LuminetCpp

Generated by Doxygen 1.10.0

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

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

Chapter 3

Class Documentation

3.1 angular_properties Struct Reference

Public Attributes

- double **start_angle** = 0.0
- double **end_angle** = M_PI
- unsigned **angular_precision** = 500
- bool **mirror** = true

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

- Include/utilities.h

3.2 BHphysics Class Reference

Public Member Functions

- double **zeta_r** (double periastron, double r, double bh_mass)
- void **get_plot** (const std::vector< double > &, const std::vector< double > &, const std::vector< double > &, const double &)
- double **cos_alpha** (double phi, double incl)
- double **alpha** (double phi, double incl)
- std::vector< double > **filter_periastrons** (const std::vector< double > &periastron, double bh_mass, double tol)
- double **phi_inf** (double periastron, double M)
- double **mu** (double periastron, double bh_mass)

Static Public Member Functions

- static double **calc_q** (double periastron, double bh_mass, double)
- static double **calc_b_from_periastron** (double periastron, double bh_mass, double)
- static double **k** (double periastron, double bh_mass)
- static double **k2** (double periastron, double bh_mass, double)
- static double **zeta_inf** (double periastron, double bh_mass, double tol)
- static double **cos_gamma** (double _a, double incl, double tol)
- static double **eq13** (double periastron, double ir_radius, double ir_angle, double bh_mass, double incl, int n, double tol)
- static std::tuple< std::vector< double >, std::vector< double >, int > **midpoint_method** (const std::function< double(double, double, double, double, double, int, double)> func, const std::unordered_map< std::string, double > &args, const std::vector< double > &x, const std::vector< double > &y, int index_of_sign_change)
- static double **improve_solutions_midpoint** (const std::function< double(double, double, double, double, double, int, double)> &func, const std::unordered_map< std::string, double > &args, const std::vector< double > &x, const std::vector< double > &y, int index_of_sign_change, int iterations)
- static double **calc_periastron** (double _r, double incl, double _alpha, double bh_mass, int midpoint_iterations, bool plot_inbetween, int n, double min_periastron, int initial_guesses)
- static double **calc_impact_parameter** (double _r, double incl, double _alpha, double bh_mass, int midpoint_iterations, bool plot_inbetween, int n, double min_periastron, int initial_guesses, bool use_ellipse)
- static double **ellipse** (double r, double a, double incl)
- static double **flux_intrinsic** (double r, double acc, double bh_mass)
- static double **flux_observed** (double r, double acc, double bh_mass, double redshift_factor)
- static double **redshift_factor** (double radius, double angle, double incl, double bh_mass, double b_)
- static int **find_index_sign_change_indices** (const std::vector< double > &)

The documentation for this class was generated from the following files:

- Include/BlackHolePhysics.h
- Source/BlackHolePhysics.cpp

3.3 BlackHole Class Reference

Public Member Functions

- **BlackHole** (double mass=1.0, double inclination=80, double acc=1e-8)
- void **sample_Sources** (int n_points=1000, const std::string &f="", const std::string &f2="")
- void **calc_isoradials** (const std::vector< double > &direct_r, const std::vector< double > &ghost_r)
- void **add_isoradial** (Isoradial &isoradial, double radius, int order)
- std::map< double, IsoRedShift > **calc_isoredshifts** (std::vector< double > redshifts={ -0.15, 0.0, 0.1, 0.2, 0.5 })

The documentation for this class was generated from the following files:

- Include/BlackHole.h
- Source/BlackHole.cpp

3.4 CSVRow Class Reference

Public Member Functions

- std::string_view **operator[ ]** (std::size_t index) const
- std::size_t **size** () const
- void **readNextRow** (const std::string &line)

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

- Include/utilities.h

3.5 find_redshift_params Struct Reference

Public Attributes

- bool **force_redshift_solution** = false
- unsigned **max_force_iter** = 5

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

- Include/utilities.h

3.6 ir_params Struct Reference

Public Attributes

- double **start_angle** = 0.0
- double **end_angle** = M_PI
- unsigned **angular_precision** = 500
- bool **mirror** = true
- double **angular_margin** = 0.3

