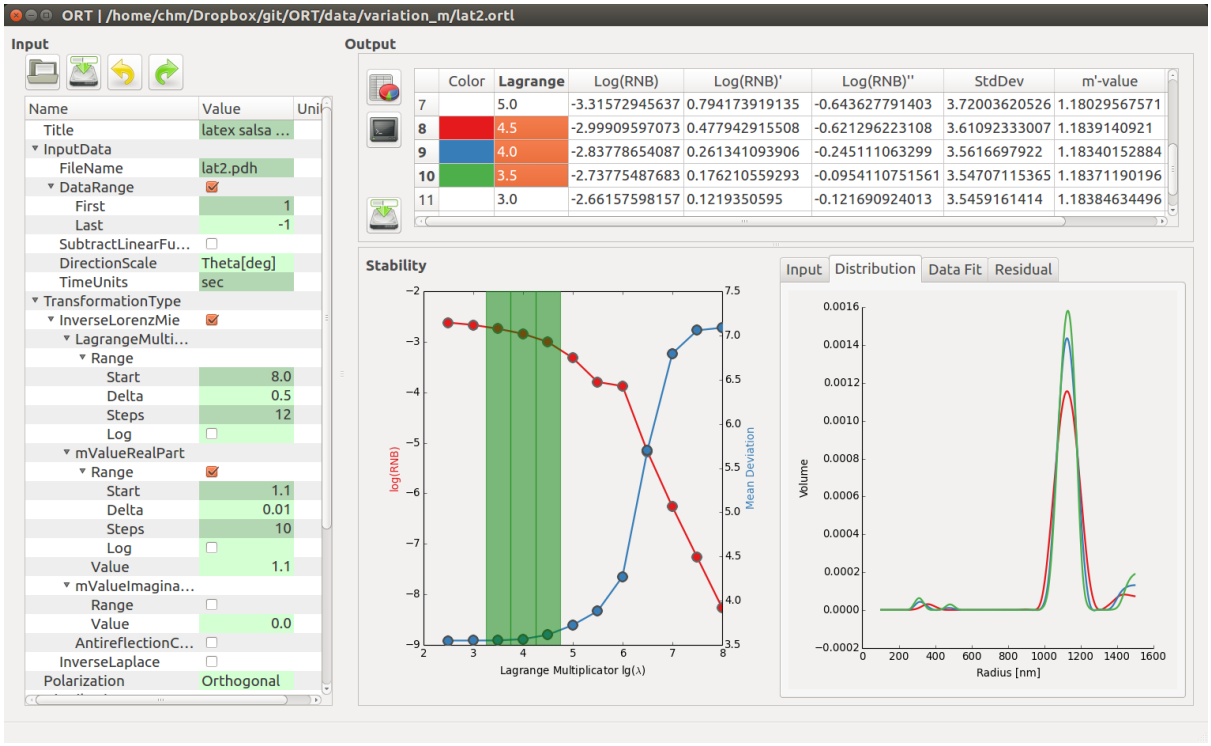

Figure 1: The Ortlight GUI

CHAPTER

ONE

INSTALLATION

1.1 Requirements:

The Ortlight software package is using Python as underlying framework and runtime. In order to run it, you need a Python installation. We currently recommend Anaconda https://store.continuum.io/cshop/anaconda/, the free version. It is working really well and comes with "pip" a tool to manage extension modules.

1.2 Windows Installer

After installing Anaconda Scientific Python, run the Ortlight installer. The installer requires an Internet connection to complete because it needs to install additional packages. It will install the dependencies. the *tools* and the Python module to your system, including Desktop and start menu links and MIME types.

1.3 Download

Get the installer here. Or if you don't have permissions please contact acsoftware@tugraz.at. We will send you a link.

1.4 License, End User Agreement (EULA)

Ortlight ("the Software Product") and accompanying documentation is licensed and not sold. The Graz University of Technology or its affiliates, own intellectual property rights in the Software Product.

