University of Stuttgart
Germany

Developing a parsing algorithm for OpenCMISS to setup a generic simulation based on input file

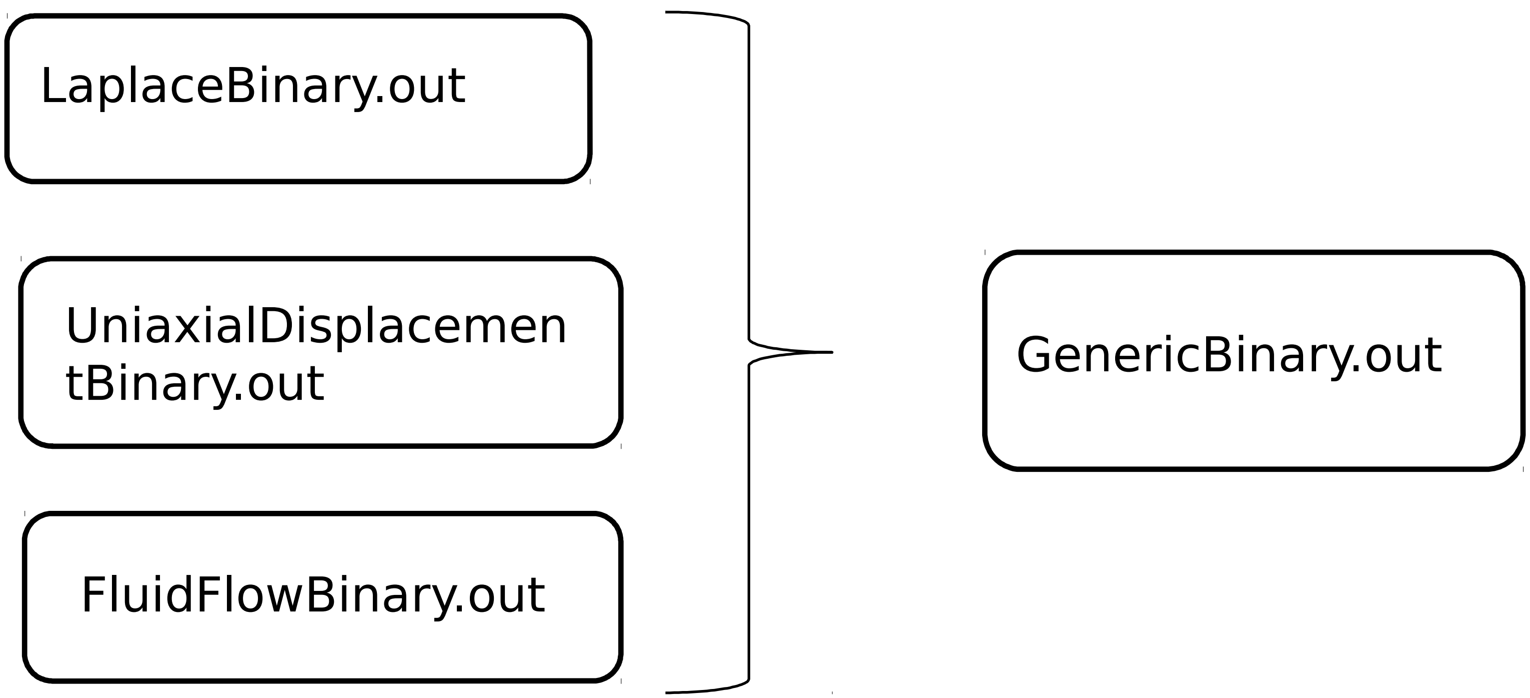
Waleed Mirza

Road Map Of Presentation

- State of the art.
- Motivation / Advantages.
- Syntax and functionalities of the input file.
- Different aspects of parsing algorithm.
- Case studies
- Future Work

State of the art

- Different binary for every case study.
- For each set of simulation parameters, one has to recompile the source code.

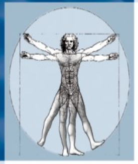


Motivation/Advantages

The developed algorithm will allow user to,

- Set up a generic simulation in OpenCMISS by providing instructions in an input file *(just like .inp file in Abaqus or .ans file in Ansys).*
- To change the simulation parameters (for instance in parametric analysis) without recompiling the code.
- Given a binary file (for instance ***GenericCaseStudy.out***) and the input file (*for instance* ***input.iron***) , a simulation can be executed by,

$ <Absolute path to the binary file >/ *GenericCaseStudy*
<Absolute path to the input file file >/ input.iron

