Introduction to Scientific Computing

Indian Institute of Technology, Madras

Assignment 3

Maximum Marks: 100

Assigned: March 18, 2024
Deadline: **March 31**, 2024

General Instructions

- You are expected to use the VM for this assignment. Create a directory in your home directory called `assignment_3`. Use this directory to work on the assignment.
- For each question (for question i), create a bash file called `question_i.sh` in the `assignment_3` directory. This bash file should contain the necessary code or commands to solve the respective question.
- We will be using an evaluation script to assess and evaluate your submission. Therefore, kindly ensure that the naming convention (as mentioned in usage section of each question) is strictly adhered to, and that the output which you get from running a script, matches the structure of the sample output.
- For submission, upload the MD5 checksum of the `assignment_3` directory on Moodle. You can use the following command. Make sure that you are in `assignment_3` directory for this command to work as intended.

```
find ./assignment_3/* -exec md5sum {} \; | cut -f 1 -d " " | md5sum
```

- After submitting the MD5 checksum on Moodle, **do not** update any file(s). Doing so will change your checksum, and your submission will not be evaluated.
- You are free to read through various resources. However, please ensure that you cite your sources to avoid plagiarism. Any detected instances of plagiarism will result in penalties.
- Please contact your assigned TA for any doubts or queries regarding this assignment.
- The **soft deadline** for this assignment is **11:59 PM** on **March 31**, 2024. Submissions after this deadline will face a linearly increasing penalty of 10 marks per late day.
- The **hard deadline** for this assignment is **11:59 PM** on **April 7**, 2024. Submissions after this deadline will not be evaluated.

[100 marks] 1. The most common use of `Makefile` is to manage dependencies of source files of programs during compilation while the programs are being compiled.

In large C/C++ projects, rebuilding the program can take significant time and effort during the compilation and linking of source files. Manually performing this task is error-prone and tedious. `Make` tracks dependencies, compiles, and links the source files *automatically* during the "build phase", and keeps track of changes, thereby recompiling programs only when necessary during the "test phase".

In this assignment, you are provided a C++ library and are tasked with

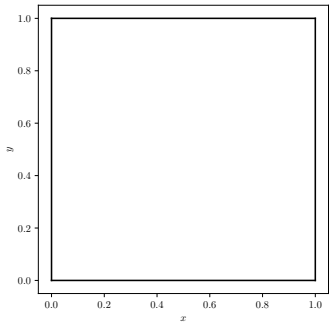
[40 marks] (a) completing functions in the library,

[60 marks] (b) creating a `Makefile` to compile and return an executable.

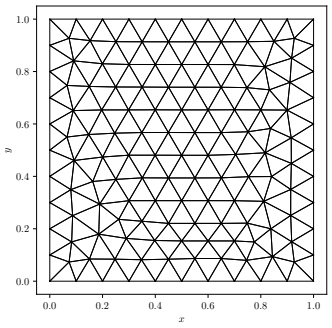
Note: This assignment does not require you to possess prior knowledge about C++, about Meshing, or any numerical technique.

Meshing is a technique commonly used in numerical methods like Finite Element Analysis (FEA) or in Computational Fluid Dynamics (CFD). Meshing involves discretizing a continuous domain Ω over which a function f needs to be computed.

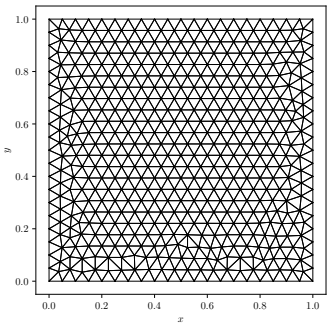
For example, consider a simple 2-dimensional domain $\Omega = [0, 1]^2$ (Fig. 1a) over which the function $f(x, y) = \sin(\pi x)\sin(\pi y)$ has to be evaluated. The first step is to mesh the domain Ω into smaller regions (Fig. 1c, 1b), followed by computing f as a piece-wise linear approximation over the domain. Figure 1d is the Analytical f over the domain Ω. Figures 1f and 1e are Numerical forms of f computed over the domain Ω for mesh_sizes 0.1 and 0.05 respectively. It is easy to verify that a finer mesh yields a more accurate solution.