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

- Include/utilities.h

3.7 irs_solver_params Struct Reference

Public Attributes

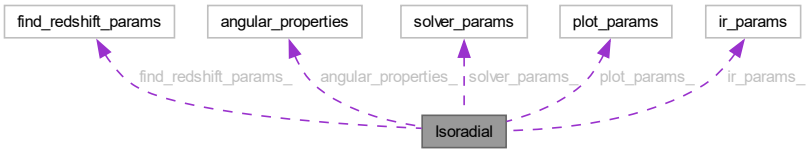
- unsigned **initial_guesses** = 12
- unsigned **midpoint_iterations** = 12
- double **times_inbetween** = 2
- unsigned **retry_angular_precision** = 15
- double **min_periastron** = 3.01
- bool **use_ellipse** = true
- unsigned **retry_tip** = 50
- unsigned **initial_radial_precision** = 15
- bool **plot_inbetween** = false
- double **angular_margin** = 0.3

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

- Include/utilities.h

3.8 Isoradial Class Reference

Collaboration diagram for Isoradial:



Public Member Functions

- **Isoradial** (double radius, double incl, double bh_mass, int order=0)
- **Isoradial** (double radius, double incl, double bh_mass, int order, angular_properties)
- std::pair< std::vector< double >, std::vector< double > > **get_bare_isoradials** ()
- std::vector< double > **get_redshift_factors** ()
- void **calculate** ()
- std::pair< std::vector< double >, std::vector< double > > **calculate_coordinates** ()
- std::vector< double > **calc_redshift_factors** ()
- std::vector< double > **find_angle** (double z)
- double **get_b_from_angle** (double angle)
- void **calc_between** (int ind)
- std::vector< double > **force_intersection** (double redshift)
- std::pair< std::vector< double >, std::vector< double > > **calc_redshift_location_on_ir** (double redshift, bool cartesian=false)

Public Attributes

- find_redshift_params **find_redshift_params_**
- angular_properties **angular_properties_**
- solver_params **solver_params_**
- plot_params **plot_params_**
- ir_params **ir_params_**
- std::vector< double > **X**
- std::vector< double > **Y**
- std::vector< double > **_radii_b**
- std::vector< double > **_angles**

The documentation for this class was generated from the following files:

- Include/IsoRadials.h
- Source/IsoRadials.cpp

3.9 IsoRedShift Class Reference

Public Member Functions

- **IsoRedShift** (const double &, const double &, const double &, const std::map< double, std::map< int, Isoradial > >)
- void **improve** ()
- void **update** ()
- void **add_solutions** (const std::vector< double > &angles, const std::vector< double > &impact_↩ parameters, double radius_ir)
- std::pair< std::vector< double >, std::vector< double > > **extract_co_from_solutions_dict** ()
- void **calc_from_isoradials** (const std::vector< Isoradial > &isoradials, bool cartesian=false)
- std::pair< std::vector< double >, std::vector< double > > **calc_core_coordinates** ()
- void **order_coordinates** (const std::string &plot_title="", bool plot_inbetween=false)
- void **calc_ir_before_closest_ir_wo_z** (double angular_margin)
- void **recalc_isoradials_wo_redshift_solutions** (bool plot_inbetween)
- void **improve_tip** (int iterations)
- void **improve_between_all_solutions_once** ()
- std::pair< std::vector< double >, std::vector< double > > **calc_redshift_on_ir_between_angles** (double radius, double begin_angle, double end_angle, int angular_precision, bool mirror, bool plot_inbetween, const std::string &title, bool force_solution)
- std::pair< std::vector< double >, std::vector< double > > **recalc_redshift_on_closest_isoradial_wo_z** (double angular_margin)
- std::pair< std::map< double, std::vector< std::vector< double > > >, std::map< double, std::vector< std↩ ::vector< double > > > > **split_co_on_solutions** ()
- std::vector< double > **get_ir_radii_w_co** ()

The documentation for this class was generated from the following files:

