Projections

1.2

Generated by Doxygen 1.9.1

Chapter 1

Projection of line

The program takes three arguments: name_file x y z name_file is file with line x, y and z are coordinates of the point The program prints output of the following form segment n parameter s point x y z n is number of segment of line s is a parameter that shows the part of the segment that the projection falls on. This parameter ranges from 0 to 1. The minimum distance to the line is defined as a perpendicular. If the perpendicular does not fall on the segment, then the nearest edge of the segment is selected.

Example:

```
./main data.dat 1 1 1
```

Output:

```
Segment 2 parameter 0.75 point 1.75 0.75 0
Segment 3 parameter 0.25 point 2.25 1 0.25
```

Computing:

$$x_2(y_2)(z_2) = x_1(y_1)(z_1) - P\cos(\alpha)(\cos(\beta))(\cos(\gamma)),$$
$$P = \frac{MM_1(x_1 - x_0) + MM_2(y_1 - y_0) + MM_3(z_1 - z_0)}{\sqrt{MM_1^2 + MM_2^2 + MM_3^2}},$$
$$\cos(\alpha) = \frac{MM_1}{\sqrt{MM_1^2 + MM_2^2 + MM_3^2}},$$
$$\cos(\beta) = \frac{MM_2}{\sqrt{MM_1^2 + MM_2^2 + MM_3^2}},$$
$$\cos(\gamma) = \frac{MM_3}{\sqrt{MM_1^2 + MM_2^2 + MM_3^2}},$$

where $O(x_1, y_1, z_1)$ – an input point, $O(x_2, y_2, z_2)$ – a projected point and $O(x_0, y_0, z_0)$ – start point of piece of line.

Version

1.2

Date

2021-06-27

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Data Structure Documentation

4.1 Compute Class Reference

Basic computing class.

`#include <compute.h>`

Public Member Functions

- Compute ()

  Construct a new Compute object.
- Compute (std::vector< Point > &line, Point &input_point)

  Construct a new Compute object. This constructor calls the method compute_projections.
- void get_points_and_input (std::vector< Point > &line, Point &input_point)

  Get the points and input object. After it, This method calls compute_projections.
- void display_projections ()

  displays found values

Private Member Functions

- void compute_projections (std::vector< Point > &line)

  Compute all projections and save save in projections.
- void compute_one_projection (Point &direction_vector, Point &current_point)

  Compute all projections and save save in temp_projection.
- void check_position (Point &start_point, Point &end_point)

  Check position projection and fix if.

Private Attributes

- Point input_point
  the input point that is projected onto the line.
- Point temp_projection
  temp projection (use in methods).
- double current_parameter
  current parameter(use in methods).
- std::vector< unsigned int > segments
  found numbers of segmetns.
- std::vector< float > parameters
  found parameters.
- std::vector< Point > projections
  found projections.

4.1.1 Detailed Description

Basic computing class.

Definition at line 14 of file compute.h.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Compute() [1/2]

```
Compute::Compute ( )
```

Construct a new Compute object.

Definition at line 12 of file compute.cpp.

4.1.2.2 Compute() [2/2]

```
Compute::Compute (
            std::vector< Point > & line,
            Point & input_point )
```

Construct a new Compute object. This constructor calls the method compute_projections.

Parameters

line	the input line (sequence of points).
input_point	the input point that is projected onto the line.

Definition at line 14 of file compute.cpp.

4.1.3 Member Function Documentation

4.1.3.1 check_position()

```
void Compute::check_position (
            Point & start_point,
            Point & end_point )  [private]
```

Check position projection and fix if.

Parameters

start_point	
end_point	

Definition at line 74 of file compute.cpp.

4.1.3.2 compute_one_projection()

```
void Compute::compute_one_projection (
            Point & direction_vector,
            Point & current_point )  [private]
```

Compute all projections and save save in temp_projection.

Parameters

direction_vector	
current_point	

Definition at line 65 of file compute.cpp.

4.1.3.3 compute_projections()

```
void Compute::compute_projections (
            std::vector< Point > & line )  [private]
```

Compute all projections and save save in projections.

Parameters

line	

Definition at line 39 of file compute.cpp.

4.1.3.4 display_projections()

```
void Compute::display_projections ( )
```

displays found values

Definition at line 24 of file compute.cpp.

4.1.3.5 get_points_and_input()

```
void Compute::get_points_and_input (
            std::vector< Point > & line,
            Point & input_point )
```

Get the points and input object. After it, This method calls compute_projections.

Parameters

line	input line (sequence of points).
input_point	he input point that is projected onto the line.

Definition at line 19 of file compute.cpp.

4.1.4 Field Documentation

4.1.4.1 current_parameter

```
double Compute::current_parameter  [private]
```

current parameter(use in methods).

Definition at line 80 of file compute.h.

4.1.4.2 input_point

```
Point Compute::input_point  [private]
```

the input point that is projected onto the line.

Definition at line 68 of file compute.h.

4.1.4.3 parameters

```
std::vector<float> Compute::parameters  [private]
```

found parameters.