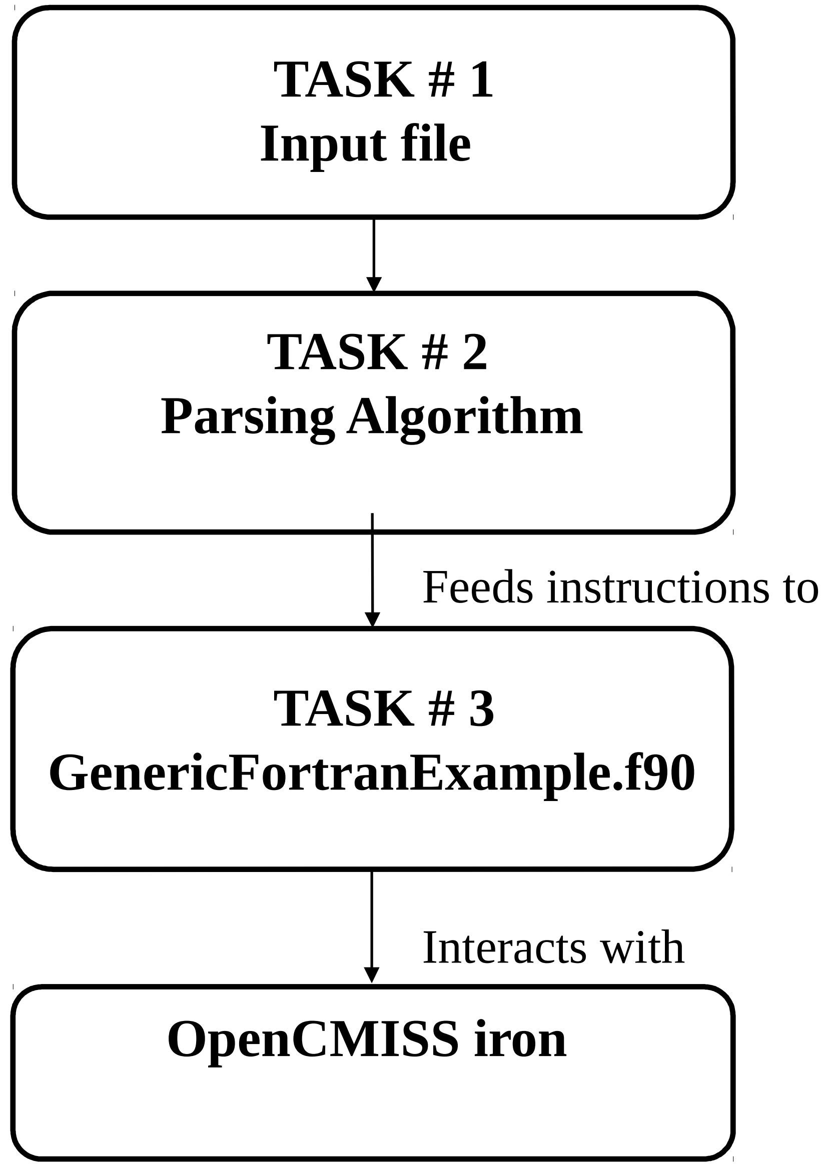


Project overview

- 1- Develop layout/structure for the input file.

- 2- Develop a *GenericFortranExample.f90* file capable of setting up a generic simulation.

- 3- Develop a parsing algorithm that can pick input instructions from the input file and feeds them in the *GenericFortranExample.f90* file



1- Preliminary Concepts

Structure of OpenCMISS involves objects of different derived types. Each derived type object contains different simulation parameters. For instance:

Objects of ***CMFE_GeneratedMesh*** Type contain information of mesh parameters such as mesh size , topology etc.

- Objects of ***CMFE_BoundaryConditions*** Type contains information of the boundary conditions such as location of BCs, prescribed values etc.
- With instructions provided in the input file, these derived types objects are created and appropriate information is stored in them.

1- Preliminary Concepts (continued)

Type (GeneratedMeshStructure) ::
SolidMesh

Type (CMFE_GeneratedMeshStructure) ::
GeneratedMesh

Type (CMISSINTEGER) ::
GeneratedMeshType

Type (CMISSINTEGER) ::
GeneratedMeshNumberOfElements(3)

Structure / Class
GeneratedMeshStructure

Type (CMFE_GeneratedMeshStructure) ::
GeneratedMesh

Type (CMISSINTEGER) ::
GeneratedMeshType

Type (CMISSINTEGER) ::
GeneratedMeshNumberOfElements(3)

Type (GeneratedMeshStructure) ::
FluidMesh

2- Syntax of the input file

- The input file is divided in 18 blocks and each block generates a derived type object and contains information which is later stored in different members of the object.

- BASIS block
- GENERATED_MESH Block
- BOUNDARY_CONDITION Block
- CONTROL_LOOP block
- MATERIAL_FIELD block
- DEPENDENT_FIELD block etc.

- *Note that each block corresponds to block of routines in FotranExample.f90 file*

2 - Syntax of Input file (Continued)

- Each block starts and ends with the keywords *START_<BLOCK NAME >*and *END_<BLOCK NAME>* respectively.
- Each block encapsulate set of input arguments.

```
START_BASIS

BASIS_ID
FLUID

NumberOfGaussXi      ! For numerical integration
3   ,   3

BASIS_INTERPOLATION_TYPE
LINEAR_LAGRANGE_INTERPOLATION

END_BASIS
```

2 - Syntax of Input file (Continued)

- The syntax of input is case insensitive.
- Comments should be started with exclamation mark *!*
- *Blank line can be introduced by the user in the input file for readability.*
- *The input file should be finished with the keyword STOP_PARSING*
- **Please be careful with the spellings . As of now there is no way to deal with typos.**

2 - Syntax of Input file (Continued)
Example of an input file

Governing equation to be solved for domain with volume *1m*x*1m* x *1m*.

