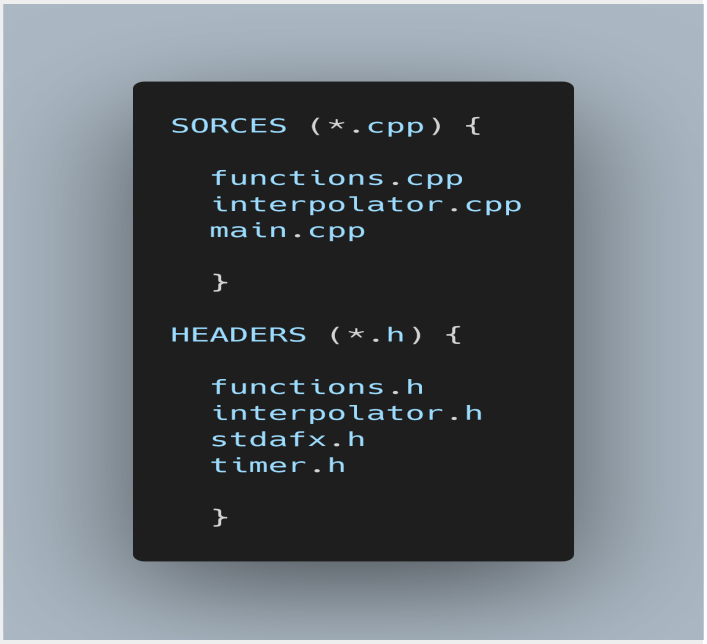
Programme module's description

Contents:

Structure

Programme's source code is represented by 3 (**.cpp*) files each having its own corresponding *header* file (**.h*):

- *functions.cpp*
- *interpolator.cpp*
- *main.cpp*



Header files:

- *functions.h*
- *interpolator.h*
- *stdafx.h*
- *timer.h*

functions.cpp

Contains realisation of methods declared in functions.h
In turn, this .cpp is visually split into 7 major groups of functions (just on purpose of an easier navigation through it):

- .TXT FILE PROCESSING FUNCTIONS,
- .INP FILE PROCESSING FUNCTIONS,
- .CSV FILE PROCESSING FUNCTIONS,
- T FIELD FILE PROCESSING FUNCTIONS,
- CONCENTRATIONS FILE PROCESSING

Each of the above sections is represented by a *parser function (txtParse, inpParse, csvParse, concParse)* and *mesh-getter function (GetMeshTXT, GetMeshINP,* GetMeshCSV*)*. Besides parser and *mesh-getter*, the CSV section has *GetTempCSV* member function, which purpose is to retrieve T values from a csv table (those are the headers of columns).

- GRID PROCESSING FUNCTIONS includes *interMesh* function, which is responsible for interpolation of meshes passed to it.
- MATH FUNCTIONS AND BOOLEAN FILTERS represent mathematical method of integration (*Trapezoidal*, once overloaded), physical functions (*Planck*), and binary predicates (isinX, isinT).

interpolator.cpp

Along with its declaration header (*interpolator.h*) represent class which implements linear interpolation (linearly weighted) to mesh passed into its class' object as parameter. Interpolation by temperatures' grid is conducted implicitly within the procedure of getting mesh out of the data file.

timer.h

Auxiliary single-filed class used within *main.cpp* to estimate physical time of programme's execution. To do the latter, just an instance of the class needs to be declared within the scope of interest (timer starts and ends within the realm enshrouds the instance).

stdafx.h

Contains preprocessor directives with libs to be used further in the programme.

main.cpp

The main file. Consists of only *main()* function.
The programme's execution goes as follows:

