Curious About Making User Defined Functions in ANSYS Fluent?

Jens-Uwe Friemann, Technical Services, ANSYS Sweden AB
Introduction

Almost all ANSYS Fluent users will end up having to write short UDFs from time to time. You will see how this can be done easily, even with only a rudimentary understanding of the C programming language.

User-Defined Functions (UDFs) enable the customization of many aspects of a CFD model:

- boundary and cell zone conditions,
- material properties,
- data output, and
- solution execution.

Data exchange between the UDF and the solver takes place via pre-defined macros.

Today I will give you a short introduction about:
- What UDFs can do
- How to write and compile a basic UDF
- How to hook UDFs to the ANSYS Fluent interface
UDF Overview

• **What** is a User Defined Function?
  – A UDF is a function *(programmed by the user)* written in C which can be dynamically linked with the ANSYS Fluent solver.
  • **Standard C functions**
    – Trigonometric, exponential, control blocks, do-loops, file i/o, etc.
  • **Pre-Defined Macros**
    – Allow access to *field variable*, *material property*, and *cell geometry data* and include many utilities
  – All data exchanged between the UDF and the solver must be in *SI units*

• **What** do UDFs offer?
  – The **standard interface cannot supply** all needed functionalities:
    • **Customization** of boundary conditions, source terms, reaction rates, material properties, etc.
    • **Customization** of physical models
    • **Additional**, user-supplied, **model (transport) equations**
    • Adjust functions (once per iteration) for data *interaction during solution*
    • Execute on Demand functions on the **fingertips** of the user
    • Solution **Initialization**
Interpreted vs. Compiled UDFs

UDFs can either be run compiled or interpreted.

• **Interpreted** code vs. **compiled** code
  – Interpreted
    • C Interpreter **bundled with ANSYS Fluent**
    • Interpreter executes code on a “line by line” basis instantaneously.
    • **Advantage** – There is **no** need to install a **third-party compiler**.
    • **Disadvantage** – Interpreter is **slower**, and **limited** to some functions and needs a **lot of RAM**.

  – Compiled
    • UDF code is **translated once** into machine language (object modules).
    • **Efficient** way to run UDFs.
    • Creates **shared libraries** which are **linked** with the rest of the solver.
    • Does **require a compilation step** between creating/editing your UDF and using it.
    • The supported compiler for FLUENT on **Windows** platforms is **Microsoft Visual Studio**
      Most **Linux** systems provide a **C compiler** as a **standard feature**.
The cell zones and face zones of a model (in the finite-volume scheme) are accessed in UDFs as **Thread** data types.

**Thread** is a FLUENT-defined data type.

In order to access data in a thread (zone), we need to provide the correct **thread pointer**, and use ANSYS Fluent provided **looping macros** to access each member (cell or face) in that thread.
UDF Step by Step

- **The basic steps** for using UDFs in ANSYS Fluent are as follows:

  1. **Identify** the problem to be solved
  2. **Check** the **usability & limitations** of the standard models of FLUENT
  3. **Program** the UDF (must be written in C)
  4. **Compile** the UDF in the ANSYS Fluent session
  5. **Assign** (hook) the UDF to the appropriate variable and zone in BC panel
  6. **Run** the calculation
  7. **Examine** the results
Step 1 – Identify the Problem to Solve

• We would like to impose a parabolic inlet velocity to the 2D elbow shown.

• The $x$ velocity is to be specified as:

$$u(y) = 20 \left[ 1 - \left( \frac{y}{0.0745} \right)^2 \right]$$
Example 1 – Parabolic Inlet Velocity Profile

Step 2 – Prepare the Source Code

• The code is stored as a text file `inlet_bc.c`

• The `DEFINE_PROFILE` macro allows the function `x_velocity` to be defined.

  – All UDFs begin with a `DEFINE_` macro

  – The name that will appear in the GUI will be `x_velocity`

  – The macro `begin_f_loop` loops over all faces `f`, pointed by thread

  – The `F_CENTROID` macro assigns cell position vector to `pos[]`

  – The `F_PROFILE` macro applies the velocity component to face `f`, using the desired arithmetic function

Header file “`udf.h`” must be included at the top of the program by the `#include` command

```c
#include "udf.h"

DEFINE_PROFILE(x_velocity,thread,nv)
{
  float pos[3]; /* an array for the coordinates */
  float y;
  face_t f; /* f is a face thread index */

  begin_f_loop(f, thread)
  {
    F_CENTROID(pos,f,thread);
    y = pos[1];
    F_PROFILE(f, thread, nv) = 20.*(1.-
                           y*y/(.0745*.0745));
  }
  end_f_loop(f, thread)
}
```
Example 1 – Parabolic Inlet Velocity Profile

Step 3 – Compile the UDF in the ANSYS Fluent Session

- **Compiled UDF**
  - Add the UDF source code to the Source Files list.
  - Click Build to compile and link the code.
  - If no errors, click Load to load the library.
  - You can also unload a library if needed.