- Include/IsoRedShift.h
- Source/IsoRedShift.cpp

3.10 OperatorsOrder2 Class Reference

Public Member Functions

- **OperatorsOrder2** (int nGrid, double delta)
- std::vector< std::vector< std::vector< double > > > **laplace** (const std::vector< std::vector< std::vector< double > > > &fct)
- std::tuple< std::vector< std::vector< std::vector< double > > >, std::vector< std::vector< std::vector< double > > >, std::vector< std::vector< std::vector< double > > > > **gradient** (const std::vector< std::vector< std::vector< double > > > &fct)
- std::vector< std::vector< std::vector< double > > > **divergence** (const std::vector< std::vector< std::vector< double > > > &v_x, const std::vector< std::vector< std::vector< double > > > &v_y, const std::vector< std::vector< std::vector< double > > > &v_z)
- std::tuple< std::vector< std::vector< std::vector< double > > >, std::vector< std::vector< std::vector< double > > >, std::vector< std::vector< std::vector< double > > > > **gradDiv** (const std::vector< std::vector< std::vector< double > > > &A_x, const std::vector< std::vector< std::vector< double > > > &A_y, const std::vector< std::vector< std::vector< double > > > &A_z)
- std::tuple< double, double, double > **partialDerivs** (const std::vector< std::vector< std::vector< double > > > &fct, int i, int j, int k)
- std::vector< std::vector< std::vector< double > > > **Add** (const std::vector< std::vector< std::vector< double > > > &, const double &, const std::vector< std::vector< std::vector< double > > > &, const double &)

Static Public Member Functions

- static std::vector< double > **linspace** (const double &, const double &, const int &)
- static std::pair< std::vector< double >, std::vector< double > > **polar_to_cartesian_lists** (const std::vector< double > &radii, const std::vector< double > &angles, const double &rotation)
- static std::pair< double, double > **polar_to_cartesian_single** (double th, double radius, double rotation)
- static std::pair< double, double > **cartesian_to_polar** (double x, double y)
- static double **get_angle_around** (const std::vector< double > &p1, const std::vector< double > &p2)

The documentation for this class was generated from the following files:

- Include/TensorCalculus.h
- Source/TensorCalculus.cpp

3.11 plot_params Struct Reference

Public Attributes

- bool **plot_isoredshifts_inbetween** = false
- bool **save_plot** = false
- bool **plot_ellipse** = false
- bool **plot_core** = true
- bool **redshift** = true
- std::string **linestyle** = "-"
- double **linewidth** = 1.
- std::string **key** = ""
- std::string **face_color** = "black"
- std::string **line_color** = "white"

- std::string **text_color** = "white"
- double **alpha** = 1.
- bool **show_grid** = false
- bool **legend** = false
- bool **orig_background** = false
- bool **plot_disk_edges** = false
- std::pair< double, double > **ax_lim** = { -100, 100 }
- std::string **title** = "Isoradials for R ="

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

- Include/utilities.h

3.12 Plotter Class Reference

Public Member Functions

- void **plot** (std::vector< double > &, std::vector< double > &, std::vector< double > &, std::vector< double > &)
- void **plot** (double, std::vector< double > &, std::vector< double > &, std::vector< double > &, std::vector< double > &, std::vector< double > &, std::vector< double > &)

The documentation for this class was generated from the following files:

- Include/plotter.h
- Source/plotter.cpp

3.13 solver_params Struct Reference

Public Attributes

- unsigned **initial_guesses** = 12
- unsigned **midpoint_iterations** = 20
- bool **plot_inbetween** = false
- double **min_periastron** = 3.001
- bool **use_ellipse** = true

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

- Include/utilities.h

3.14 Source Struct Reference

Public Attributes

- double **X**
- double **Y**
- double **impact_parameter**
- double **angle**
- double **z_factor**
- double **flux_o**