1.4.1 Attribution License

Redistribution and derivative use of Ortlight, with or without modification, in source or binary form is explicitly permitted provided that:

You include a copy of this EULA in all copies of the derived software. You cite the original authors in Publications that include Data created by Ortlight or derived software. In derivative work which includes parts of Ortlight in source or binary form you mention that your product either includes or derives from Ortlight. The attribution requirement does not apply for any GPL or LGPL software contained in Ortlight. For citations we suggest the following form.

```
@Misc{OrtLigth2014,
  Title        = {ORTLight Light Scattering Analysis Software},
  Author       = { Meisenbichler C. and
                         Glatter O. and  Chemelli A. and  Uhlig F.},
  HowPublished = {[online] http://ac-software.tugraz.at/ORT},
  Year         = {2014},
}
```

1.4.2 No Implied Warranty or Fitness for Any Use

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS AS IS AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE

1.5 Sourcecode

If you have access the source code can be obtained by cloning from http://ac-software.tugraz.at/git/ort.git:

```
git clone http://ac-software.tugraz.at/git/ort.git
```

CHAPTER

TWO

TOOLS

The ORT software package consists of a collection of tools for light scattering data analysis.

2.1 Ortlight

Ortlight is the central tool, and the only one most users will use. It is a GUI application.

```
$ ortlight  --help
Usage:   ortlight  [input.ortl]

Options:
  -h, --help  show this help message and exit
```

2.2 Ortconvert

Ortconvert is a commandline tool to convert between the legacy ort input file (`.ort`) and the new JSON based ort input file (`.ortl`)

```
$ ortconvert  --help
Usage: This program can convert '.ort' files to '.ortl' and vice versa.
 usage: ortconvert input output

Options:
  -h, --help  show this help message and exit
```

2.3 ORT

This tool is a small python wrapper for the ort.exe fortran kernel. It will check the input file for correct syntax and launch the kernel.

```
$ ort --help
Usage: ort  [input.ortl]

Options:
  -h, --help  show this help message and exit
```

CHAPTER

THREE

INPUTFILE

Ortlight has a all new input file format. It is designed to be readable as it is, without consulting separate documentation, and to be very easily manipulated by other programs and scripts. It is written in JSON which is a very common way to encode data structures in text.

The keys and values allowed, are defined in JSON Schema, a standard for a formal description of the grammar of JSON objects. The documentation of the file format is automatically generated from the schema. The tree view controls in the GUI, the tool tips, the default values and the converter are als configured from the schema.

3.1 Schemadoc

.. _required The '*' signifies a required Field.

The ORT configuration file specifies the parameters for a fit.

Type object

Contains *comment*, *Title*, *InputData**, *TransformationType**, *Polarized*, *Distribution**

Required True

JSON Path

- *#*

Example JSON:

```
{
  "TransformationType": {
    "InverseLorenzMie": {
      "LagrangeMultiplier": {
        "Range": {
          "Start": 6.0,
          "Steps": 10,
          "Delta": 1
        }
      },
      "mValueRealPart": {
        "Value": 1.0
      },
      "mValueImaginaryPart": {
        "Value": 1.0
      }
    }
  },
  "Distribution": {
    "Interval": {
      "Rmin": 100.0,
      "Rmax": 1500.0
    },
    "Splines": {}
```

```
  },
  "InputData": {
    "FileName": "in.pdh"
  }
}
```

3.1.1 Title

Title or description. Free to choose.

Type string

Required False

Default Title

JSON Path

- *#* ['*Title*']

Example JSON:

```
{"Title": "Title"}
```

3.1.2 InputData

Describes the input data, usually scattering data. Time dependent or direction dependent.

Type object

Contains *comment*, *FileName**, *DataRange*, *SubtractLinearFunction*, *DirectionScale*, *TimeUnits*

Required True

JSON Path

- *#* ['*InputData*']

Example JSON:

```
{"InputData": {"FileName": "in.pdh"}}
```

3.1.3 FileName

Path to input data file (PDH).