$$\nabla^2 \varphi = 0$$

- **Case *a***

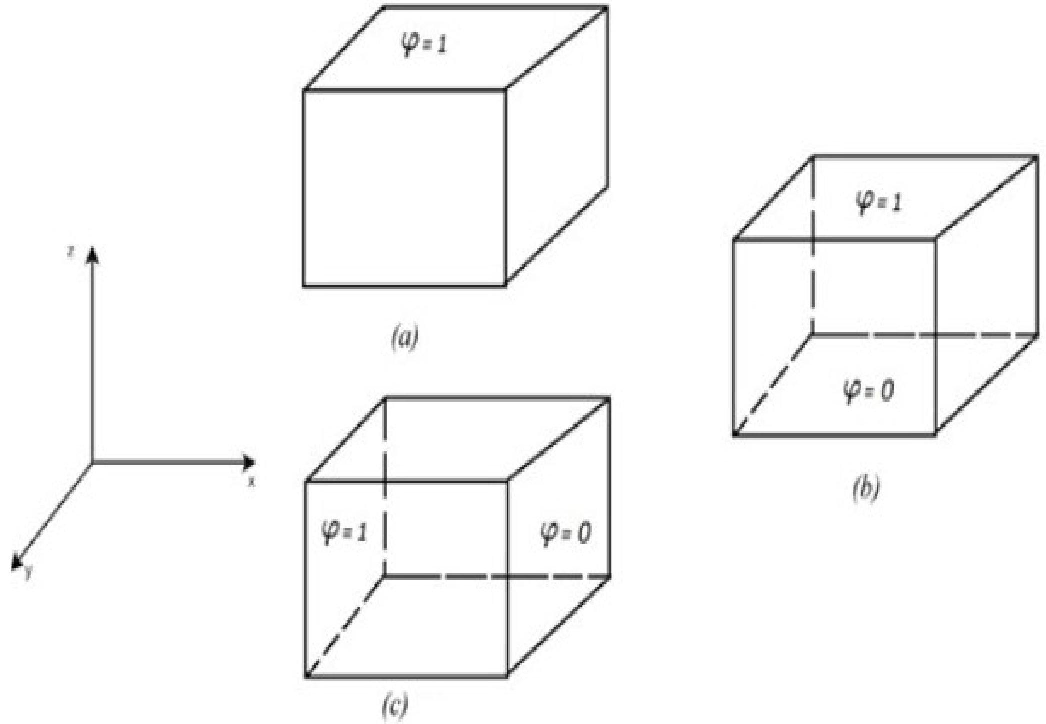
$$\varphi(x, y, 1) = 1$$

- **Case *b***

$$\varphi(0, y, z) = 1 \qquad \varphi(1, y, z) = 0$$

- ***Case c***

$$\varphi(x, y, 1) = 1 \qquad \varphi(x, y, 0) = 0$$



University of Stuttgart
Germany

3- Parsing Algorithm

3.1- Nomenclature of data structures

- The nomenclature of the data structures have been established with an aim to make them **self descriptive** and **clear in terms of readability.**
- For instance in

  all_Basis(:)%BasisInterpolationType

  all_Basis(:) is an array of objects of generated mesh.

- Size of the array = number of times the *Basis* block defined in the input file.
- For example for an FSI study there will be three
- GENERATED_MESH blocks i.e. for SOLID , FLUID and
- INTERFACE.”

3.1- Nomenclature Of Data Structures (Conti.)

- ***BasisInterpolationType*** is a string type member of object ***all_Basis(:)***.
- For instance, in an FSI study each domain (Solid, Fluid and Interface) can have a different Interpolation type

all_Basis(1)%BasisInterpolationType = "QUADRATIC_LAGRANGE_INTERPOLATION"

all_Basis(2)%BasisInterpolationType = "QUADRATIC_LAGRANGE_INTERPOLATION"

all_Basis(3)%BasisInterpolationType = "LINEAR_LAGRANGE_INTERPOLATION"

3.1- Nomenclature Of Data Structures (Conti.)

- **Question:** In a multidomain problem how will the algorithm recognize which interpolation type belongs to which domain ?

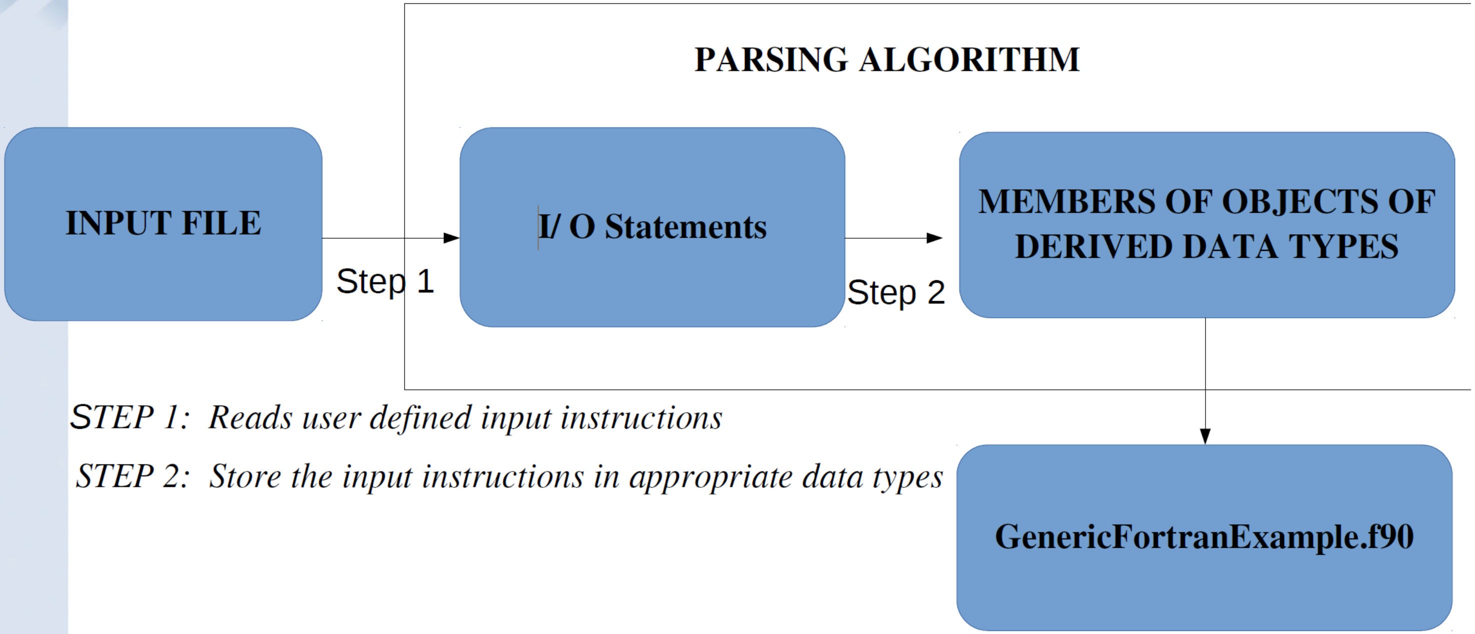
- **Answer:** Block Ids will help algorithm with that.