Definition at line 92 of file compute.h.

4.1.4.4 projections

```
std::vector<Point> Compute::projections  [private]
```

found projections.

Definition at line 98 of file compute.h.

4.1.4.5 segments

```
std::vector<unsigned int> Compute::segments  [private]
```

found numbers of segmetns.

Definition at line 86 of file compute.h.

4.1.4.6 temp_projection

```
Point Compute::temp_projection  [private]
```

temp projection (use in methods).

Definition at line 74 of file compute.h.

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

- include/compute.h
- src/compute.cpp

4.2 Point Class Reference

Point is point in 3D space (because constant DIM = 3)

`#include <point.h>`

Public Member Functions

- Point ()

  Default constructor that defines a point at the origin.
- Point (double x, double y, double z)

  The constructor defines point.
- void setPoint (double x, double y, double z)

  The method sets the coordinates of the point.
- void printPoint () const

  The method prints point.
- double dist_between ()

  Calculates the sum of the squares of the coordinates of a point.
- double dist_between (Point &right)

  Computes the distance between two input points.
- double sum_coordinates () const

  Computes sum of coordinates point.
- double & operator[ ] (const int)

  Indexing operator. It returns the x, y, z coordinate depending on the index from the range [0, 2].

Private Attributes

- double point [DIM]

  point(x, y, z)

Friends

- Point operator- (const Point &, const Point &)

  The operator calculates a point that is the difference between all coordinates of the other two points.
- double operator∗ (const Point &, const Point &)

  The operator calculates the dot product.
- Point operator∗ (const Point &, const double)

  The operator calculates the multiplication point by double.
- Point operator/ (const Point &, const Point &)

  The operator alculates the division of the coordinates of points.
- Point operator/ (const Point &, const double)

  The operator alculates left per double number.

4.2.1 Detailed Description

Point is point in 3D space (because constant DIM = 3)

Definition at line 12 of file point.h.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Point() [1/2]

```
Point::Point ( )
```

Default constructor that defines a point at the origin.

Definition at line 17 of file point.cpp.

4.2.2.2 Point() [2/2]

```
Point::Point (
            double x = 0,
            double y = 0,
            double z = 0 )
```

The constructor defines point.

Parameters

x,y,z	are coordinates of input point.

Definition at line 23 of file point.cpp.

4.2.3 Member Function Documentation

4.2.3.1 dist_between() [1/2]

```
double Point::dist_between ( )
```

Calculates the sum of the squares of the coordinates of a point.

Returns

the sum of the squares of the coordinates of a point

Definition at line 39 of file point.cpp.

4.2.3.2 dist_between() [2/2]

```
double Point::dist_between (
            Point & right )
```

Computes the distance between two input points.

Parameters

right	second point

Returns

distance between this point and the second point

Definition at line 43 of file point.cpp.

4.2.3.3 operator[]()

```
double & Point::operator[] (
            const int index )
```

Indexing operator. It returns the x, y, z coordinate depending on the index from the range [0, 2].

Parameters

index	0 – x, 1 – y, 2 – z.

Definition at line 82 of file point.cpp.

4.2.3.4 printPoint()

```
void Point::printPoint ( ) const
```

The method prints point.

Parameters

left	left operand.
right	right operand.

Definition at line 35 of file point.cpp.

4.2.3.5 setPoint()

```
void Point::setPoint (
            double x = 0,
            double y = 0,
            double z = 0 )
```

The method sets the coordinates of the point.

Parameters

x,y,z	are coordinates of input.

Definition at line 29 of file point.cpp.

4.2.3.6 sum_coordinates()

```
double Point::sum_coordinates ( ) const
```

Computes sum of coordinates point.

Returns

sum of coordinates point.

Definition at line 49 of file point.cpp.

4.2.4 Friends And Related Function Documentation

4.2.4.1 operator* [1/2]

```
Point operator* (
            const Point & left,
            const double right )  [friend]
```

The operator calculates the multiplication point by double.

Parameters

left	left operand (Point).
right	right operand (double).

Definition at line 67 of file point.cpp.

4.2.4.2 operator* [2/2]

```
double operator* (
            const Point & left,
            const Point & right )  [friend]
```

The operator calculates the dot product.

Parameters

left	left operand (Point).
right	right operand (Point).

Returns

dot product.

Definition at line 61 of file point.cpp.

4.2.4.3 operator-

```
Point operator- (
            const Point & left,
            const Point & right )  [friend]
```

The operator calculates a point that is the difference between all coordinates of the other two points.

Parameters

left	left operand.
right	right operand.

Definition at line 56 of file point.cpp.

4.2.4.4 operator/ [1/2]

```
Point operator/ (
            const Point & left,
            const double right )  [friend]
```

The operator alculates left per double number.

Parameters

left	left operand (Point).
right	right operand (double).

Definition at line 77 of file point.cpp.

4.2.4.5 operator/ [2/2]

```
Point operator/ (
            const Point & left,
            const Point & right )  [friend]
```

The operator alculates the division of the coordinates of points.