- **Interpreted UDF**
  - Add the UDF source code to the Source File Name list.
  - Click Interpret.
  - The assembly language code will display in the Fluent console.
  - Click Close if there is no error.
Step 4 – Hook the UDF in FLUENT GUI

- Open the boundary condition panel for the surface to which you would like to apply the UDF
- Switch from Constant to **udf x_velocity** in the drop-down list
- The macro name is the first argument of **DEFINE_PROFILE** in the UDF code
Example 1 – Parabolic Inlet Velocity Profile

Step 5 – Run the Calculations

• You can change the Profile Update Interval in the Run Calculation panel (default value is 1)
  – This setting controls how often (either iterations or time steps if unsteady) the UDF profile is updated

➢ ... Run the calculation as usual ...
Example 1 – Parabolic Inlet Velocity Profile

Step 6 – Examine the Results

• The figure on the left shows the velocity field through the 2D elbow.

• The figure on the right shows the velocity vectors at the inlet. Notice the imposed parabolic velocity profile.

\[ u(y) = 20 \left[ 1 - \left( \frac{y}{0.0745} \right)^2 \right] \]
Example 2 – Custom Initialization

- Initialize as above shown
- The **domain pointer** is passed to this UDF **through** the **argument**
- Define > User-Defined > Function Hooks

```c
#include "udf.h"

DEFINE_INIT(my_init_function, domain) {
    cell_t c;
    Thread *ct;
    real xc[ND_ND];
    thread_loop_c(ct,domain) {
        begin_c_loop (c,ct) {
            C_CENTROID(xc,c,ct);
            if (sqrt(ND_SUM(pow(xc[0]-0.5,2.),
                       pow(xc[1] - 0.5,2.),
                       pow(xc[2] - 0.5,2.))) < 0.25)
                C_T(c,ct) = 600.;
            else
                C_T(c,ct) = 300.;
        } end_c_loop (c,ct)
    }
}
```
Some DEFINE_ macros are hooked into the solver using the User-Defined Function Hooks panel

- **Initialization**
  - Executes once per initialization

- **Solution adjustment**
  - Executes every iteration

- **Wall heat flux**
  - Defines fluid-side diffusive and radiative wall heat fluxes in terms of heat transfer coefficients
  - Applies to all walls

- **User-defined surface and volumetric reactions**

- **Read/write to/from case and data files**
  - Read order and write order must be same

- **Execute-on-Demand capability**
  - Does not participate in the solver iterations
Use of Macros in the Examples

- UDFs **communicate** with the solver through **pre-defined macros**

- Macros can be loosely categorized as either
  - **DEFINE** macros
    
    - **DEFINE_PROFILE** \((x\_velocity, thread, nv)\)
    - **DEFINE_INIT** \((my\_init\_function, domain)\)
  - **Looping** macros
    
    - **thread_loop\_c** \((ct, domain)\)
    - **begin\_f\_loop** \((f, thread)\)
    - **end\_f\_loop** \((f, thread)\)
  - **Type declaration** macros
    
    - **Domain** \(*d; thread *t; cell_t c; face_t f; real x, y, z; cxbolean true_or_false;**
  - **Geometry** macros
    
    - **C\_CENTROID** \((xc, c, ct)\)
    - **F\_AREA** \((A, f, tf)\)
  - **Solution data access** macros
    
    - **C\_T** \((c, t)\)
    - **C\_R** \((c, tc)\)
    - **C\_U** \((c, tc)\)
    - **C\_V** \((c, tc)\)
    - **C\_W** \((c, tc)\)

- Still **many more** DEFINE macros are available in the following categories and are documented in the **UDF manual**:
  - **Turbulence** models
  - **Multiphase** models
  - **Reacting** flows
  - **Dynamic** mesh
  - **Input/Output**
User Defined Memory (UDM)

• While macros are available for all solution variables, for instance C_T in Example 2, sometimes it can be helpful to create additional variables
  – For instance to store value of custom source terms, or custom post-processing variables