all_Basis(1)%BasisId = "SOLID"
all_Basis(2)%BasisId = "FLUID"
all_Basis(3)%BasisId = "INTERFACE"

- Section 4 will explain this feature of the algorithm more in detail.

3.1- Nomenclature Of Data Structures (Conti.)

- In a Nutshell



STEP 1: Reads user defined input instructions

STEP 2: Store the input instructions in appropriate data types

4- GenericFortranExampleFile.f90

- This is the file where simulation is setup using the parameters define in the input file.
- As of now, the *GenericFortranExampleFile.f90* is evolved enough to setup laplace , fluid and solid mechanics problems.
- Nevertheless, *GenericFortranExampleFile.f90* is quite amenable and modifiable for simulations of other classes.

4- GenericFortranExampleFile.f90 (Continued)

- Objects of different data type are linked here. For instance
- Objects of *all_Region(:)%Region* and *all_CoordinateSystem(:)%CoordinateSystem* with similar data types are linked together i.e.

CALL cmfe_Region_CoordinateSystemSet(*all_Region(RegionIdx)%Region*,
& *all_CoordinateSystem(CoordinateSystemIdx)%CoordinateSystem*,Err)

- The Algorithm has to make sure in the subroutine above , *CoordinateSystemIdx* and *RegionIdx* are such that,
- *all_CoordinateSystem(CoordinateSystemIdx)&*

%CordinateSystemId = “*SOLID*”
all_Region(RegionIdx)%RegionId = “*SOLID*”

4- GenericFortranExampleFile.f90

- Thats where the ***MATCH_ID( )*** subroutine kicks in
- call subroutine MATCH_ID(all_Region(RegionIdx)%RegionId, &
- & all_CoordinateSystem(:)%CoordinateSystemId,
- & CoordinateSystemIdx)

all_Region(RegionIdx)
%RegionId = “FLUID”

all_CoordinateSystem(1)
%CoordinateSystemId =
“SOLID”

all_CoordinateSystem(2)
%CoordinateSystemId =
“FLUID”

all_CoordinateSystem(3)
%CoordinateSystemId =
“INTERFACE”