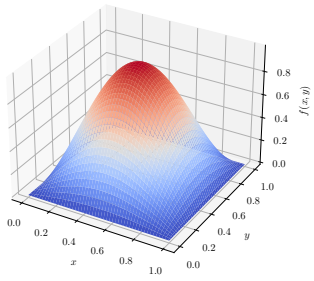
(a) Domain $\Omega = [0, 1]^2$



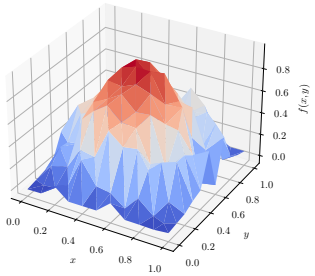
(b) Meshed Domain (mesh_size = 0.1)



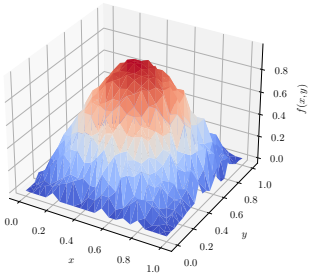
(c) Meshed Domain (mesh_size = 0.05)



(d) Analytical Solution



(e) Numerical Solution (mesh_size = 0.1)



(f) Numerical Solution (mesh_size = 0.05)

Meshing is an effective strategy for computing f, especially for complex geometries for which analytical expression for the boundaries cannot be established. It is trivial that for $f = 1$, the method returns the area of the domain.

This strategy can also be extended to solving differential equations over domains Ω. For instance, solving the Navier-Stoke's Equation in Fluid Mechanics or the Constitutive Equation in Solid Mechanics employ this strategy.

In this assignment, you are tasked to find the area of IIT Madras[1] using a custom library written in C++ (Program files are available in `/var/home/Jan24/assignments/assignment_3`).

[40 marks] (a) The majority of the implementation has already been done. However, a few functions are not fully implemented. The first part of the assignment requires you to locate partially implemented functions and complete them. (These functions are explicitly marked with comments that start with "TO DO". You may use a combination of `find`, `cat` and `grep` to locate them).

The library is structured as

```
assignment_3
├── geometry
│   ├── point.h & point.cpp
│   ├── line.h & line.cpp
│   └── triangle.h & triangle.cpp
├── main.cpp
└── map.txt
```

- `point` file contains the necessary constructs to represent a `Point(x, y)`,
- `line` file contains the necessary constructs to represent a line segment that connects `Point p1` and `Point p2`,
- `triangle` file contains the necessary constructs to represent a triangle formed by line segments `Line(p1, p2)`, `Line(p2, p3)` and `Line(p3, p1)`,
- `main.cpp` file contains the scripts to read the mesh data from a `.txt` file and also to compute the area enclosed by a triangle,
- `map.txt`[2] is a space separated file containing the coordinates of the vertices of the triangulated mesh (Fig. 2b).

Note: This assignment does not require you to possess prior knowledge about C++ (or its syntax). The sub-part (a) requires you to use only the complete logic of the code and not syntax.

Hint: Given any domain, its mesh (triangulated), the area enclosed by the domain can be computed as

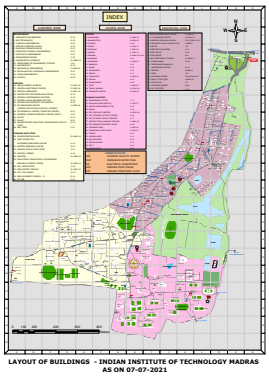
$$\text{Area of Domain} = \sum_{t \in \text{Mesh}} \text{Area of Triangle } t.$$

The Area of Triangle t can be computed using Heron's Formula from its vertices (x_j, y_j), $j \in \{1, 2, 3\}$.

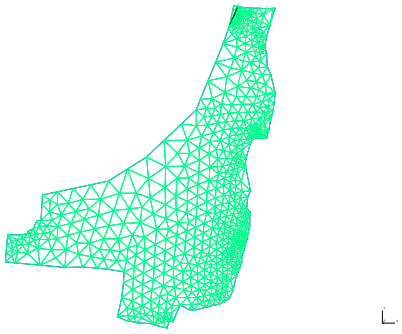
[60 marks] (b) Since the library is made of multiple C++ files, compiling them requires more than one terminal command. Create a `Makefile` in the directory that identifies all `.cpp` files, compiles and links them, and creates an executable for `main.cpp` as `main`. The `Makefile` should have the following functions

- `make`: Compile and create the executables,
- `make clean`: Remove the executables.

[1] Inspired from ME5204: Finite Element Analysis (Jul-Nov 23)
[2] Obtained using AutoCad and meshed using GMSH



(a) Scale: 800 m = 1.7094 units



(b) Meshed Map (mesh_size = 0.5)

Figure 2: Map of IIT Madras

Input:

```
map.txt
n
x_11 y_11 x_12 y_12 x_13 y_13
x_21 y_21 x_22 y_22 x_23 y_23
```

and so on ... where `(x_ij, y_ij)` represents the x-coordinate and the y-coordinate of the j^{th} vertex ($j \in \{1, 2, 3\}$) of the i^{th} triangle in the mesh and n is the total number of points.

Usage:

```
make && ./main map.txt
```

Output: (in m^2)

```
2479646.4320
```

Note: Actual reported area of IIT Madras around 2500000 m^2.