1. **On lines 8 – 26** here are t (object of class *Timer* declared in *timer.h*; calculates physical time of code execution within the scope it encased in), paths to the input files, and some data containers (*kabs*, *mesh2*, *Temperatures, ConcentrationsS2*).

```
6  int main()
7  {
8      Timer t; //time counter
9
10     std::string path_csv = "..\\input\\data_sample.csv";
11
12     std::string path_inp = "..\\input\\data_sample.inp";
13
14     std::string path_txt = "..\\input\\flow_sample.txt";
15
16     std::string path_con = "..\\input\\con_sample.csv";
17
18     std::string path_tmp = "..\\input\\T";
19
20     std::vector<std::vector<double>> kabs;
21
22     std::vector<std::pair<double, double>> mesh2;
23
24     std::vector<double> Temperatures = GetTempField(path_tmp);
25
26     std::vector<std::pair<double, double>> ConcentrationS2 = concParse(path_con, Temperatures);
```

```
28    // Choose the input file format for K_ABS: "inp" or "csv"
29    std::string mode = "csv";
30
31    if (mode == "inp") {
32
33        thread t1([&path_inp, &kabs]() {kabs = inpParse(path_inp);});
34
35        thread t2([&path_txt, &mesh2]() {mesh2 = GetMeshTXT(txtParse(path_txt));});
36
37        t1.join();
38
39        t2.join();
40    }
41
42    if (mode == "csv") {
43
44        thread t1([&path_csv, &kabs]() {kabs = csvParse(path_csv);});
45
46        thread t2([&path_txt, &mesh2]() {mesh2 = GetMeshTXT(txtParse(path_txt));});
47
48        t1.join();
49
50        t2.join();
51    }
```

2. **On line 29** here is *std::string mode = "csv";* – mode switcher. Switches programme's data input format for absorption coefficients.
Default: "csv"
Alternative: "inp"
Note: If you change the input format, ensure you have put a file with appropriate extension to *"..\calculation\input"* with respect to the file names suggested in lines 10 or/and 12.

3. **On lines 31 – 40 and 42 – 51** here is main programme's threading branches. With respect the mode chosen, the execution is divided into 2 sub-processes of *parsing file with absorption coefficients* and *retrieving mesh from txt-file data (intensities)*.

GENERAL MESH STRUCTURE
All the mesh (*mesh1, mesh2, mesh1new, mesh2new, func*) from now on are constructed as

std::vector<std::pair<double,double>>

where the *first double* number stands for the *wavelength*,
and the *second double* – specific mesh-related value.

```
53 /*=================CREATING=AN=OUTPUT=FILE=====================================*/
54
55     std::ofstream data;
56
57     data.open("..\\output\\output", std::ios_base::out);
58
59     data << Temperatures.size() << "\n(\n";
60
61 /*==============ITERATING=OVER=TEMPERATURE=RANGE==============================*/
62
63     for (size_t i = 0; i < Temperatures.size(); ++i){
64
65     double T = Temperatures[i];
66
67     vector<pair<double, double>> mesh1, mesh1new, mesh2new;
68
69 /*=============GETTING=MESH=FROM=KABS=========================================*/
70
71     if (mode == "inp") {mesh1 = GetMeshINP(kabs, T);}
72
73     if (mode == "csv") {mesh1 = GetMeshCSV(kabs, T);}
```

4. Snippet called **CREATING AN OUTPUT FILE** is responsible for the output file creation.
On line 57, to change output destination and/or output file name, change the first, – *std::string-type* – parameter, *"..\calculation\output"*

5. Next, **ITERATING OVER TEMPERATURE RANGE** snippet stands for repeating following procedures due to different temperatures' values previously retrieved from *T field file*.

6. **GETTING MESH FROM KABS** section, with respect to the aforechosen mode (see p.2), is responsible for obtaining mesh from inp/csv parsed data, stored in ***kabs***.

```
/*=============INTERPOLATION=============================================*/

    mesh1new = interMesh(mesh1, mesh2);

    mesh2new = interMesh(mesh2, mesh1);

/*====================ERASING=REDUNDANT=POINTS============================*/

    mesh1new.erase(std::unique(mesh1new.begin(), mesh1new.end(),
             [](const std::pair<double, double>& a, const std::pair<double, double>& b)
             {return std::abs(a.first - b.first) < 1e-11;}), mesh1new.end());

    mesh2new.erase(std::unique(mesh2new.begin(), mesh2new.end(),
             [](const std::pair<double, double>& a, const std::pair<double, double>& b)
             {return std::abs(a.first - b.first) < 1e-11;}), mesh2new.end());

```

7. **INTERPOLATION** section implements *interMesh* function to the meshes retrieved from txt-file with intensities and inp/csv-file with absorption coefficients.

```
/*=======================SORTING=CHECK=====================================*/

    if (!std::is_sorted(std::begin(mesh1new), std::end(mesh1new),
                  [](std::pair<double, double>& a, std::pair<double, double>& b)
                  {return a.first < b.first;}))
             {std::sort(std::begin(mesh1new), std::end(mesh1new),
             [](std::pair<double, double>& a, std::pair<double, double>& b)
             {return a.first < b.first;});}

    if (!std::is_sorted(std::begin(mesh2new), std::end(mesh2new),
                       [](std::pair<double, double>& a, std::pair<double, double>& b)
                       {return a.first < b.first;}))
             {std::sort(std::begin(mesh2new), std::end(mesh2new),
             [](std::pair<double, double>& a, std::pair<double, double>& b)
             {return a.first < b.first;});}
```

8. **ERASING REDUNDANT POINTS** piece takes a double-check on whether the interpolated meshes contain equal points by sieving latter with the accuracy given (*1e-11*, for wavelength).