Answer:
CoordinateSystemIdx = 2

4- Case Studies

4.1 Laplace Problem

Governing equation to be solved for domain with volume *1m*x*1m* x *1m.*

$$\nabla^2 \varphi = 0$$

- **Case *a***

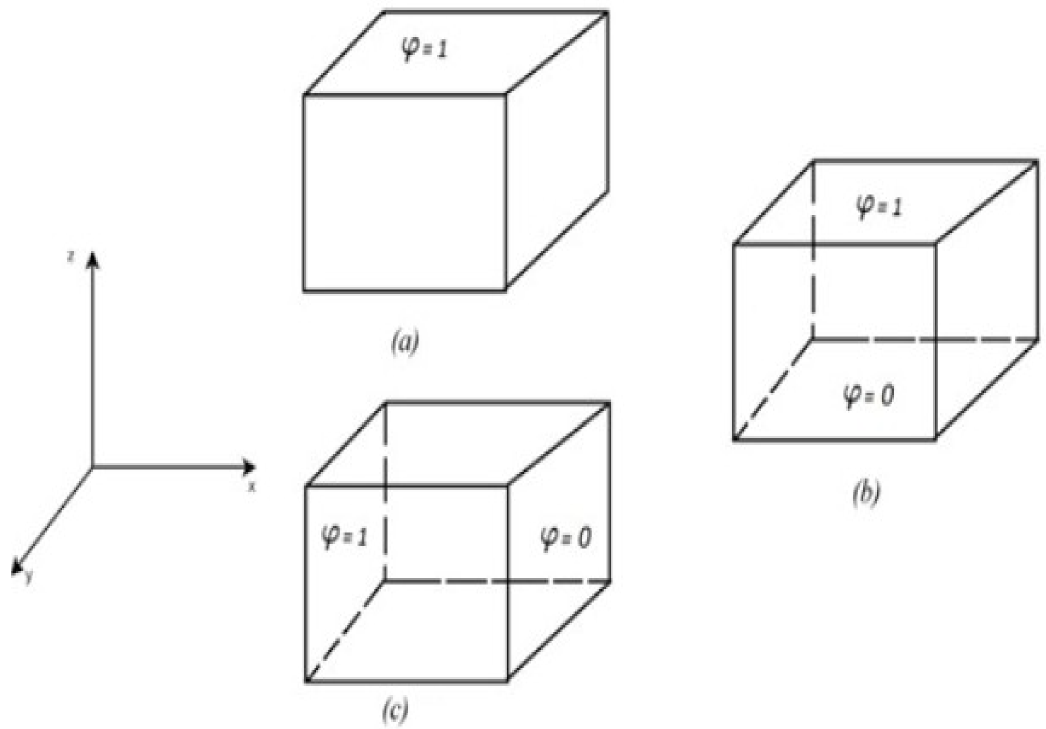
$$\varphi(x, y, 1) = 1$$

- **Case *b***

$$\varphi(x, y, 1) = 1 \qquad \varphi(x, y, 0) = 0$$

- ***Case c***

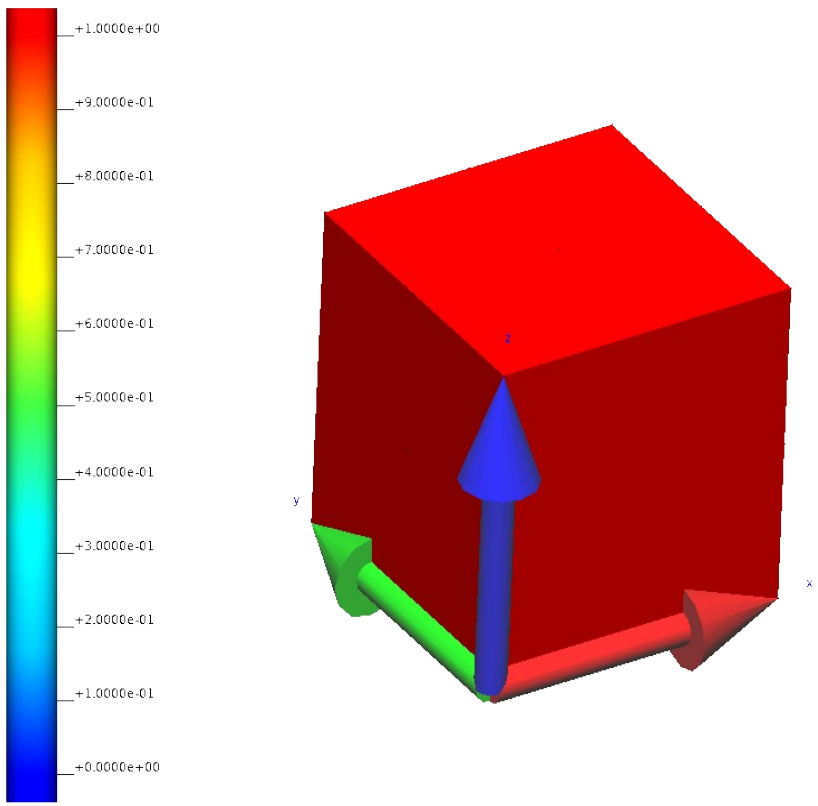
$$\varphi(0, y, z) = 1 \qquad \varphi(1, y, z) = 0$$

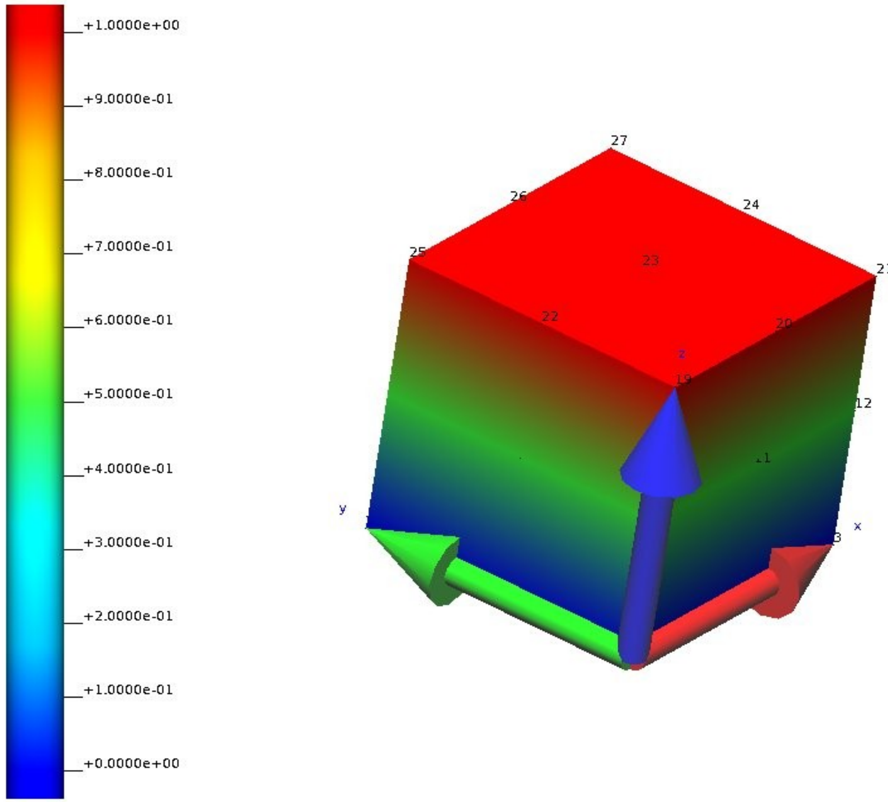


4.1- Laplace Problem (continued)