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

- Include/BlackHole.h

Chapter 4

File Documentation

4.1 BlackHole.h

```
00001 #pragma once
00002 #ifndef BLACKHOLE_H
00003 #define BLACKHOLE_H
00004 #include <iostream>
00005 #include <cmath>
00006 #include <fstream>
00007 #include <random>
00008 #include <algorithm>
00009 #include <unordered_map>
00010
00011 #include "TensorCalculus.h"
00012 #include "BlackHolePhysics.h"
00013 #include "IsoRadials.h"
00014 #include "IsoRedShift.h"
00015 #include "utilities.h"
00016
00017 struct Source {
00018     double X, Y, impact_parameter, angle, z_factor, flux_o;
00019 };
00020
00021 class BlackHole {
00022 public:
00023     // Constructor and other methods go here...
00024     BlackHole();
00025     BlackHole(double mass = 1.0, double inclination = 80, double acc = 1e-8);
00026     ~BlackHole();
00027     void sample_Sources(int n_points = 1000, const std::string& f = "", const std::string& f2 = "");
00028     void calc_isoradials(const std::vector<double>& direct_r, const std::vector<double>& ghost_r);
00029     void add_isoradial(Isoradial& isoradial, double radius, int order);
00030     std::map<double, IsoRedShift> calc_isoredshifts(std::vector<double> redshifts = { -0.15, 0.0, 0.1,
      0.2, 0.5 });
00031
00032 private://variables
00033     double inclination;
00034     double t;
00035     double M;
00036     double acc;
00037     double disk_outer_edge;
00038     double disk_inner_edge;
00039
00040     //std::vector<double>  isoradials;
00041     //std::vector<double> isoredshifts;
00042     std::map<double, std::map<int, Isoradial» isoradials;
00043     std::map<double, IsoRedShift> isoredshifts;
00044
00045     //make struct from this variables
00046     //std::unordered_map<std::string, double> settings;
00047     plot_params plot_params_;
00048     ir_params ir_parameters_;
00049     angular_properties angular_properties_;
00050     irs_solver_params irs_solver_params_;
00051     solver_params solver_params_;
00052
00053     /*int initial_guesses;
00054     int midpoint_iterations;
00055     bool plot_inbetween;
00056     bool use_ellipse;*/
00057     double min_periastron;
```

```
00058
00059     double critical_b;
00060     int angular_precision;
00061 private://methods
00062     Isoradial calc_apparent_outer_disk_edge();
00063     Isoradial calc_apparent_inner_disk_edge();
00064     double get_apparent_outer_edge_radius(Isoradial&, double angle, double rotation);
00065     double get_apparent_inner_edge_radius(Isoradial&, double angle, double rotation);
00066     std::pair<std::vector<double>, std::vector<double» apparent_inner_edge(Isoradial&, bool cartesian
      = true, double scale = 0.99);
00067     std::map<double, std::map<int, Isoradial» get_dirty_isoradials();
00068 };
00069 #endif
```

4.2 BlackHolePhysics.h

```
00001 #pragma once
00002 #include <iostream>
00003 #include <cmath>
00004 #include <vector>
00005 #include <cmath>
00006 #include <functional>
00007 #include <map>
00008 #include <boost/math/special_functions/jacobi_elliptic.hpp>
00009 #include "TensorCalculus.h"
00010 #include "utilities.h"
00011
00012 //using namespace std;
00013 //-------------------------------------------------------------BLACK-HOLE-MATH.PY
00014
00015 class BHphysics
00016 {
00017 public:
00018     BHphysics();
00019     static double calc_q(double periastron, double bh_mass, double);
00020     static double calc_b_from_periastron(double periastron, double bh_mass, double);
00021
00022     static double k(double periastron, double bh_mass);
00023
00024     static double k2(double periastron, double bh_mass, double);
00025     static double zeta_inf(double periastron, double bh_mass, double tol);
00026     double zeta_r(double periastron, double r, double bh_mass);
00027     static double cos_gamma(double _a, double incl, double tol);
00028     void get_plot(const std::vector<double>&, const std::vector<double>&, const std::vector<double>&,
      const double&);
00029
00030     double cos_alpha(double phi, double incl);
00031
00032     double alpha(double phi, double incl);
00033     std::vector<double> filter_periastrons(const std::vector<double>& periastron, double bh_mass,
      double tol);
00034
00035     static double eq13(double periastron, double ir_radius, double ir_angle, double bh_mass, double
      incl, int n, double tol);
00036     static std::tuple<std::vector<double>, std::vector<double>, int> midpoint_method(
00037         const std::function<double(double, double, double, double, double, int, double)> func,
00038         const std::unordered_map<std::string, double>& args,
00039         const std::vector<double>& x,
00040         const std::vector<double>& y,
00041         int index_of_sign_change);
00042
00043     static double improve_solutions_midpoint(
00044         const std::function<double(double, double, double, double, double, int, double)>& func,
00045         const std::unordered_map<std::string, double>& args,
00046         const std::vector<double>& x,
00047         const std::vector<double>& y,
00048         int index_of_sign_change,
00049         int iterations
00050     );
00051
00052     static double calc_periastron(double _r, double incl, double _alpha, double bh_mass, int
      midpoint_iterations, bool plot_inbetween, int n, double min_periastron, int initial_guesses);
00053
00054     static double calc_impact_parameter(double _r, double incl, double _alpha, double bh_mass, int
      midpoint_iterations, bool plot_inbetween, int n, double min_periastron, int initial_guesses, bool
      use_ellipse);
00055
00056     double phi_inf(double periastron, double M);
00057
00058     double mu(double periastron, double bh_mass);
00059     static double ellipse(double r, double a, double incl);
00060
00061     static double flux_intrinsic(double r, double acc, double bh_mass);
```

```
00062
00063     static double flux_observed(double r, double acc, double bh_mass, double redshift_factor);
00064     static double redshift_factor(double radius, double angle, double incl, double bh_mass, double
      b_);
00065     static int find_index_sign_change_indices(const std::vector<double>&);
00066 };
```