Type string

Required True

Default in.pdh

JSON Path

- *#* ['*InputData*']['*FileName*']

Old Symbol `TX(1)`

Example JSON:

```
{"FileName": "in.pdh"}
```

3.1.4 DataRange

Give indices for *First* and *Last* Point to use from the scattering data.

Type object

Contains *comment*, *First**, *Last**

Required False

JSON Path

- *#* ['*InputData*']['*DataRange*']

Example JSON:

```
{"DataRange": {"Last": -1,"First": 0}}
```

3.1.5 First

First data point to take from data file

Type integer

Required True

Default 0

JSON Path

- *#* ['*InputData*']['*DataRange*']['*First*']

Old Symbol `IC(13)`

Example JSON:

```
{"First": 0}
```

3.1.6 Last

Last data point to take from data file. The very last is '-1'

Type integer

Required True

Default -1

JSON Path

- *#* ['*InputData*']['*DataRange*']['*Last*']

Old Symbol `IC(14)`

Example JSON:

```
{"Last": -1}
```

3.1.7 SubtractLinearFunction

Subtract a linear function, $y = a_0 + a_1 x$ from the input data

Type object

Contains *comment*, *a0**, *a1**

Required False

JSON Path

- *#* ['*InputData*']['*SubtractLinearFunction*']

Example JSON:

```
{"SubtractLinearFunction": {"a1": 0,"a0": 0}}
```

3.1.8 a0

a0

Type number

Required True

Default 0

JSON Path

- *#* ['*InputData*']['*SubtractLinearFunction*']['*a0*']

Old Symbol `RC(19)`

Example JSON:

```
{"a0": 0}
```

3.1.9 a1

a1

Type number

Required True

Default 0

JSON Path

- *#* ['*InputData*']['*SubtractLinearFunction*']['*a1*']

Old Symbol `RC(20)`

Example JSON:

```
{"a1": 0}
```

3.1.10 DirectionScale

Chose if scattering data uses an angle or the wavelength dependent scattering vector as scale.

Type string

values `[u'Theta[deg]', u'q[1/nm]']`

Required False

Default q[1/nm]

JSON Path

- *#* ['*InputData*']['*DirectionScale*']

Old Symbol `IC(20)`

Example JSON:

```
{"DirectionScale": "Theta[deg]"}
```

3.1.11 TimeUnits

Units for time values

Type string

values [u'sec', u'msec', u'nsec']

Required False

Default sec

JSON Path

- *#* ['*InputData*']['*TimeUnits*']

Example JSON:

```
{"TimeUnits": "sec"}
```

3.1.12 TransformationType

Select the kind of scattering data for analysis. Choose one of the possible entries that may be contained here.

Type object

Contains *comment*, *InverseLorenzMie*, *InverseLaplace*

Required True

Default InverseLorenzMie

JSON Path

- *#* ['*TransformationType*']

Old Symbol IC(4)

Example JSON:

```
{
  "TransformationType": {
    "InverseLorenzMie": {
      "LagrangeMultiplier": {
        "Range": {
          "Start": 6.0,
          "Steps": 10,
          "Delta": 1
        }
      },
      "mValueRealPart": {
        "Value": 1.0
      },
      "mValueImaginaryPart": {
        "Value": 1.0
      }
    }
  }
}
```

3.1.13 InverseLorenzMie

Use the Lorenz Mie theory for the inverse transformation of the static light scattering data (SLS).

Type object

Contains *comment*, *LagrangeMultiplier**, *mValueRealPart**, *mValueImaginaryPart**, *Antireflection-Correction*

Required False

JSON Path

- *#* ['*TransformationType*']['*InverseLorenzMie*']

Old Symbol `IC(4)`

Example JSON:

```
{
  "InverseLorenzMie": {
    "LagrangeMultiplier": {
      "Range": {
        "Start": 6.0,
        "Steps": 10,
        "Delta": 1
      }
    },
    "mValueRealPart": {
      "Value": 1.0
    },
    "mValueImaginaryPart": {
      "Value": 1.0
    }
  }
}
```

3.1.14 mValueRealPart

Range or value of the m', Relative refractive index (real part)

Type object

Contains *comment*, *Range*, *Value**

Required True

JSON Path

- *#* ['*TransformationType*']['*InverseLorenzMie*']['*mValueRealPart*']

Example JSON:

```
{"mValueRealPart": {"Value": 1.0}}
```

3.1.15 mValueImaginaryPart

Range or value of the m'', Relative refractive index (imaginary part).

