Numerical Integration - Program Description

Compiling the program

In the file of your choice, you can define the main characteristics of the numerical integration and write the functions you want to integrate like that:

Step 1: define the function to integrate using vector:

```
double myFunction(vector<double> x) { return [write your function]; }
```

Step 2: define the 3 input objects:

Function, using pointer to the method of Step 1:

```
double (*functionPointer)(vector<double>);
functionPointer = &myFunction;
FunctionRntoR function(functionPointer);
```

Domain. Either 1D or n-dimension:

```
DomainCartesian domain1D(a, b, n); // integration from a to b, n subdivisions for 1 dimension
// or
DomainCartesian domainND(boundaries, subdivision); // boundaries; a vector<vector<double>>, subdivision; a vector<int> for N dimensions
```

Method. Either “Midpoint”, “Trapz” or “Simpson”:

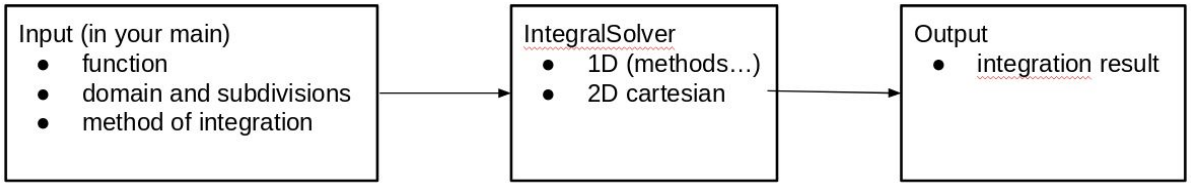
```
Method MethodOfIntegration([Name of integration method]);
```

Step 3: Solve the integral. Either 1D or 2D:

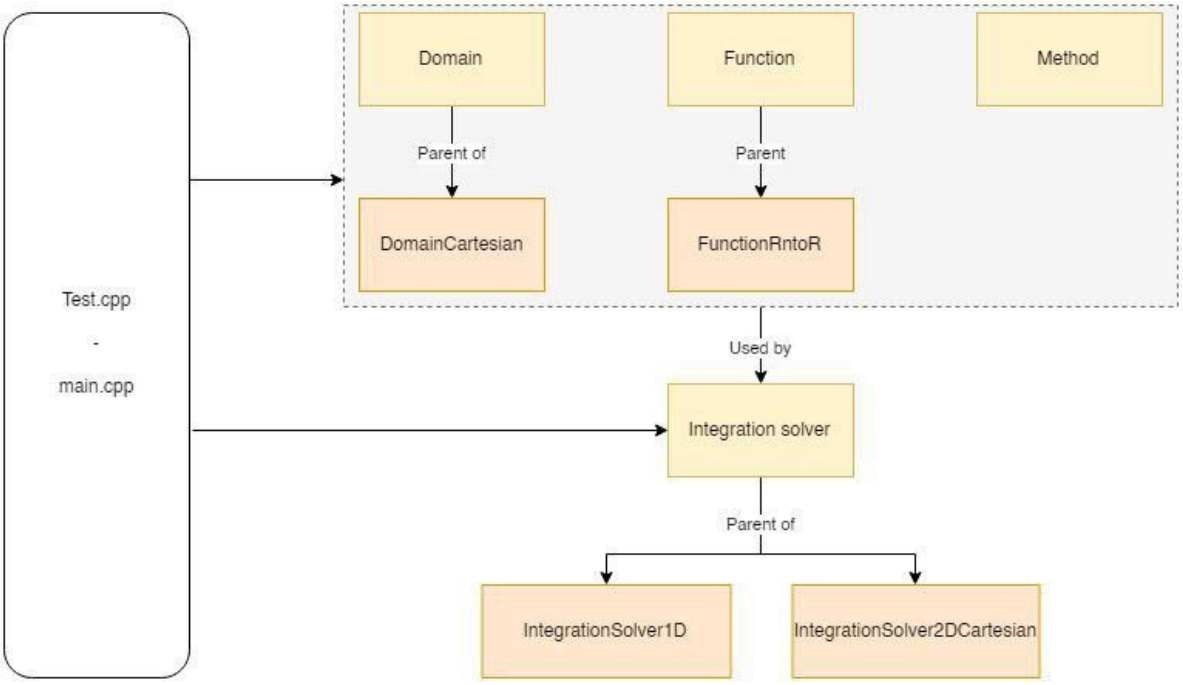
```
IntegrationSolver Integration(function, domain1D, MethodOfIntegration);
// or
IntegrationSolver2DCartesian Integration(function, domain2D, MethodOfIntegration);

double IntegrationResult = Integration.GetResult(); // Printing results
```

Flow and usage

General flow of the code:



Below a detailed diagram describing the program classes and their connections:



Features

The main features of this program are: 3 different numerical integration methods; the midpoint method, the trapeze method and the Simpson method in 1D and 2D for Cartesian planes. You can also choose the number of subdivisions in the integration.

Tests

We tested the accuracy of the integration methods by comparing the analytical values to the numerical values of:

$$\int_0^5 x^2 dx = 125/3 \text{ and } \int_0^5 \int_0^5 (x+y)dxdy = 125$$

We could verify that the results were less than 0.01 different to the analytical solutions with $n = 100$, n and n^2 subdivisions except for the 2D Simpson method: We got an error of 0.207 (if we increase the number of subdivisions, this error decreases and converges slowly so the error is probably due to a mathematical issue of the algorithm implemented).
We judge these errors as acceptable and the numerical integration methods as working.

Issues and perspectives

The structure of the code is made with such as it is easily expandable. It is possible to create new type of domains (e.g triangular), functions (e.g complex), and integration solver method (e.g Romberg).

However, some details need to be corrected to be fully expandable and flexible, such as a test only valid with double in GetResult of the parent class IntegrationSolver, or the necessity to externalize the integration methods as member classes of IntegrationSolver, and not member functions.