Case (a): $\varphi(x, y, 1) = 1$

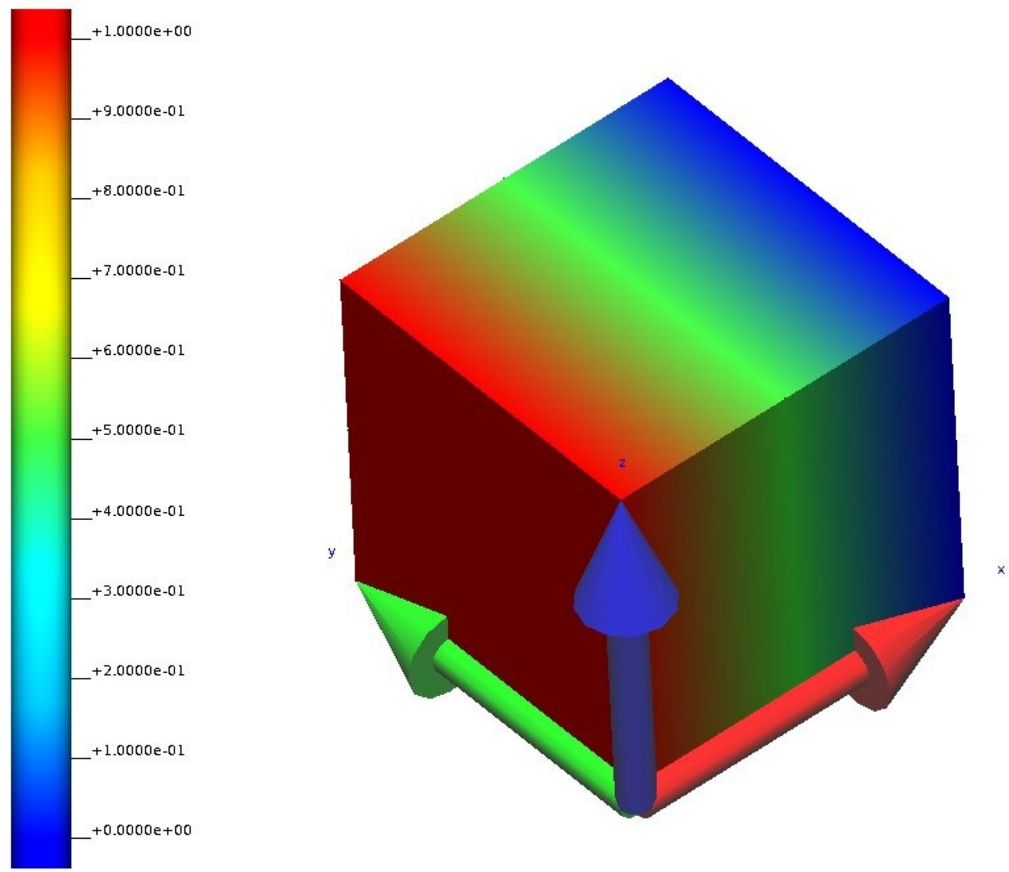
Case (b): $\varphi(x, y, 1) = 1 \quad \varphi(x, y, 0) = 0$





4.1- Laplace Problem (continued)

Case (c): $\varphi(0, y, z) = 1 \quad \varphi(1, y, z) = 0$



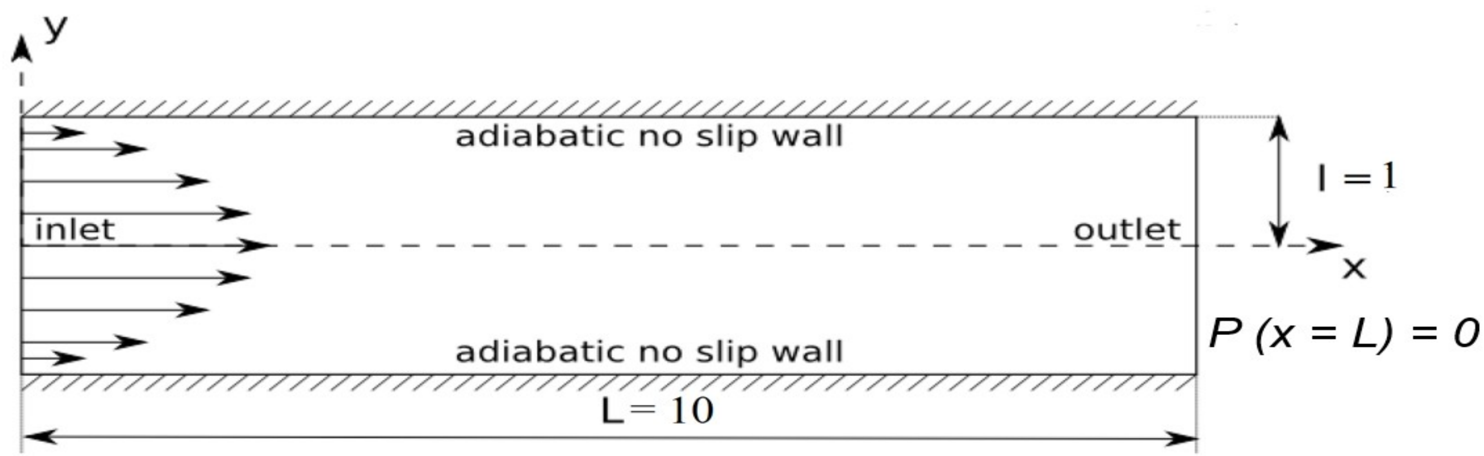
4.2 Case Study # 2
2D Fluid Flow Problem