Type object

Contains *comment*, *Range*, *Value**

Required True

JSON Path

- *#* ['*TransformationType*']['*InverseLorenzMie*']['*mValueImaginaryPart*']

Example JSON:

```
{"mValueImaginaryPart": {"Value": 1.0}}
```

3.1.16 AntireflectionCorrection

Corrects for reflections in round containers.

Type object

Contains *ReflectionParameter**, *CellRefractiveIndex**

Required False

JSON Path

- *#* ['*TransformationType*']['*InverseLorenzMie*']['*AntireflectionCorrection*']

Old Symbol `IC(16)`

Example JSON:

```
{
  "AntireflectionCorrection": {
    "CellRefractiveIndex": 0,
    "ReflectionParameter": {
      "Value": 1.0
    }
  }
}
```

3.1.17 ReflectionParameter

Parameter c_r, to compensate for the light-scattering cell

Type object

Contains *comment*, *Range*, *Value**

Required True

JSON Path

- *#* ['*TransformationType*']['*InverseLorenzMie*']['*AntireflectionCorrection*']['*ReflectionParameter*']

Example JSON:

```
{"ReflectionParameter": {"Value": 1.0}}
```

3.1.18 CellRefractiveIndex

Refractive index of light scattering cell

Type number

Required True

Default 0

JSON Path

- *#* ['*TransformationType*']['*InverseLorenzMie*']['*AntireflectionCorrection*']['*CellRefractiveIndex*']

Old Symbol `RC(11)`

Example JSON:

```
{"CellRefractiveIndex": 0}
```

3.1.19 InverseLaplace

Fit dynamic light scattering data (DLS).

Type object

Contains *comment*, *ScatteringModel*, *LagrangeMultiplier**, *Baseline**

Required False

JSON Path

- *#* ['*TransformationType*']['*InverseLaplace*']

Old Symbol `IC(4)`

Example JSON:

```
{
  "InverseLaplace": {
    "LagrangeMultiplier": {
      "Range": {
        "Start": 6.0,
        "Steps": 10,
        "Delta": 1
      }
    },
    "Baseline": {
      "Value": 1.0
    }
  }
}
```

3.1.20 ScatteringModel

Choose the model for calculating a direction dependent correction for the sensor position.

Type string

values `[u'default', u'RDG', u'Lorenz Mie']`

Required False

Default default

JSON Path

- *#* ['*TransformationType*']['*InverseLaplace*']['*ScatteringModel*']

Old Symbol `IC(5)`

Example JSON:

```
{"ScatteringModel": "default"}
```

3.1.21 Baseline

Baseline of the intensity time-autocorrelation function $g_2(t) = B + \beta g_1(t)^2$.

Type object

Contains *comment*, *Range*, *Value**, *FromFile*

Required True

JSON Path

- *#* ['*TransformationType*']['*InverseLaplace*']['*Baseline*']

Old Symbol `RC(5),RC(6)`

Example JSON:

```
{"Baseline": {"Value": 1.0}}
```

3.1.22 FromFile

Load baseline parameter from file

Type boolean

Required False

Default False

JSON Path

- *#* ['*TransformationType*']['*InverseLaplace*']['*Baseline*']['*FromFile*']

Old Symbol `RC(5)`

Example JSON:

```
{"FromFile": false}
```

3.1.23 Polarized

'true' if light source is polarized

Type boolean

Required False

Default True

JSON Path

- *#* ['*Polarized*']

Old Symbol `RC(18)`

Example JSON:

```
{"Polarized": true}
```

3.1.24 Distribution

This block specifies the representation and domain of the distribution function.