4.3 IsoRadials.h

```
00001 #pragma once
00002 /*#include <iostream>
00003 #include <cmath>
00004 #include <fstream>
00005 #include <random>
00006 #include <algorithm>
00007 #include <unordered_map>*/
00008
00009 #include "TensorCalculus.h"
00010 #include "BlackHolePhysics.h"
00011 #include "utilities.h"
00012
00013 class Isoradial {
00014 public:
00015     Isoradial();
00016     Isoradial(double radius, double incl, double bh_mass, int order = 0);
00017     Isoradial(double radius, double incl, double bh_mass, int order, angular_properties);
00018     //Isoradial(const std::vector<double>& angles, const std::vector<double>& radius_b);
00019     std::pair<std::vector<double>, std::vector<double» get_bare_isoradials();
00020     std::vector<double> get_redshift_factors();
00021     void calculate();
00022
00023 private://variables
00024     double M;  // mass of the black hole containing this isoradial
00025     double theta_0;  // inclination of the observer's plane
00026     double radius;
00027     int order;
00028     struct params {
00029         std::string param = "isoradial_solver_parameters";
00030     };
00031
00032     std::vector<double> redshift_factors;//TO DO: pack isoradials and redshift
00033     std::tuple<std::vector<double>, std::vector<double» cartesian_co;
00034     std::pair<std::vector<double>, std::vector<double» bare_isoradials;//TEMPORARY for debugging:
      holds the polar coordinates (angles, radii) of the projected isoradial
00035
00036 private://methods
00037
00038 public://methods
00039
00040     std::pair<std::vector<double>, std::vector<double» calculate_coordinates();
00041     std::vector<double> calc_redshift_factors();
00042
00043     std::vector<double> find_angle(double z);
00044     double get_b_from_angle(double angle);
00045     void calc_between(int ind);
00046     std::vector<double> force_intersection(double redshift);
00047     std::pair<std::vector<double>, std::vector<double» calc_redshift_location_on_ir(double redshift,
      bool cartesian = false);
00048
00049 public://variables ? make get method?
00050
00051     find_redshift_params find_redshift_params_;
00052     angular_properties angular_properties_;
00053     solver_params solver_params_;
00054     plot_params plot_params_;
00055     ir_params ir_params_;
00056
00057     std::vector<double> X;
00058     std::vector<double> Y;
00059     std::vector<double> _radii_b;
00060     std::vector<double> _angles;
00061 };
```

