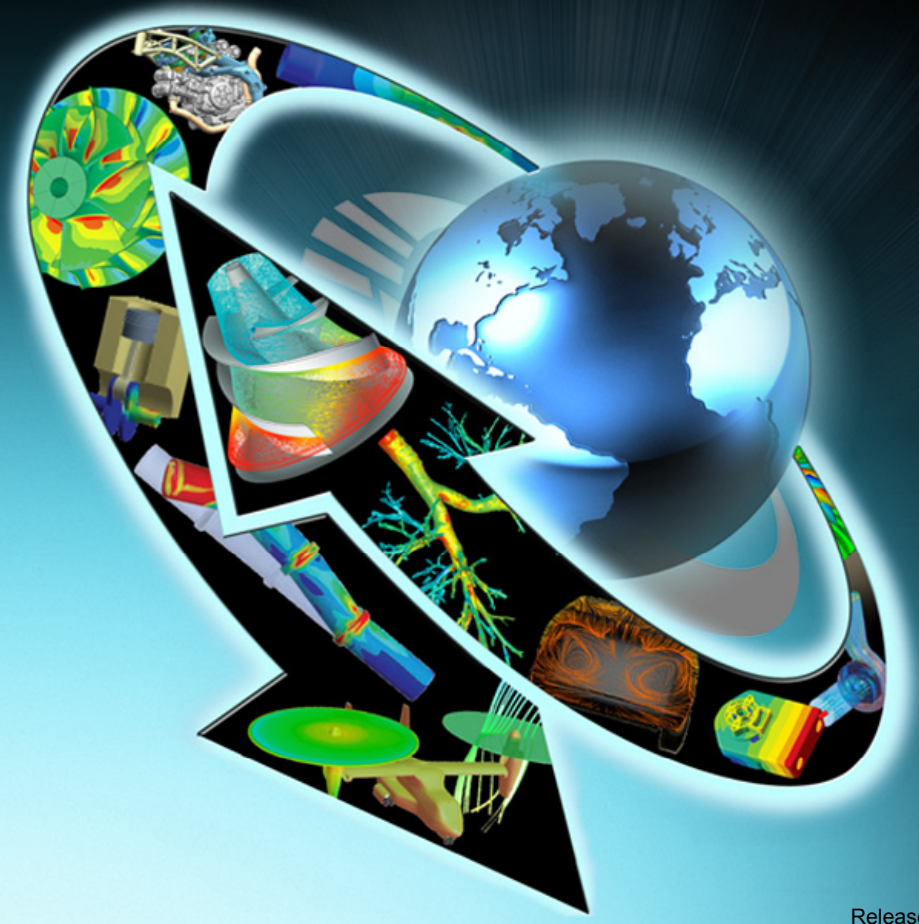
ANSYS®

Chapter 4

User Defined Scalars and Memories



Advanced FLUENT User-Defined Functions

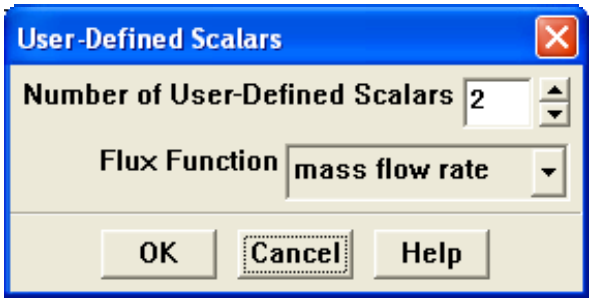
Release 13.0
December 2010

User-Defined Scalars (1)

- FLUENT allows users to solve generic transport equations called User Defined Scalars (UDS) governed by:

$$\frac{\partial(\rho\phi_k)}{\partial t} + \frac{\partial}{\partial x_i}\left(F_i\phi_k - \Gamma_k\frac{\partial\phi_k}{\partial x_i}\right) = S_{\phi_k} \quad k = 1, ..., N$$

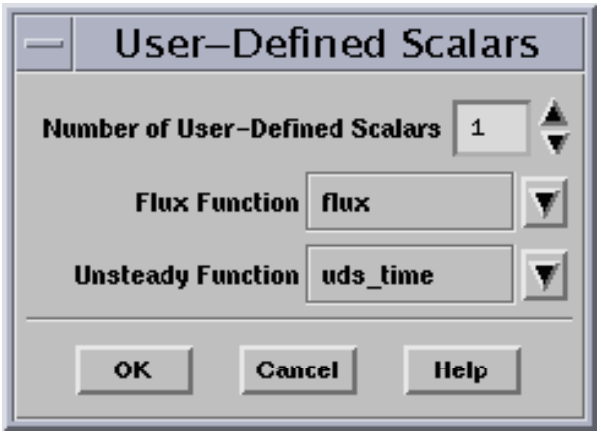
- The menu is accessed through **Define→Models→User-Defined Scalars…**
- User specifies number of User- Defined Scalars and UDF can be used for parts of scalar transport equation :
  - Advective flux : **DEFINE_UDS_FLUX** for the non-default definition
  - Unsteady term: **DEFINE_UDS_UNSTEADY**
  - Diffusivity: **DEFINE_DIFFUSIVITY**