Type object

Contains *comment*, *Type*, *Interval**, *Splines**

Required True

JSON Path

- *#* ['*Distribution*']

Example JSON:

```
{
  "Distribution": {
    "Interval": {
      "Rmin": 100.0,
      "Rmax": 1500.0
    },
    "Splines": {}
  }
}
```

3.1.25 Type

The kind of Distribution function to calculate

Type string

values `[u'Number', u'Volume', u'Intensity']`

Required False

Default Volume

JSON Path

- *#* ['*Distribution*']['*Type*']

Old Symbol `IC(6)`

Example JSON:

```
{"Type": "Number"}
```

3.1.26 Interval

Minimum and maximum particle size.

Type object

Contains *comment*, *Rmin**, *Rmax**

Required True

JSON Path

- *#* ['*Distribution*']['*Interval*']

Example JSON:

```
{"Interval": {"Rmin": 100.0,"Rmax": 1500.0}}
```

3.1.27 Rmin

Minimum radius.

Type number in nm

Required True

Default 100.0

Constraints min:0

JSON Path

- *#* ['*Distribution*']['*Interval*']['*Rmin*']

Old Symbol `RC(3)`

Example JSON:

```
{"Rmin": 100.0}
```

3.1.28 Rmax

Maximum radius.

Type number in nm

Required True

Default 1500.0

Constraints min:0

JSON Path

- *#* ['*Distribution*']['*Interval*']['*Rmax*']

Old Symbol `RC(4)`

Example JSON:

```
{"Rmax": 1500.0}
```

3.1.29 Splines

Number and spacing of the spline elements.

Type object

Contains *comment*, *Middle*, *Left*, *Right*, *Spacing*, *NonNegativityConstraint*

Required True

JSON Path

- *#* ['*Distribution*']['*Splines*']

Example JSON:

```
{"Splines": {}}
```

3.1.30 Middle

Number of splines inside domain.

Type integer

Required False

Default 30

JSON Path

- *#* ['*Distribution*']['*Splines*']['*Middle*']

Old Symbol `IC(8)`

Example JSON:

```
{"Middle": 30}
```

3.1.31 Left

Up to three left splines. The middle splines cover the whole domain but force the solution to be 0 on the boundaries. You can add up to 3 splines on each side of the window to over come this sometimes unwanted constraint.

Type integer

Required False

Default 0

Constraints min:0,max:3

JSON Path

- *#* ['*Distribution*']['*Splines*']['*Left*']

Old Symbol `IC(9)`

Example JSON:

```
{"Left": 0}
```

3.1.32 Right

Up to three right splines. Same as Left splines, just at the upper boundary

Type integer

Required False

Default 0

Constraints min:0,max:3

JSON Path

- *#* ['*Distribution*']['*Splines*']['*Right*']

Old Symbol `IC(10)`

Example JSON:

```
{"Right": 0}
```

3.1.33 Spacing

Choose the spacing of the splines.

Type string

values `[u'linear(R)', u'log10(R)', u'sqrt(R)']`

Required False

Default linear(R)

JSON Path

- *#* ['*Distribution*']['*Splines*']['*Spacing*']

Old Symbol `IC(12)`

Example JSON:

```
{"Spacing": "linear(R)"}
```

3.1.34 NonNegativityConstraint

Constrain the distribution to positive values

Type boolean

Required False

Default True

JSON Path

- *#* ['*Distribution*']['*Splines*']['*NonNegativityConstraint*']

Old Symbol `IC(19)`

Example JSON:

```
{"NonNegativityConstraint": true}
```

3.1.35 Range

Give a Range for variation

Type object

Contains *comment*, *Start**, *Delta**, *Steps**, *Log*

Required True

JSON Path

- *//*['*Range*']
- */*['*TransformationType*']['*InverseLorenzMie*']['*mValueRealPart*']['*Range*']
- */*['*TransformationType*']['*InverseLorenzMie*']['*mValueImaginaryPart*']['*Range*']
- */*['*TransformationType*']['*InverseLorenzMie*']['*AntireflectionCorrection*']['*ReflectionParameter*']['*Range*']
- */*['*TransformationType*']['*InverseLaplace*']['*Baseline*']['*Range*']
- *//*['*LagrangeMultiplier*']['*Range*']