4.4 IsoRedShift.h

```
00001 #pragma once
00002 #include <iostream>
00003 #include <cmath>
00004 #include <fstream>
```

```
00005 #include <random>
00006 #include <algorithm>
00007 #include <unordered_map>
00008 #include <iostream>
00009 #include <cmath>
00010 #include <vector>
00011 #include <cmath>
00012 #include <functional>
00013 #include <map>
00014 #include <numeric>
00015
00016 #include "TensorCalculus.h"
00017 #include "BlackHolePhysics.h"
00018 #include "IsoRadials.h"
00019 #include "utilities.h"
00020
00021 class IsoRedShift {
00022 private://variables
00023     double theta_0;  // Inclination
00024     double redshift;
00025     double M;  // Black hole mass
00026     std::unordered_map<double, std::vector<std::vector<double>> radii_w_coordinates_dict;
00027     //*****************************************************************************************
00028     //                                                v
00029     //std::unordered_map<std::pair<double, double>, double> coordinates_with_radii_dict;//This give a
    problem
00030     // I have to rewrite this for the methods
00031     //     std::unordered_map<std::pair<double, double>, double> init_co_to_radii_dict();
00032     // and std::pair<std::vector<double>, std::vector<double>> extract_co_from_solutions_dict();
00033     //                                                 o
00034     //*****************************************************************************************
00035
00036     std::vector<double> angles;
00037     std::vector<double> radii;
00038     std::vector<double> x;
00039     std::vector<double> y;
00040
00041     //std::map<double, std::vector<double>> radii_w_coordinates_dict;
00042     //std::vector<double> ir_radii_w_co;
00043     //std::pair<std::vector<double>, std::vector<double>> co;
00044     double max_radius;
00045     irs_solver_params irs_solver_params_;
00046     find_redshift_params find_redshift_params_;
00047     angular_properties angular_properties_;
00048     solver_params solver_params_;
00049     plot_params plot_params_;
00050     ir_params ir_params_;
00051
00052 private://methods
00053     //void update();
00054     /*void add_solutions(const std::vector<double>& angles, const std::vector<double>&
    impact_parameters, double radius_ir);
00055     std::unordered_map<std::pair<double, double>, double> init_co_to_radii_dict();
00056     std::pair<std::vector<double>, std::vector<double>> extract_co_from_solutions_dict();
00057     void calc_from_isoradials(const std::vector<Isoradial>& isoradials, bool cartesian = false);
00058     std::pair<std::map<double, std::vector<std::vector<double>>>, std::map<double,
    std::vector<std::vector<double>>> split_co_on_solutions();
00059     std::pair<std::vector<double>, std::vector<double>> calc_core_coordinates();
00060     void order_coordinates(const std::string& plot_title = "", bool plot_inbetween = false);
00061     void calc_ir_before_closest_ir_wo_z(double angular_margin);
00062     std::pair<std::vector<double>, std::vector<double>> calc_redshift_on_ir_between_angles(
00063         double radius, double begin_angle, double end_angle, int angular_precision, bool mirror,
00064         bool plot_inbetween, const std::string& title, bool force_solution);*/
00065
00066 public://methods
00067     IsoRedShift();
00068     IsoRedShift(const double&, const double&, const double&, const std::map<double, std::map<int,
    Isoradial>>);
00069     ~IsoRedShift();
00070     void improve();
00071     void update();
00072
00073     void add_solutions(const std::vector<double>& angles, const std::vector<double>&
    impact_parameters, double radius_ir);
00074     //std::unordered_map<std::pair<double, double>, double> init_co_to_radii_dict();
00075     std::pair<std::vector<double>, std::vector<double>> extract_co_from_solutions_dict();
00076     void calc_from_isoradials(const std::vector<Isoradial>& isoradials, bool cartesian = false);
00077
00078     std::pair<std::vector<double>, std::vector<double>> calc_core_coordinates();
00079     void order_coordinates(const std::string& plot_title = "", bool plot_inbetween = false);
00080     void calc_ir_before_closest_ir_wo_z(double angular_margin);
00081     void recalc_isoradials_wo_redshift_solutions(bool plot_inbetween);
00082     void improve_tip(int iterations);
00083     void improve_between_all_solutions_once();
00084     std::pair<std::vector<double>, std::vector<double>> calc_redshift_on_ir_between_angles(
00085         double radius, double begin_angle, double end_angle, int angular_precision, bool mirror,
00086         bool plot_inbetween, const std::string& title, bool force_solution);
```

```
00087     std::pair<std::vector<double>, std::vector<double»
      recalc_redshift_on_closest_isoradial_wo_z(double angular_margin);
00088     std::pair<std::map<double, std::vector<std::vector<double»>, std::map<double,
      std::vector<std::vector<double»» split_co_on_solutions();
00089
00090     std::vector<double> get_ir_radii_w_co();
00091 public://variables ? make get method?
00092 };
```

