

NUMERICAL SOLUTIONS

# FluxSol

## Version 0.1.0

User's Guide

*Luciano Buglioni*

supervised by  
Dr. Mark BROWN

April 7, 2016

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Input File</b>	<b>2</b>
<b>3</b>	<b>User Defined Objects (UDOs)</b>	<b>2</b>
3.1	UDO via Command Line . . . . .	3
3.2	UDO via Graphic User Interface . . . . .	3
3.3	UDO Usage . . . . .	3

# 1 Introduction

FluxSol is a multiplatform, opensource CFD solver written in C++. FluxSol is released under GNU GPL v.3+.

## 2 Input File

Input file consist of a plain text document which comprises several fields. Each of these fields consists of a name followed by “=” character. After this, a corresponding value of the field must to follow, ending with a “;” character. Some other fields (like “material”, “BC”, “patch”), whose values are more than one, do not contain “=” character after, they have instead certain parameters between parentheses, all of them with a parameter id followed by “=” and separated between them with a semicolon. An example of input file is presented below:

```
//Example of Input file.
grid.1 file = square.cgns; dimension = 2; solution_scheme = navier-stokes; material ( k=1.; rho=1.); //ma-
terial(dbname=air); patch.1(name=top;type=faces;list=[1,2,3]);
BC.2 (patch=BC-2;type=wall;U=[1.,0.,0.]);
BC.1 (patch=BC-4;type=wall;U=[0.,0.,0.]);
BC.3 (patch=BC-3;type=wall;U=0.0.);
BC.4 (patch=BC-1;type=wall;U=[0.,0.,0.]);
```

## 3 User Defined Objects (UDOs)

UDO (User Defined Object) represents a custom defined behavior (value, field, algorithm) which can be given to a different part (material, zone, condition) of the model. For this version it is restricted only to boundary conditions. UDO can be defined by two different ways: by CAE or via command line. In both cases a C++ code must be written and compiled in order to describe this behavior. Once it is compiled, it can be linked during runtime with the solver and thus, it can be implemented. To compile an UDO are required the following: - cmake If the operating system is Windows, it is required mingw-w64. Once the code containing UDO is written, it must be compiled. An example of a C++ code is presented below.

File “udf\_boundary.cpp”:

```
#include "udf_boundary.h"
using namespace FluxSol;
void ufield::Calculate()
{
    cout << "UDF Calculating" << endl;
    Scalar x;
    double xveloc;

    for (int f=0;f<this->_Patch().Num_Faces();f++)

    x=this->_Patch().Face(f).Center()[0]-0.5;
    xveloc=(0.25-(x.Val()*x.Val()))*100.0;
    this->value[f]=Vec3D(xveloc,0.,0.);

}
```

File “udf\_boundary.h”:

```
#include "FluxSol.h"
```

```
UD_VelocityPatchField ufield;
```

### 3.1 UDO via Command Line

It must be compiled UDO file. The way to do that is running “udogen.bat” or “udogen.sh” depending on system. There are two options: it can be run this file without arguments, or either can be assigned the name of the .cpp file, only if this is only one file. //TO MODIFY This file runs at first “CodeWriter.exe” followed by the file containing the code. This is a parsing program which creates another c++ file, named “UDOCreator.cpp”, which is compiled to obtain the shared library file. The batchfile output will ask for the confirmation of the user in order to overwrite libUDO shared library in case of successful compilation.

This is a classic output of “CodeWriter.exe”

```
[I] Reading ...
```

```
[I] Found input file ./udf_boundary.h
```

```
[I] 10 lines readed ...
```

```
(Inherited) Defined Class Name: ufield
```

```
Base Class Name: UD_VelocityPatchFieldpublic:ufield():UD_VelocityPatchField()voidCalculate()
```

```
Ending
```

```
./udf_boundary.h: 0 Including./udf_boundary.h UDO Ids size1
```

### 3.2 UDO via Graphic User Interface

To define an UDO by FluxSol GUI, it must be compiled first.

### 3.3 UDO Usage

Once compiled the UDO, it can be used by the input file. Here is an example of input file. In the example below, object compiled in the previous example is implemented in the Patch named “BC-2”. To implement an UDO, the parameter “def” followed by their value “UDO”, must be present in the patch parameter list. patch\_1 type

```
grid_1 file = Mesh_Dens_10.cgns;
```

```
solution_scheme = navier-stokes;
```

```
patch_1(type=face);
```

```
material (k=1.;rho=1.);
```

```
//Definition default is by constant value BC_1 (patch=BC-2;type=wall;def=UDO;udo=ufield); //Here is called UDO named ufield
```

```
BC_2 (patch=BC-1;type=wall;U=0.0.);
```

```
BC_3 (patch=BC-3;type=wall;U=0.0.);
```

```
BC_4 (patch=BC-4;type=wall;U=[0.,0.,0.]);
```

Figure 1: Dummy figure

Table 1: Dummy table

...

## List of Figures

1	Dummy figure . . . . .	4
---	------------------------	---

## List of Tables

1	Dummy table . . . . .	4
---	-----------------------	---