Governing equation to be solved

$$\underbrace{\rho\left(\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u}\right)}_{\text{Acceleration}} = \underbrace{-\nabla p}_{\text{Pressure}} + \underbrace{\nu \triangle \vec{u}}_{\text{Viscosity}}$$

$$Case(a) : V_x(0, y) = 4$$

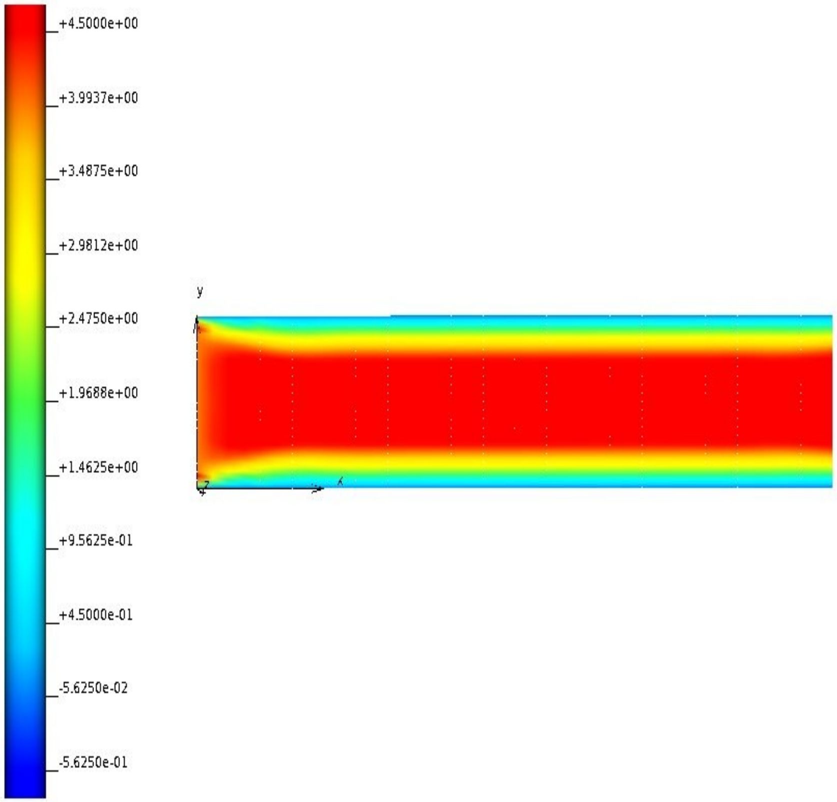
$$Case(b) : V_x(0, y) = 4V_{max}\frac{y}{h}\left(\frac{y}{h} - 1\right)$$

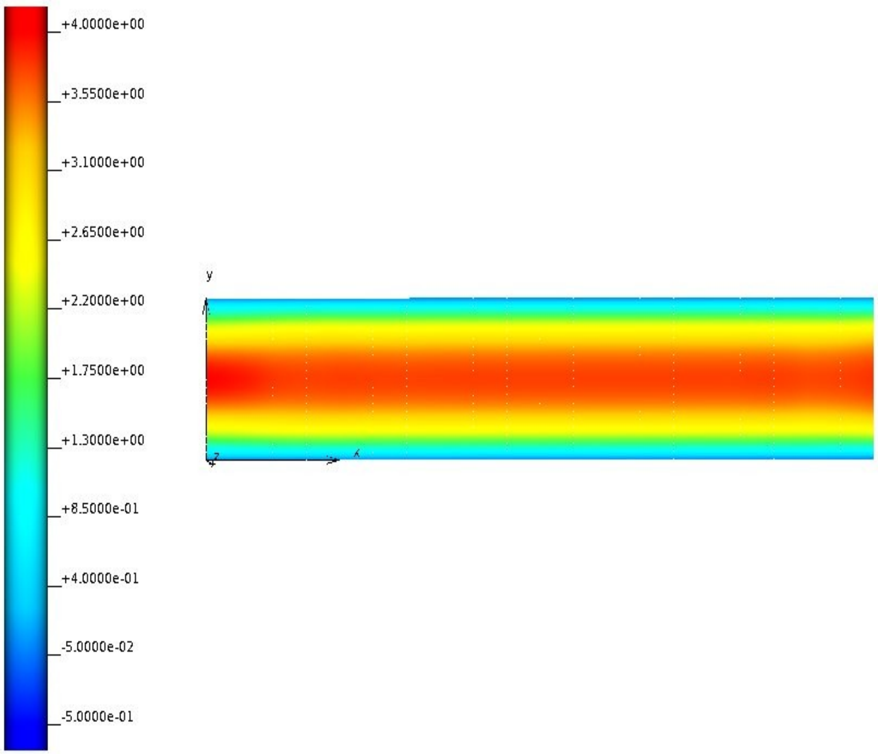


2d Fluid Flow Problem (Conti.)