User Defined Scalars (2)

- User Defined Scalar convective and time derivatives can be modified

```
DEFINE_UDS_FLUX(flux, f, t, i)
{
 if (i == 0) return 0.;
 if NNULLP(THREAD_STORAGE(t,SV_FLUX))
    return F_FLUX(f,t);
  return 0.;
}
```

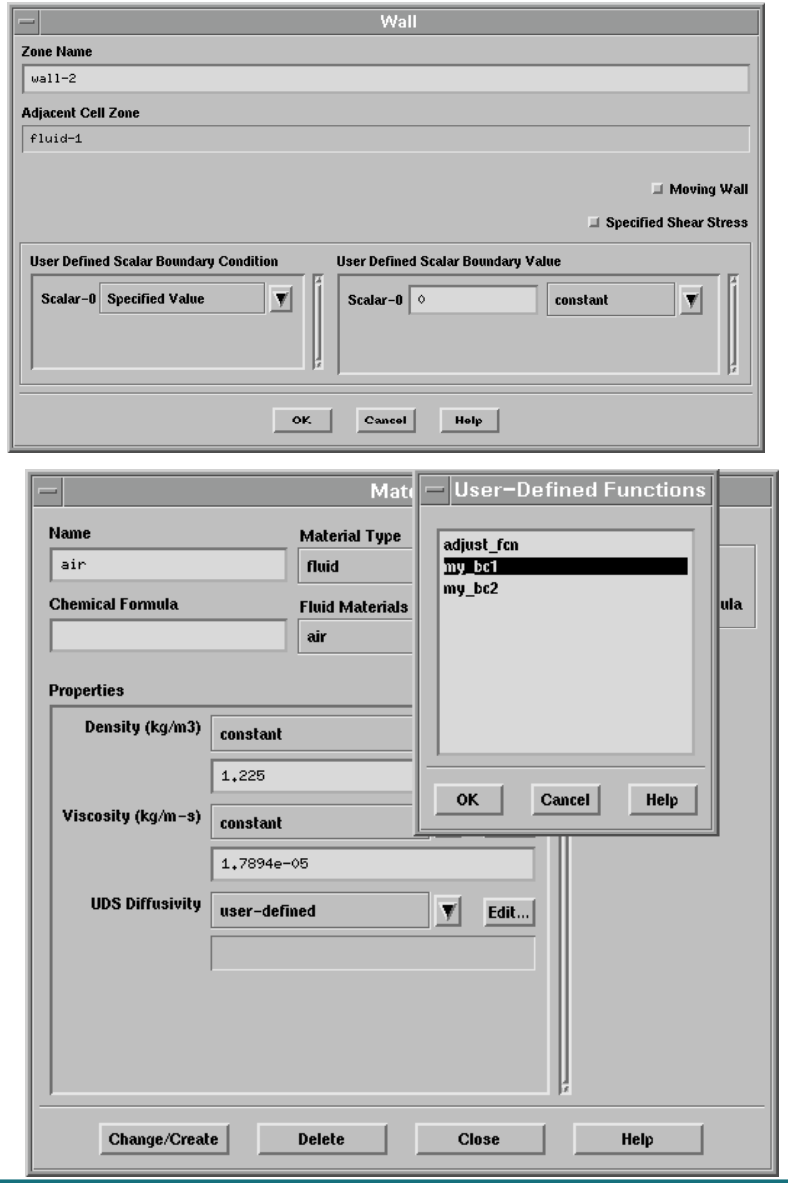



```
DEFINE_UDS_UNSTEADY(uns_time, cell,
    thread, i, apu, su)
{
 real physical_dt, vol, rho, phi_old;
  physical_dt = RP_Get_Real("physical-
    time-step");
  vol = C_VOLUME(cell,thread);

  rho = Rhod;
  *apu = -rho*vol /
    physical_dt;/*implicit part*/
  phi_old =
    C_STORAGE_R(cell,thread,SV_UDSI_M1(i
    ));
  *su  =
    rho*vol*phi_old/physical_dt;/*explic
    it part*/
}
```

User Defined Scalars (3)

- The Boundary Conditions for the User Defined Scalar can be specified as **Specified Flux** or **Specified Value**
- **DEFINE_DIFFUSIVITY(name,c,t,i)** is to specify the diffusivity for the species transport equations or for the user-defined scalars, **i** is an integer index identifying to which species or user-defined scalar a diffusivity is assigned. This UDF is hooked up to the FLUENT solver in the Materials panel
- Other types of boundary condition can be implemented by user-supplied **DEFINE_PROFILE** macro