4.5 plotter.h

```
00001 #pragma once
00002 #ifndef PLOTTER_H
00003 #define PLOTTER_H
00004
00005 #include <vector>
00006 #include <iostream>
00007 #include <fstream>
00008 #include <string>
00009 #include <type_traits>
00010 #include <sstream>
00011 #include <array>
00012 #include <exception>
00013 #include <stdexcept>
00014 #include <utility>
00015 #include <unordered_map>
00016 #include <algorithm>
00017
00018 #include <discpp.h>
00019 const double a_PI = 3.14159265358979323846;
00020
00021 class Plotter {
00022 public:
00023     Plotter();
00024     ~Plotter();
00025     void plot(std::vector<double>&, std::vector<double>&, std::vector<double>&,
      std::vector<double>&);//bare isoradials
00026     void plot(double, std::vector<double>&, std::vector<double>&, std::vector<double>&,
      std::vector<double>&, std::vector<double>&, std::vector<double>&);//isoradials with redshift
00027
00028 private:// Functions to convert a value to RGB format
00029     std::vector<std::tuple<double, double, double> >convertToRGB(std::vector<double>);
00030     std::vector<double> normalize_vector(std::vector<double>);
00031 private://variables
00032     Dislin* g;
00033     double x_max;
00034     double x_min;
00035     double y_max;
00036     double y_min;
00037     int Npoints;
00038     // Vectors to store RGB color values
00039     std::vector<std::tuple<double, double, double» rgbVector;
00040     std::vector<std::tuple<double, double, double» rgbVector_g;
00041 };
00042 #endif
```

4.6 TensorCalculus.h

```
00001 #pragma once
00002 #ifndef TENSORCALCULUS_H
00003 #define TENSORCALCULUS_H
00004
00005 /*
00006 General library in progress with common operators
00007 To extend with tensor algebra operations.
00008 */
00009 #include <iostream>
00010 #include <vector>
00011 #include <tuple>
00012 #include <functional>
00013 #include <cmath>
00014
00015 #include "utilities.h"
00016 //#include <Eigen/Dense>
00017
00018 //using namespace Eigen;
00019
00020 //const double M_PI = 3.14159265358979323846;
00021 class OperatorsOrder2 {
```

```
00022 public:
00023     OperatorsOrder2(int nGrid, double delta);
00024     ~OperatorsOrder2();
00025 
00026     std::vector<std::vector<std::vector<double>> laplace(const
      std::vector<std::vector<std::vector<double>>& fct);
00027     std::tuple<std::vector<std::vector<std::vector<double>>,
      std::vector<std::vector<std::vector<double>>, std::vector<std::vector<std::vector<double>>>
      gradient(const std::vector<std::vector<std::vector<double>>& fct);
00028     std::vector<std::vector<std::vector<double>> divergence(const
      std::vector<std::vector<std::vector<double>>& v_x,
00029         const std::vector<std::vector<std::vector<double>>& v_y,
00030         const std::vector<std::vector<std::vector<double>>& v_z);
00031     std::tuple<std::vector<std::vector<std::vector<double>>,
      std::vector<std::vector<std::vector<double>>, std::vector<std::vector<std::vector<double>>>
      gradDiv(const std::vector<std::vector<std::vector<double>>& A_x,
00032         const std::vector<std::vector<std::vector<double>>& A_y,
00033         const std::vector<std::vector<std::vector<double>>& A_z);
00034     std::tuple<double, double, double> partialDerivs(const
      std::vector<std::vector<std::vector<double>>& fct, int i, int j, int k);
00035     std::vector<std::vector<std::vector<double>> Add(const
      std::vector<std::vector<std::vector<double>>&, const double&, const
      std::vector<std::vector<std::vector<double>>&, const double&);
00036     static std::vector<double>  linspace(const double&, const double&, const int&);
00037 
00038     static std::pair<std::vector<double>, std::vector<double> polar_to_cartesian_lists(const
      std::vector<double>& radii, const std::vector<double>& angles, const double& rotation);
00039     static std::pair<double, double> polar_to_cartesian_single(double th, double radius, double
      rotation);
00040     //static std::vector<double> polar_to_cartesian_single_as_vector(double th, double radius, double
      rotation);
00041     static std::pair<double, double> cartesian_to_polar(double x, double y);
00042     static double get_angle_around(const std::vector<double>& p1, const std::vector<double>& p2);
00043 
00044 private:
00045     static std::vector<double> matrixMultiply(const std::vector<std::vector<double>& matrix, const
      std::vector<double>& vector);
00046     int nGrid;
00047     double delta;
00048 };
00049 #endif
```

