UDFs for Multiphase Flows

Advanced UDF Modeling Course
Domain and Threads

Analysis Domain

Upper wall

Porous medium

Inlet

Porous medium

Solid #1

Solid #2

Outlet

Lower wall

Analysis Domain

Corresponding UDF Structure

Domain

Threads

wall

solid-2

fluid-2

inlet

porous medium

solid-1

fluid-1

outlet
Data Structure for Multiphase UDFs

- Data Structure in multiphase models involve multiple domains:
  - Super Domain: This is the top-level domain contains all phase-independent and mixture data: geometry, connectivity, property
  - Subdomains (phase domains): Each phase has a sub-domain that inherits the mixture-specific data and maintains the phase-specific data
  - Interaction Domain: To activate the phase interaction mechanisms
The Threads in Multi-Domains

- The mixture:
  - In single phase, a mixture represents the sum over all the species.
  - In multiphase it represents the sum over all the phases.
- This distinction is important!
  - The code will later be extended to multiphase multicomponent fluids (where, for example, a phase could be a mixture of species)
- Thread data structures:
  - Threads must be associated with the superdomain and all sub-domains
    - For each cell/face thread within the superdomain, there is a corresponding cell/face thread for each subdomain.
    - Some of the information defined in one thread of the superdomain is shared with the corresponding threads of each of the subdomains.
    - Threads associated with the superdomain are referred to as superthreads
    - Threads associated with the subdomain are referred to as phase-level threads or subthreads
Domain IDs and Thread IDs

- For multiphase models, all domains (including the interaction domain) must be identified by a unique ID.
- **Domain_ID** of the super (mixture) domain is always equal to 1.
- **Domain_IDs** are not necessarily ordered sequentially.
- Therefore, to access the phase domains, each phase also has a **phase_domain_index**:
  - 0 for the primary phase
  - \(N - 1\) for the last secondary phase
  - **phase_domain_index** is used in UDFs to retrieve phase thread pointers.
    - Useful when you want to access data for another phase from an UDF for a particular phase
Domain Looping Macro

- **sub_domain_loop(subdomain, mixture_domain, phase_index)**
  - Loops over all phases (sub-domains) in a mixture.
  - *mixture_domain* is already available
  - *subdomain* and *phase_index* are defined locally, initialized within the macro

- An example for the loop, *Domain_ID* and the *phase_domain_index*:

```c
DEFINE_ADJUST(print_id, mix_domain)
{
    Domain *s_d;/*subdomain pointer, locally defined*/
    int p_d; /* loop counter for phase_domain_index, locally defined*/
    int p_d_id; /* mix_domain is available*/
    sub_domain_loop(s_d, mix_domain, p_d)
    {
        p_d_id = DOMAIN_ID(s_d);
        Message("the phase domain id = %d and the phase
domain index = %d\n", p_d_id, p_d);
    }
}
```
Thread Looping Macro

- **sub_thread_loop**(subthread,mixture_thread,phase_index)
  - Loops over all threads in a mixture.
  - **mixture_thread** is already available.
  - **subthread** and **phase_index** are defined locally, initialized by the macro.

```c
/* compute bulk density of mixture and store it in a UDM*/
DEFINE_ADJUST(calc_den, mix_domain)
{
    Thread *mix_thread;
    thread_loop_c(mix_thread, mix_domain)
    {
        cell_t c;
        begin_c_loop(c, mix_thread)
        {
            Thread *s_t;
            int p_d_i;
            C_UDMI(c, mix_thread, 0) = 0.;
            sub_thread_loop(s_t, mix_thread, p_d_i)
            C_UDMI(c, mix_thread, 0) += C_VOF(c, s_t)*C_R(c, s_t);
        }
        end_c_loop(c, mix_thread)
    }
}
```
Other Looping Macros

- **mp_thread_loop_c(cell_thread, mixture_domain, pt)**
  - `cell_thread` is a pointer to mixture thread in the `mixture_domain`
  - `mixture_domain` is already assumed to be available
  - `pt` is an array of thread pointers

- **mp_thread_loop_f(face_threads, mixture_domain, pt)**
  - `face_threads` is a pointer to face thread in the `mixture_domain`
  - `mixture_domain` is already assumed to be available
  - `pt` is an array of thread pointers pointing to the phase-level threads

```c
DEFINE_ADJUST(print_vof, mix_domain)
{ Thread *mix_thread;
  Thread **pt;
  mp_thread_loop_c(mix_thread, mix_domain, pt)
  { cell_t c;
    begin_c_loop(c, mix_thread)
    Message("cell volume fraction = %f\n", C_VOF(c,pt[0]));
    end_c_loop(c, mix_thread)
  }
}
```

© 2008 ANSYS, Inc. All rights reserved.
Access the Correct Thread and/or Domain

- When writing UDFs, it is important that the correct thread / domain is accessed
  - C_R(cell,thread) will return
    - The mixture density if thread is the mixture thread or
    - The phase densities if it is the phase thread
- In general the type of DEFINE macro determines which thread or domain (mixture or phase) gets passed to your UDF.
  - DEFINE_INIT and DEFINE_ADJUST functions always get passed the domain structure associated with the super domain
  - DEFINE_ON_DEMAND functions are not passed any domain structures
  - If your UDF is not explicitly passed the pointer to the thread or domain required, then you can use a multiphase-specific utility macro to retrieve it
Subthreads and Superthreads

- Each thread is also in a hierarchy that matches that of the domains.
- The superthreads are where the mixture of the phases is stored and so are often called the mixture threads.
- Shared values such as the cell geometry are stored in the superthread.
- Each phase has its own set of threads called subthreads or phase thread.