• This can be done with User-Defined Memory variables, which are user-allocated memory for each cell
  – Up to 500 field variables can be defined
  – The UDF macros for cell values and face values, respectively, are
    • C_UDMI(cell,thread,index);
    • F_UDMI(face,thread,index);
  – Information is stored in the FLUENT data file and is accessible in post-processing operations such as contour plots
User Defined Scalars (UDS)

- FLUENT can solve **up to 50 generic transport equations** for user-defined scalars
  - The UDS equations can be solved as a standard **steady** or **transient convection-diffusion transport equation**, or they **can be customized** through UDFs, to represent other partial differential equations, for instance the electromagnetic field equations
- All terms in the equations can be controlled through UDFs

\[
\frac{\partial \phi_k}{\partial t} + \frac{\partial}{\partial x_i} \left( F_i \phi_k - \Gamma_k \frac{\partial \phi_k}{\partial x_i} \right) = S_{\phi_k} \quad k = 1, 2, \ldots, N_{\text{scalar}}
\]

- **DEFINE_UDS_UNSTEADY**
- **DEFINE_UDS_FLUX**
- **DEFINE_DIFFUSIVITY**
- **DEFINE_SOURCE**

- **This, together with the UDF macro structure, makes ANSYS Fluent to the most open commercial CFD code in the world!**
UDF applications for ANSYS Fluent

• Many of the add-on models supplied by ANSYS and third-party suppliers are programmed inside the ANSYS Fluent UDF frame using UDS:

  – Supplied by ANSYS:
    • MagnetoHydroDynamics (MHD) Module
    • Population Balance Module
    • Battery Module
    • Fuel Cell Module
    • Continues Fiber Module
    • Adjoint Solver Module

  – Third Party modules:
    • RBF Morph (Adjoint Solver), RBF Morph – Marco Biancoli
    • V2f Turbulence model for FLUENT (before integration)
    • Many connections to third party products a programmed with UDFs

So, why not implement your very own model? By only supplying the *.dll or *.so files for the UDF project you can protect your intellectual property!
UDF/UDS example: Electrostatic lens

• The **problem**: Investigate the function of a device which **controls** a **droplet flow** with an applied **electrostatic potential**

• ANSYS Fluent has **no build-in electrostatic model** (if we do not consider the MHD model)

**Positive charged** water droplets

- q = 1 C/kg
- D = 1 mm
- V\text{in} = 100 m/s
- M = 10 g/s

**Particles** are simulated with **DPM**:

**Thin air atmosphere**: 1/1000:th of normal density
UDF/UDS example: Electrostatic lens

• We need one UDS to compute the electrostatic potential:

\[ \Delta \varphi = 0 \]

— When comparing with the transport equation solved by Fluent we can achieve the above equation through

\[
\frac{\partial \phi_k}{\partial t} + \frac{\partial}{\partial x_i} \left( F_i \phi_k - \Gamma_k \frac{\partial \phi_k}{\partial x_i} \right) = S_{\phi_k}
\]

Steady state  zero (no) flux function  zero source

• The particles “feel” a force \( \vec{F} = -q \vec{V} \varphi \)

• How do we tell Fluent that?

Follow me!
Where to Get More Information and Help

• UDF User Guide
  – Installed on your machine already as part of standard installation
  – Contains macro definitions, numerous code examples and code fragments.

• Start your own UDF program by modifying an existing UDF program which is close to what you want to do, then step by step add your own code to the program.

• Use as much you can from what ANSYS Fluent UDF framework is offering you. It is tested and works for the most cases. Don’t invent the wheel again!

• Attend the Advanced UDF Training course
Upcoming ANSYS Events

• Two on-line training courses given as a series

• "Standard C for writing UDF:s in ANSYS Fluent’’
  – September 18 9.00 - 10.30
  – September 20 9.00 - 10.30
  – September 25 9.00 - 10.30
  – September 27 9.00 - 10.30

Already familiar with C? Skip the first part!

• "Using User Defined Functions in ANSYS Fluent’’
  – October 2 9.00 - 10.30
  – October 4 9.00 - 10.30
  – October 9 9.00 - 10.30
  – October 11 9.00 - 10.30

• You can register and find more information on

Curious About Making User Defined Functions in ANSYS Fluent?

To Ask a Question:

Click on the Q&A tab in the WebEx Toolbar

Webinar Recording:

Available shortly afterwards