Brief tutorial on using User-Defined Functions (UDFs) on the CARES cluster

0 – This is not intended to be a primer on how to write a UDF. There is a plethora of good information out there on how to write UDFs. Look at the User's Center at www.fluent.com for some starting points. Specifically, FLUENT has a UDF manual with many example scripts.

1 - OK, so you've got your UDF. If possible, I would recommend running it in interpreted mode on your desktop machine first. To do this (on a Windows machine) you need to have Visual C++ installed, or at least have the function 'nmake' available. If your UDF can be interpreted, it can be run on the cluster. The only difference will be that you need to compile the function, rather than interpret it. It's a great deal easier to debug a UDF in interpreted mode than in compiled mode on the cluster.

NOTE, compiled UDFs tend to run faster as well. There are a few more steps, but it isn't overly difficult.

Compiling a UDF on the cluster.

- Place your *.bash, *.flin, *.cas and *.c file in a directory of your cluster. The *.c file is your UDF.

Here's a short, example UDF. Note that all UDFs require the #include "udf.h" line in order to work properly.

```
/* Viscous Resistance Profile UDF in a Porous Zone */
#include "udf.h"
DEFINE_PROFILE(end_lower,t,i)
{
  real x[ND_ND];
  real a;
  cell_t c;

  begin_c_loop(c,t)
  {
    if((x[2] > 0) && (x[2] < 0.001))
     a = 2.7385e+10;
    else
     a = 2.7385e+12;
    F_PROFILE(c,t,i) = a;
  }
  end_c_loop(c,t)
}
```

- To compile this udf you must add the following line to your *.flin file.

```
/define/user-defined/compiled-functions compile "libudf"
yes "EndCap-Lower1.c" "" ""
```

This invokes the compile command for user-defined functions, specifies that you'll create a directory called "libudf" to hold the associated files and tells FLUENT that the name of the source code is "EndCap-Lower1.c".

If this UDF functions properly, a new directory called "libudf" will be created within the directory with your other FLUENT files. These files are then accessed by FLUENT to perform whatever function you wish to perform with the UDF. For the above example UDF, the permeability of a porous region is varied. A lower resistance is specified in the 1st 1mm of a zone, in the z direction.

In order for the UDF to work properly it must be 'hooked' to FLUENT. For each type of UDF there are different places that the UDF must be hooked, depending on what the function will actually do. For the above example a fluid zone is being modified to include the porous media variations just discussed. Thus, we hook the UDF in the boundary conditions of the loaded *.cas file. The command for this is as follows.

```
/define/boundary-conditions/fluid porous no no no yes 0 0 0
0 0 1 no yes no no 1 no 0 no 0 no 0 no 1 no 0 yes yes yes
"udf" "end_lower::libudf" yes yes "udf" "end_lower::libudf"
yes yes "udf" "end_lower::libudf" no no 0 no 0 no 0 0 0 no
no 0 no 0 0 0 no 0.16
```

(Note, as written in my *.flin file, this all is on one line. There are different ways to write it, but this is the method I use).

The important commands in the above line are the
`yes yes "udf" "end_lower::libudf"`
sections, which tell FLUENT,

- yes, I want to use a profile in the x (y and z) directions for my viscous resistance (ie permeability)
- yes, I want to specify a udf to use as the profile
- the udf can be found in the "udf" names already compiled
- the udf name is "end_lower" in the "libudf" directory

The other inputs in the above line specify other properties in the porous medium, not really important for this udf discussion.

And once again, there is a great deal of information out there about specific UDFs that may be useful to your problem. Find those.

In summary
1 – Write your udf.
2 – Test/debug your udf on a desktop machine (if possible), using the interpreted method.
3 – Save your *.cas, and other files to a separate directory on your CARES account.
4 – Compile your udf using text commands in you *flin file
5 – Hook your udf using the appropriate text commands in your *flin file.

Good Luck and Happy Computing

- Dustin Crandall 12/2007