4.7 utilities.h

```
00001 #pragma once
00002 #ifndef UTILITIES_H
00003 #define UTILITIES_H
00004 
00005 #include <vector>
00006 #include <iostream>
00007 #include <fstream>
00008 #include <string>
00009 #include <type_traits>
00010 #include <sstream>
00011 #include <array>
00012 #include <exception>
00013 #include <stdexcept>
00014 #include <utility>
00015 #include <unordered_map>
00016 #include <algorithm>
00017 
00018 const double M_PI = 3.14159265358979323846;
00019 //INI settings are wrapped in struct
00020 struct irs_solver_params {
00021     unsigned initial_guesses = 12;
00022     unsigned midpoint_iterations = 12;
00023     double  times_inbetween = 2;// amount of times to double the precision of an isoredshift line when
      improving
00024     unsigned retry_angular_precision = 15;//angular precision to calculate isoradials with when
      improving solutions
00025     double  min_periastron = 3.01;// minimum distance to black hole(must be strictly larger than 3M),
      in units of black hole mass(photon sphere is at 3M)
00026     bool use_ellipse = true;
00027     unsigned retry_tip = 50;
00028     unsigned initial_radial_precision = 15;
00029     bool plot_inbetween = false;// plot isoredshifts while improving them
00030     double angular_margin = 0.3;
00031 };
00032 
00033 struct angular_properties {
00034     double start_angle = 0.0;
00035     double end_angle = M_PI;
00036     unsigned angular_precision = 500;
```

```
00037     bool mirror = true;
00038 };
00039
00040 struct ir_params {
00041     double start_angle = 0.0;
00042     double  end_angle = M_PI;
00043     unsigned angular_precision = 500;
00044     bool    mirror = true;// if True, calculates only half of the isoradial and mirrors it
00045     double angular_margin = 0.3;
00046 };
00047
00048 struct plot_params {
00049     bool plot_isoredshifts_inbetween = false;
00050     bool save_plot = false;
00051     bool plot_ellipse = false;
00052     bool plot_core = true;
00053     bool redshift = true;
00054     std::string linestyle = "-";
00055     double  linewidth = 1.;
00056     std::string key = "";
00057     std::string face_color = "black";
00058     std::string line_color = "white";
00059     std::string text_color = "white";
00060     double alpha = 1.;
00061     bool show_grid = false;
00062     bool legend = false;
00063     bool orig_background = false;
00064     bool plot_disk_edges = false;
00065     std::pair<double, double>   ax_lim = { -100, 100 };
00066     std::string title = "Isoradials for R =";
00067 };
00068
00069 struct solver_params {
00070     unsigned initial_guesses = 12;
00071     unsigned midpoint_iterations = 20;
00072     bool plot_inbetween = false;// plot isoredshifts while improving them
00073     double min_periastron = 3.001;// minimum distance to black hole, in units of black hole
      mass(photon sphere is at 3M)
00074     bool use_ellipse = true;
00075 };
00076
00077 struct  find_redshift_params {
00078     bool force_redshift_solution = false;
00079     unsigned max_force_iter = 5;
00080 };
00081
00082 class CSVRow
00083 {
00084 public:
00085     std::string_view operator[](std::size_t index) const
00086     {
00087         return std::string_view(&m_line[m_data[index] + 1], m_data[index + 1] - (m_data[index] + 1));
00088     }
00089     std::size_t size() const
00090     {
00091         return m_data.size() - 1;
00092     }
00093     void readNextRow(const std::string& line)
00094         //void readNextRow(std::istream& str)
00095     {
00096         //std::getline(str, m_line);
00097         m_line = line;
00098         //std::cout « m_line « std::endl;
00099         m_data.clear();
00100         m_data.emplace_back(-1);
00101         std::string::size_type pos = 0;
00102         while ((pos = m_line.find(',', pos)) != std::string::npos)
00103         {
00104             m_data.emplace_back(pos);
00105             ++pos;
00106         }
00107         // This checks for a trailing comma with no data after it.
00108         pos = m_line.size();
00109         m_data.emplace_back(pos);
00110     }
00111 private:
00112     std::string         m_line;
00113     std::vector<int>    m_data;
00114 };
00115 /*
00116 std::istream& operator»(std::istream& str, CSVRow& data)
00117 {
00118     data.readNextRow(str);
00119     return str;
00120 }*/
00121
00122 #endif
```