Parameters

left	left operand (Point).
right	right operand (Point).

Definition at line 71 of file point.cpp.

4.2.5 Field Documentation

4.2.5.1 point

```
double Point::point[DIM]  [private]
```

point(x, y, z)

Definition at line 106 of file point.h.

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

- include/point.h
- src/point.cpp

4.3 Read_Data Class Reference

Read_Data class interface.

```
#include <read_data.h>
```

Public Member Functions

- Read_Data (const std::string &namefile)
  Construct a new Read_Data object.
- void open (const std::string &namefile)
  open ifstream.
- void close ()
  close if stream.
- void read_to_line (std::vector< Point > &line)
  read point from file and save in line

Private Attributes

- std::ifstream file

4.3.1 Detailed Description

Read_Data class interface.

Read line from file.

Definition at line 16 of file read_data.h.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 Read_Data()

```
Read_Data::Read_Data (
            const std::string & namefile )
```

Construct a new Read_Data object.

Parameters

namefile	file name with points

Definition at line 10 of file read_data.cpp.

4.3.3 Member Function Documentation

4.3.3.1 close()

```
void Read_Data::close ( )
```

close if stream.

Definition at line 26 of file read_data.cpp.

4.3.3.2 open()

```
void Read_Data::open (
            const std::string & namefile )
```

open ifstream.

Parameters

namefile	file name with points.

Definition at line 17 of file read_data.cpp.

4.3.3.3 read_to_line()

```
void Read_Data::read_to_line (
            std::vector< Point > & line )
```

read point from file and save in line

Parameters

line	input line (sequence of points).

Definition at line 31 of file read_data.cpp.

4.3.4 Field Documentation

4.3.4.1 file

```
std::ifstream Read_Data::file  [private]
```

Definition at line 45 of file read_data.h.

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

- include/read_data.h
- src/read_data.cpp

Chapter 5

File Documentation

5.1 include/compute.h File Reference

Compute class interface.

```
#include "point.h"
#include <vector>
```

Data Structures

- class Compute

  Basic computing class.

5.1.1 Detailed Description

Compute class interface.

5.2 include/consts.h File Reference

Constants.

5.2.1 Detailed Description

Constants.

5.3 include/point.h File Reference

Point class interface.

```
#include "consts.h"
```

Data Structures

- class Point

  Point is point in 3D space (because constant DIM = 3)

5.3.1 Detailed Description

Point class interface.

5.4 include/read_data.h File Reference

```
#include <string>
#include <fstream>
#include <vector>
#include "point.h"
```

Data Structures

- class Read_Data

  Read_Data class interface.

5.5 src/compute.cpp File Reference

Implementing the Compute interface.

```
#include "compute.h"
#include "consts.h"
#include <cmath>
#include <iostream>
#include <float.h>
```

5.5.1 Detailed Description

Implementing the Compute interface.

5.6 src/main.cpp File Reference

```
#include "point.h"
#include "read_data.h"
#include "compute.h"
#include <vector>
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>
```

Functions

- int main (int argc, char ∗argv[ ])

5.6.1 Function Documentation

5.6.1.1 main()

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

Definition at line 64 of file main.cpp.

5.7 src/point.cpp File Reference

Implementing the Point interface.

```
#include "point.h"
#include <stdexcept>
#include <iostream>
#include <cmath>
```

Functions

- Point operator- (const Point &left, const Point &right)
- double operator∗ (const Point &left, const Point &right)
- Point operator∗ (const Point &left, const double right)
- Point operator/ (const Point &left, const Point &right)
- Point operator/ (const Point &left, const double right)

5.7.1 Detailed Description

Implementing the Point interface.

5.7.2 Function Documentation

5.7.2.1 operator∗() [1/2]

```
Point operator* (
            const Point & left,
            const double right )
```

Parameters

left	left operand (Point).
right	right operand (double).

Definition at line 67 of file point.cpp.

5.7.2.2 operator*() [2/2]

```
double operator* (
            const Point & left,
            const Point & right )
```

Parameters

left	left operand (Point).
right	right operand (Point).

Returns

dot product.

Definition at line 61 of file point.cpp.

5.7.2.3 operator-()

```
Point operator- (
            const Point & left,
            const Point & right )
```

Parameters

left	left operand.
right	right operand.

Definition at line 56 of file point.cpp.

5.7.2.4 operator/() [1/2]

```
Point operator/ (
            const Point & left,
            const double right )
```

Parameters

left	left operand (Point).
right	right operand (double).

Definition at line 77 of file point.cpp.

5.7.2.5 operator/() [2/2]

```
Point operator/ (
            const Point & left,
            const Point & right )
```

Parameters

left	left operand (Point).
right	right operand (Point).

Definition at line 71 of file point.cpp.

5.8 src/read_data.cpp File Reference

Implementing the Compute interface.

```
#include <read_data.h>
#include <stdexcept>
#include "point.h"
#include "consts.h"
```

5.8.1 Detailed Description

Implementing the Compute interface.