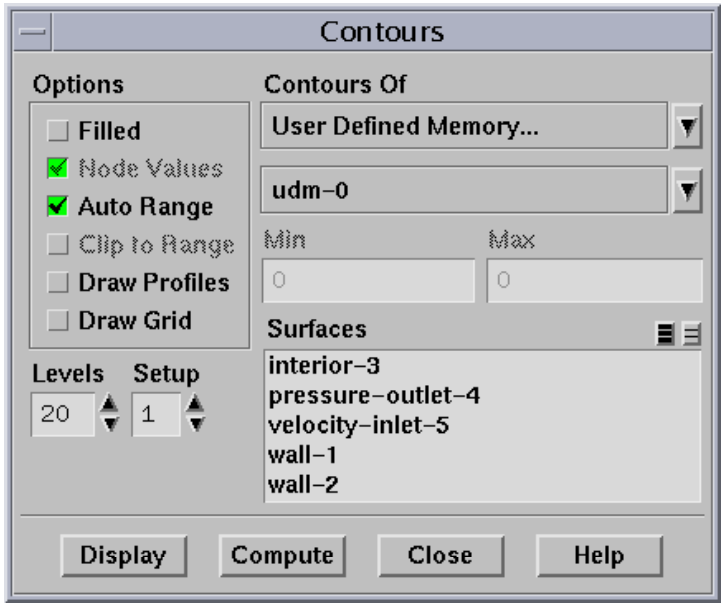
User Defined Scalars (4)

- The User Defined Scalars and their gradients can be used in UDFs. One example is shown here
- **Data_Valid_P( )** is a boolean macro which returns true when variable's storage is allocated. In this example, function exits immediately if C_UDSI_G(c,t,0) is not allocated

```
DEFINE_ADJUST(adjust_fcn,domain)
{
  Thread *t;
  int nt;
  cell_t c;
  face_t f;
  int ns;
  real p_dis = 0.;
/* Don't execute below here if UDS gradient isn't
   allocated yet. */
if (! Data_Valid_P())return;
/* Compute power dissipated. */
  thread_loop_c (t, domain)
  if (FLUID_THREAD_P(t))
   {
   begin_c_loop_all (c,t)
     {
    C_UDSI(c,t,1)+=K_EL*
      NV_MAG2(C_UDSI_G(c,t,0))*C_VOLUME(c,t);
     }
   end_c_loop_all (c, t)
   }
}
```

User Defined Memory (1)

- User-allocated memory
  - Allow users to allocate memory (up to 500 locations) to store and retrieve the values of *field variables* computed by UDF's (for postprocessing and use by other UDFs)
  - Same array dimension and size as any flow variable
  - UDMs are not automatically solved by the solver
  - Number of User-Defined Memory Locations is specified in the User-Defined Memory panel
  - Accessible via macros
    - Cell values: C_UDMI(c,t,i)
    - Face values: F_UDMI(f,t,i)
  - Saved to FLUENT data file



User Defined Memory (2)

- Example: Use UDM to compute a user-defined non-dimensional temperature:

```
DEFINE_ON_DEMAND(scaled_temp)
{ Domain *domain = Get_domain(1);
  Thread *t;
  cell_t c;
  real temp;
/* Compute scaled temperature store in user-defined memory
   */
thread_loop_c(t,domain)
 {
 begin_c_loop(c,t)
  {
    temp = C_T(c,t);
    C_UDMI(c,t,0)=(temp - tmin)/(tmax-tmin);
  }
end_c_loop(c,t)
 }
}
```

Execute on Demand

- Execute On Demand UDF provides a hook to execute any calculations or I/O operations at will of the user while the solver is not iterating
- Executed instantaneously when activated by user
- Define/User-Defined/Execute on Demand...
- Can be used for testing user's own code without changing the existing solution fields


```
#define SETMIN(a,b)((b)<(a)?(a=b):(a))
#define SETMAX(a,b)((b)>(a)?(a=b):(a))
DEFINE_ON_DEMAND(scaled_temp)
{
Domain *domain;
thread_loop_c(t,domain)
{
  real tmin=-1.e10, tmax=1.e10;
  /* Compute min & max temperature */
  begin_c_loop(c,t)
    {
     SETMIN(tmin,C_T(c,t));
     SETMAX(tmax,C_T(c,t));
    }
     end_c_loop(c,t)
 }
}
```

Execute On Loading

- DEFINE_EXECUTE_ON_LOADING (name, libname)

  This is a general function which is executed immediately after the UDF library is loaded. This UDF is handy to initialize the variables or fields in the UDF library, and used in conjunction with UDS and UDM