---

**Diagram**

```
  superthreads
     /       \
    /         \
   /           \
  P0 Threads   P1 Threads   P2 Threads
```

Phase Subthreads

Phase Thread Interaction
Accessing Variables External to a UDF

◆ **Get_Domain(Domain-ID)**
  - Usage: `Domain *domain=Get_Domain(n);`
  - `n` is the Domain ID, as appears in Define Phases panel. It is always 1 for the mixture domain.

◆ **DOMAIN_ID(domain)**
  - Usage: `int domain_id = Domain_ID(subdomain);`
  - `subdomain` is the pointer to a phase domain; `domain_id` upon return is the same integer ID displayed in the Define Phases panel.

◆ **DOMAIN_SUB_DOMAIN(mixture_domain,ph_domain_index)**
  - Returns the phase pointer subdomain for the given `phase_domain_index`.
  - Usage: `Domain *mixture_domain; Domain *subdomain = DOMAIN_SUB_DOMAIN (mixture_domain, phase_domain_index);`
Accessing Variables External to a UDF

- **THREAD_SUB_THREAD(mixture_thread,ph_domain_index)**
  - Returns the phase level thread pointer for the given `ph_d_index`.
  - Usage: ```
        int ph_d_index = 0; /* primary phase index is 0 */
        Thread *mix_th; /* mixture-level thread pointer */
        Thread *subth=THREAD_SUB_THREAD(mix_th, ph_d_index);
      ```

- **THREAD_SUB_THREADS(mixture_thread)**
  - Returns the pointer array, `pt`, whose elements contain pointers to phase level threads (subthreads).
  - Usage: ```
        Thread *mixture_thread;
        Thread **pt; /* initialize pt: pointer array */
        pt = THREAD_SUB_THREADS(mixture_thread)
      ```
Accessing Variables External to a UDF

- **THREAD_SUPER_THREAD(subthread)**
  - Returns the phase-level thread pointer for the given `ph_d_index`.
  - Usage:
    ```c
    Thread *subthread; /* phase thread pointer */
    Thread *mix_th;    /* mixture thread pointer */
    Thread *subth = THREAD_SUB_THREAD(mix_th,ph_d_index);
    ```

- **DOMAIN_SUPER_DOMAIN(subdomain)**
  - Returns the mixture domain pointer
  - Usage:
    ```c
    Domain *subdomain; /* phase domain pointer */
    Domain *mixture_domain = DOMAIN_SUPER_DOMAIN(subdomain);
    ```
Accessing Variables External to a UDF

- **THREAD_SUPER_THREAD(subthread)**
  - Given a phase thread pointer, returns the superthread (mixture thread) pointer.
  - Usage:
    ```c
    Thread *subthread; /* phase thread pointer */
    Thread *mix_thread=THREAD_SUPER_THREAD(subthread)
    ```

- **DOMAIN_SUPER_DOMAIN(subdomain)**
  - Returns the mixture domain pointer.
  - Usage:
    ```c
    Domain *subdomain; /* phase domain pointer */
    Domain *mixture_domain =
    DOMAIN_SUPER_DOMAIN(subdomain)
    ```
Accessing Variables External to a UDF

- **PHASE_DOMAIN_INDEX(subdomain)**
  - Returns the phase domain index for the phase domain (subdomain) pointer; It is an integer that starts with ‘0’ for the primary phase and is incremented by one for each secondary phase.
  - Usage:
    ```c
    Domain *subdomain; /* phase domain pointer */
    int phase_domain_index =
        PHASE_DOMAIN_INDEX(subdomain)
    ```

- **THREAD_DOMAIN(thread)**
  - Returns domain pointer for the thread.
  - Usage:
    ```c
    Thread *subthread; /* phase thread pointer */
    Thread *mix_thread = THREAD_SUPER_THREAD(subthread)
    Domain *subd = THREAD_DOMAIN(subthread); /* phase domain pointer */
    Domain *mixd = THREAD_DOMAIN(subthread); /* mixture domain pointer */
    ```
**Exchange Property Macros**