9. **SORTING CHECK** section is made up to ensure the mesh being processed (*mesh1new*, *mesh2new*) is sorted in ascending order by the first argument of a point (*wavelength*).

```
107 /*=========================CALCULATION=======================================*/
108
109     std::vector<std::pair<double, double>> func/*, func_eq*/;
110
111     for (size_t i = 0; i < mesh1.size(); ++i) {
112
113         double mlt = mesh1[i].second * mesh2new[i].second *
114                      mesh1[i].first * 5.0307e33;
115         if (mlt > 1e-10 && !std::isnan(mlt)) {func.push_back(
116                         std::make_pair(mesh1[i].first, mlt));}
117     }
118
119     func.pop_back();
```

10. **CALCULATION** section is responsible for creating a container *func* and filling it with points of under-integral function of:

$$W = 4\pi z \int_{\lambda 1^*}^{\lambda 2^*} \frac{F(\lambda) * 10^9 k_{abs}(\lambda)\,\mathrm{d}\lambda}{h\frac{c}{\lambda}}$$

but without $4\pi z/C$ coefficient (C – *concentration, [1/m^3]*).

```
121 /*=====================ERASING=REDUNDANT=POINTS==============================*/
122
123     func.erase(std::unique(func.begin(), func.end(),
124             [](const std::pair<double, double>& a, const std::pair<double, double>& b)
125             {return std::abs(a.first - b.first) < 1e-11;}), func.end());
126
127 /*=======================INTEGRATION=========================================*/
128
129     double result = 4 * M_PI * integrateTrapezoidal(func) / ConcentrationS2[i].second;
130
131     data << result << '\n';     //writing into output file
132
133 /*==============OUTPUTTING=RESULTS=========================================*/
134
135 //    std::cout << T <<  " K\tRate:\t " << result  << '\n';
136
137 /*=========================================================================*/
138
139     }
140
141     data << ")\n;\n";
142
143     data.close();   //closing output file
```

11. **ERASING REDUNDANT POINTS** snippet is on the same purpose as described in p.8. Although, this time it operates with the wavelength-vs-under-integral-function points' set.

12. **INTEGRATION** section implements *integrateTrapezoidal* function (declared in *functions.cpp)* over *func* data range as parameter.

13. **OUTPUTTING RESULTS** is an optional section which, if uncommented, would show the integral's value in console.

14. The last, **on lines 141 and 143**, bunch of commands manages to end the insertion of the results into the output file and closes the latter, respectively.

Input/Output

All input files should by default be put in *".\calculation\input".*
NOTE: It is possible to use one of the two – csv or inp – file formats.
Input files:

data.csv – file with absorption coefficients [*1/cm*], contains header line, each column separated either by a comma or by a semicolon.
The first column is wavelength-range [*nm*].
The rest of columns are absorption coefficients' ranges corresponding to a *particular temperature*.

	A	B	C
1	lam,nm\T,K	T=300	T=800
2	20	147.262	55.4043
3	20.025	147.815	55.6123
4	20.05	148.369	55.8209
5	20.075	148.925	56.0299
6	20.1	149.482	56.2395
7	20.125	150.04	56.4496
8	20.15	150.6	56.6603
9	20.175	151.161	56.8714
10	20.2	151.724	57.0831

data.inp – file with absorption coefficients [*1/m*], contains lines with those coefficients, but the first two values of a line are the *line's index* and *temperature* respectively.
The first line contains two numbers separated by *space*: *total lines number*, and *the number of values within each one* (excluding first two numbers within).
The last line contains number of wavelength-values followed by a heading and then by wavelength-values themselves.