$$Case(a) : V_x(0, y) = 4$$

$$Case(b) : V_x(0, y) = 4V_{max}\frac{y}{h}(\frac{y}{h} - 1)$$





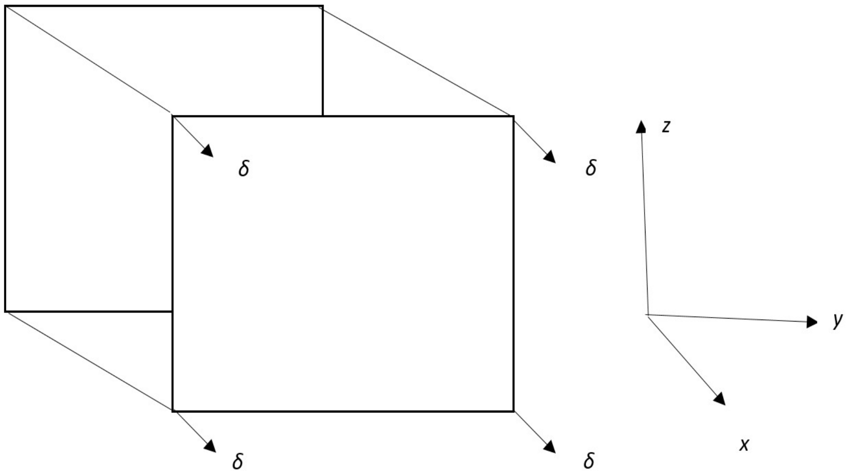
4.3 - Case study # 3
Cube subjected to uniaxial displacement

Governing equation to be solved

$$0 = \frac{1}{\rho}\Delta \cdot \sigma + g$$

Boundary conditions

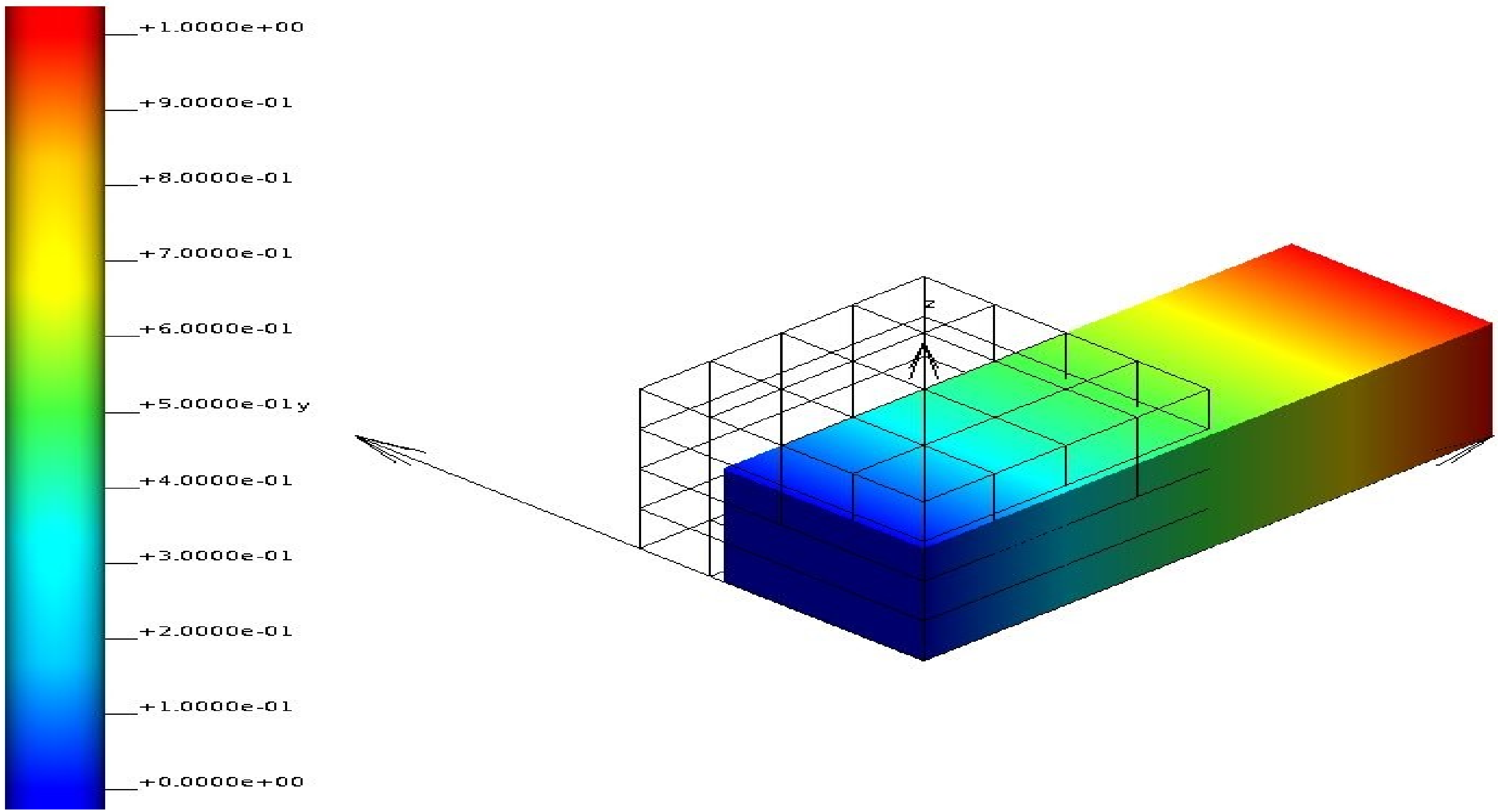
$$u(0, y, z) = 0$$
$$u(1, y, z) = 1$$
$$u(x, y, 0) = 0$$



Material model

Transverse Isotropic Material model

4.3 Cube subjected to uniaxial displacement (Continued)



5- Conclusion and Future Research

- The developed parsing algorithm can ideally solve Fluid Mechanics , Solid Mechanics and Laplace problems.
- Over the course of following months, the algorithm will be further developed and tested to solve a **fluid solid interaction study**.
- **Please talk to me if you want me to add any feature in my algorithm of your interest.**