- **DEFINE EXCHANGE_PROPERTY**
  - This macro is used to specify custom drag and lift coefficients for the Eulerian multiphase model.
  - Usage: `DEFINE_EXCHANGE_PROPERTY(name, c, mixture_thread, second_column_phase_index, first_column_phase_index)`
  - `mixture_thread` points to the mixture thread.
  - `C` is the index of a cell on the `mixture_thread`.
  - `first_column_phase_index` and `second_column_phase_index` are integer identifiers corresponding to the pair of phases in your multiphase flow.
  - The identifiers correspond to the phases that are selected in the Phase Interaction panel.
  - The UDF returns the real value of the lift or drag coefficient (`name`, `c`, `mixture_thread`, `second_column_phase_index`, and `first_column_phase_index`).
Exchange Property Macros

- **DEFINE_VECTOR_EXCHANGE_PROPERTY**
  - This macro is used to specify custom slip velocities for the Mixture multiphase model.
  - Usage: `
```c
DEFINE_VECTOR_EXCHANGE_PROPERTY
(name, c, mixture_thread,
 second_column_phase_index,
 first_column_phase_index, vector_result)
```
  - `mixture_thread` points to the mixture thread.
  - `c` is the index of a cell on the `mixture_thread`.
  - `first_column_phase_index` and `second_column_phase_index` are integer identifiers corresponding to the pair of phases in your multiphase flow.
  - The identifiers correspond to the phases that are selected in the Phase Interaction panel.
  - The UDF is passed the real pointer to the slip velocity vector `vector_result`, and it will need to set the components of the slip velocity vector.
Exchange Property Macro Example

```c
#include "udf.h"
#include "sg_mphase.h"

DEFINE VECTOR EXCHANGE PROPERTY(custom_slip, c, mixture_thread, second_column_phase_index, first_column_phase_index, vector_result)
{
    real K = 5.e4;
    real pgrad_x, pgrad_y;
    Thread *pt, *st; /* Primary and secondary phase thread pointers */
    pt = THREAD_SUB_THREAD(mixture_thread, second_column_phase_index);
    st = THREAD_SUB_THREAD(mixture_thread, first_column_phase_index);

    /* Now the threads are known for primary (0) and secondary (1) phases */
    pgrad_x = C_DP(c, mixture_thread)[0];
    pgrad_y = C_DP(c, mixture_thread)[1];
    vector_result[0] = -(pgrad_x/K)+(((C_R(c, st)-C_R(c, pt))/K)*grav[0]);
    vector_result[1] = -(pgrad_y/K)+(((C_R(c, st)-C_R(c, pt))/K)*grav[1]);
}
```
Cavitation Macros

- **DEFINE_CAVITATION_RATE**

- You can use this macro to model the creation of vapor due to pressure tension in a multiphase flow
  - It is applied in the User-Defined Function Hooks panel.
  - \( t \) is a pointer to the mixture level thread
  - \( c \) is the index of a cell on the thread pointed to by \( t \)
  - The remaining arguments are real pointers to the following data:
    - Shared pressure (\( p \)), vapor density (\( \rho_{V} \)), liquid density (\( \rho_{L} \)), vapor volume fraction (\( \text{vofV} \)), vaporization pressure (\( p_{v} \)), number of bubbles per unit volume (\( n_{b} \)), and rate of vapor formation (\( \text{mdot} \))
  - The UDF sets the value referenced by the real pointer \( \text{mdot} \), to the cavitation rate.
  - Usage: \( \text{DEFINE_CAVITATION_RATE}(\text{name}, \ c, \ t, \ p, \ \rho_{V}, \ \rho_{L}, \ \text{vofV}, \ p_{v}, \ n_{b}, \text{mdot}) \)
Miscellaneous Multiphase Macros

- Phase diameter
  \[ \text{C\_PHASE\_DIAMETER}(c, \text{phase\_thread}) \]

- Phase Volume fraction
  \[ \text{C\_VOF}(c, \text{phase\_thread}) \]

- Phase velocity gradients
  \[ \text{C\_U\_G}(c, \text{phase\_thread}) \]
  \[ \text{C\_V\_G}(c, \text{phase\_thread}) \]
  \[ \text{C\_W\_G}(c, \text{phase\_thread}) \]
Miscellaneous Multiphase Macros

- Phase volume fraction gradients: `C_VOF_G(c,phase_thread)`
  - User-defined memory must also be allocated.
  - Gradients must be explicitly calculated.

```c
Domain *pDomain =
    DOMAIN_SUB_DOMAIN(domain,P_PHASE);
Alloc_StorageVars
    (pDomain, SV_VOF_RG, SV_VOF_G, SV_NULL);
Scalar_Reconstruction(pDomain, SV_VOF, -1,
    SV_VOF_RG, NULL);
Scalar_Derivatives
    (pDomain, SV_VOF, -1, SV_VOF_G, SV_VOF_RG,
        Vof_Deriv_Accumulate);
```
Miscellaneous Multiphase Macros

- Check if a given thread is a super or sub thread.
  - THREAD_SUPER_THREAD(thread) is NULL for mixture thread, and not NULL for phase threads.
    - Mixture: if ( NULLP (THREAD_SUPER_THREAD(thread)))
    - Phase: if (!NULLP (THREAD_SUPER_THREAD(thread)))