1	70 7201
2	0 300.0 1.472619e+04 1.478149e+0
3	2.096757e+04 2.103754e+04
4	2.876210e+04 2.884847e+04
5	3.828235e+04 3.838685e+04
6	4.970090e+04 4.982526e+04
7	6.319032e+04 6.333626e+04

flow.txt – file containing 4 columns of numbers, wherein the *first column* contains wavelength in [*m*], the *second* – intensity [*W/m2/sr/nm*]. Other columns are beyond our interest.

```
2.0000E-08  1.8752E-01 NaN          4.0316E+00
2.1000E-08  3.9440E-01 NaN          4.6670E+00
2.2000E-08  6.9927E-01 NaN          5.3660E+00
2.3000E-08  1.0611E+00 NaN          6.1315E+00
2.4000E-08  1.3957E+00 NaN          6.9665E+00
2.5000E-08  1.6091E+00 NaN          7.8741E+00
2.6000E-08  1.6422E+00 NaN          8.8573E+00
2.7000E-08  1.4971E+00 NaN          9.9191E+00
2.8000E-08  1.2295E+00 NaN          1.1063E+01
```

T – temperatures' field. See example:

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.x                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "1";
    object      T;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 1 0 0 0];

internalField   nonuniform List<scalar>
1 // Number of following T values within ();
(
5000 //T values
)
;

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform 1323;
    }
    outlet
    {
        type            inletOutlet;
        inletValue      uniform 1323;
        value           uniform 1338.61;
    }
    wall
    {
        type            zeroGradient;
    }
    wedgeRight
    {
        type            wedge;
    }
    wedgeLeft
    {
        type            wedge;
    }
    defaultFaces
    {
        type            empty;
    }
}


// ************************************************************************* //
```

con.csv – file with concentrations [1/m^3] generated by FWB. The result of the integration is divided by the respective concentration value to yield the frequency of S_2 generation in [1/s].

	A	B	C
1	T	S	S2
2	5.000000000000e+03	1.447887824649e+24	1.991680906408e+22

The first column represents T values [K].
The rest of the columns are concentration values of the proper substance mentioned in header line.

Note: At the current time the programme reads *conc_sample.csv* file just by tying the temperature value given in the first column with the value inn the third column, S_2 concentration. No *RegEx* is used due to the features of the task.

Output:

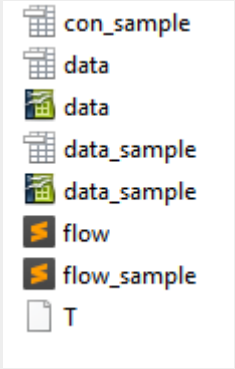
output – file with the integral values with respect to the temperatures given in T file, [1/s]. See example:

```
1 1 //Number of the following Integral's values within ();
2 (
3 4.96652e+021 //Values
4 )
5 ;
6
```

Build

Built by means of *Desktop Qt 5.14.2 MinGW 32-bit*

F.A.Q.



1. What are the other files within the *"..\calculation\input"*?

   As long as we're now working on the S_2 excitation case, we only need files named with use of *"_sample"* suffix. (Apart from *T file*. It's still used as it is: no need to remove it) Other files are related to the original TT-case.

2. How do I change input/output file destinations or names?

   To change input files' names or placements, change proper parameters in *lines 10 – 18 of main.cpp*. *For output file – line 57.*

3. How do I launch the programme?

   To launch, run *"..\calculation_build\release\calculation.exe"*.

4. How do I change something in the programme?

   To be able to put your changes in action, you need to re-*build* the programme after having changed it. See Build section to ensure use of the proper building system.

5. How do I direct the output to the console?

   Go to *main.cpp*. Uncomment the following block of code:

```
145 /*===========FLOW=FILE=GENERATING=============================================*/
146
147 //    double te = 5000;
148
149 //    std::ofstream flow("..\\output\\flow_my.txt", std::ios_base::out);
150
151 ////    std::vector<std::vector<double>> table_csv = csvParse("..\\input\\absS2_my.csv", ';');
152
153 ////    std::vector<std::pair<double, double>> m_csv = GetMeshCSV(table_csv, te);
154
155 //    for (size_t i = 20; i < 801; ++i) {
156
157 //        flow << " " << i * 1e-9 << " " << Planck(i * 1e-9, te) / 1e-9
158 //             << " " << NAN << "  " << NAN << '\n';
159 //    }
160
161 //    flow.close();
162
163 /*=============================================================================*/
```

By the way, to call off the DURATION outputting, comment the *line 8 in main.cpp (Timer t;)*