Example JSON:

```
{"Range": {"Start": 6.0,"Steps": 10,"Delta": 1}}
```

3.1.36 Start

Start Value

Type number

Required True

Default 6.0

JSON Path

- *//*['*Range*']['*Start*']

Example JSON:

```
{"Start": 6.0}
```

3.1.37 Delta

Step size

Type number

Required True

Default 1

JSON Path

- //['*Range*']['*Delta*']

Example JSON:

```
{"Delta": 1}
```

3.1.38 Steps

Number of Steps

Type integer

Required True

Default 10

JSON Path

- //['*Range*']['*Steps*']

Example JSON:

```
{"Steps": 10}
```

3.1.39 Log

Logarithmic spacing of steps

Type boolean

Required False

Default False

JSON Path

- //['*Range*']['*Log*']

Old Symbol `IC(22)`

Example JSON:

```
{"Log": false}
```

3.1.40 Value

If no *Range* is given this is the constant value

Type number

Required True

Default 1.0

JSON Path

- //['*Value*']
- /['*TransformationType*']['*InverseLorenzMie*']['*mValueRealPart*']['*Value*']
- /['*TransformationType*']['*InverseLorenzMie*']['*mValueImaginaryPart*']['*Value*']
- /['*TransformationType*']['*InverseLorenzMie*']['*AntireflectionCorrection*']['*ReflectionParameter*']['*Value*']
- /['*TransformationType*']['*InverseLaplace*']['*Baseline*']['*Value*']

Example JSON:

```
{"Value": 1.0}
```

3.1.41 LagrangeMultiplier

Give the *Range* for the Lagrange multiplier λ.

Type object

Contains *comment*, *Range**

Required False

JSON Path

- //['*LagrangeMultiplier*']
- /['*TransformationType*']['*InverseLorenzMie*']['*LagrangeMultiplier*']
- /['*TransformationType*']['*InverseLaplace*']['*LagrangeMultiplier*']

Example JSON:

```
{
  "LagrangeMultiplier": {
    "Range": {
      "Start": 6.0,
      "Steps": 10,
      "Delta": 1
    }
  }
}
```

3.1.42 comment

The comment object is alowed everywhere and can be used to deactivate sections or to add additional information. It might contain any valid JSON.

Type object

Contains

Required

False

Default OrderedDict()

JSON Path

- //['*comment*']
- /['*comment*']
- /['*InputData*']['*comment*']
- /['*InputData*']['*DataRange*']['*comment*']

- /['*InputData*']['*SubtractLinearFunction*']['*comment*']
- /['*TransformationType*']['*comment*']
- /['*TransformationType*']['*InverseLorenzMie*']['*comment*']
- /['*TransformationType*']['*InverseLorenzMie*']['*mValueRealPart*']['*comment*']
- /['*TransformationType*']['*InverseLorenzMie*']['*mValueImaginaryPart*']['*comment*']
- /['*TransformationType*']['*InverseLorenzMie*']['*AntireflectionCorrection*']['*ReflectionParameter*']['*comment*']
- /['*TransformationType*']['*InverseLaplace*']['*comment*']
- /['*TransformationType*']['*InverseLaplace*']['*Baseline*']['*comment*']
- /['*Distribution*']['*comment*']
- /['*Distribution*']['*Interval*']['*comment*']
- /['*Distribution*']['*Splines*']['*comment*']
- //['*Range*']['*comment*']
- //['*LagrangeMultiplier*']['*comment*']

Example JSON:

```
{"comment": {}}
```

CHAPTER

FOUR

ORTLIGHT MODULE API

ortlight.**callort**(*filename='input.ortl'*)
driver for ort.exe fortran kernel

ortlight.**ort2json**(*text*)
convert .ort to .ortl

ortlight.**json2ort**(*text*)
Convert .ortl to .ort

C

J

O