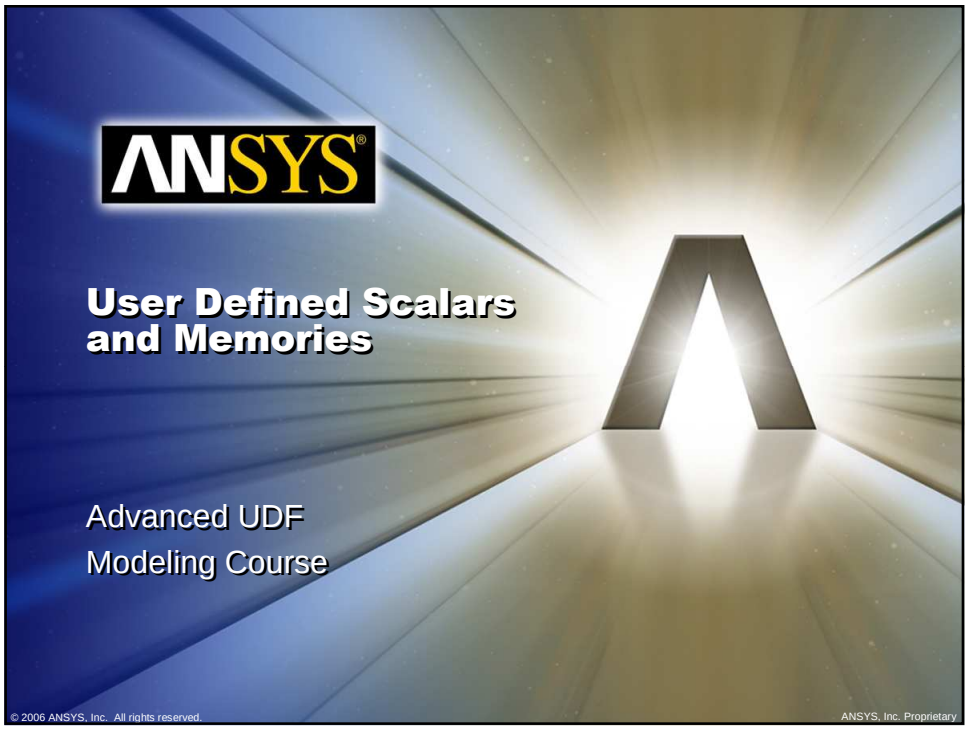
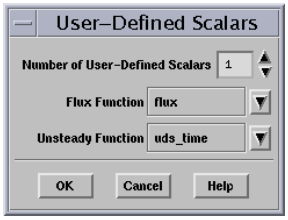


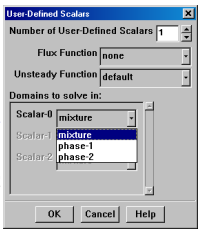
User Defined Scalars (1)

- FLUENT can solve generic transport equations for User Defined Scalars
- 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: **`DEFINE_UDS_FLUX`**
  - Unsteady: **`DEFINE_UDS_UNSTEADY`**
  - Diffusivity: **`DEFINE_DIFFUSIVITY`**



$$\frac{\partial}{\partial t}(\rho\phi) + \frac{\partial}{\partial x_j}(\rho u_j \phi) = \frac{\partial}{\partial x_j}\left(D\frac{\partial \phi}{\partial x_j}\right) + S$$

$$\frac{\partial}{\partial t}(\alpha\rho\phi) + \frac{\partial}{\partial x_j}(\alpha\rho u_j \phi) = \frac{\partial}{\partial x_j}\left(D\frac{\partial \phi}{\partial x_j}\right) + S$$

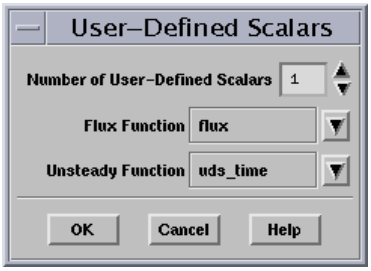


Scalars are phase-specific in multiphase models
Will be discussed later

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.;
}
```

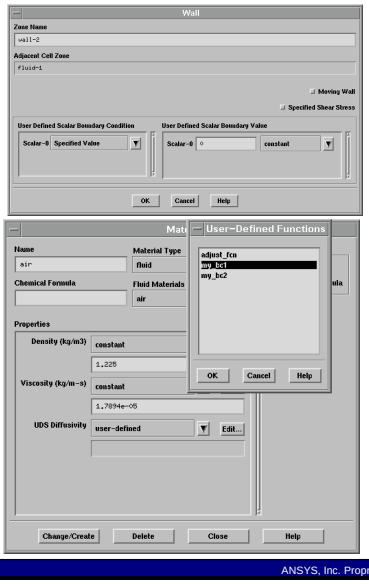



```
DEFINE_UDS_UNSTEADY(uns_time, cell,
    thread, i, apu, su)
{
 real physical_dt, vol, rho, phi_old;
  physical_dt = RP_Get_Real("physical-
    time-step");
  vol = C_VOLUME(cell,thread);

  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;/*expli
    cit part*/
}
```


User Defined Scalars (3)

- The Boundary Conditions for the User Defined Scalar can be specified as **`Specified Flux`** or **`Specified Value`**
- The diffusivity for the User Defined Scalar can be specified through **`Material→user-defined→diffusivity`** panel as a constant or as User-Defined Function



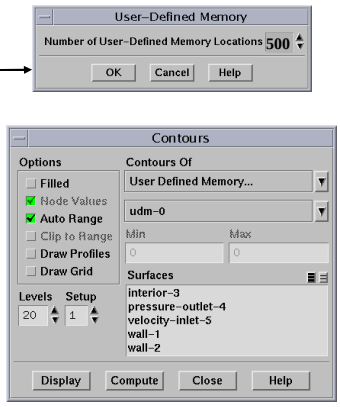
User Defined Scalars (4)

- The User Defined Scalars and their gradients can be used in UDF's

```
DEFINE_ADJUST(adjust_fcn,domain)
{
  Thread *t;
  int nt;
  cell_t c;
  face_t f;
  int ns;
  real p_dis = 0.;
/* Do nothing if 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 (UDM)

- 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 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)

```
DEFINE_ON_DEMAND(scaled_temp)
{
  Domain *domain = Get_domain(1);
/* 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

- This provides a hook to execute any set of calculation 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...


```
extern Domain *domain;
#define SETMIN(a,b)((b)<(a)?(a=b):(a))
#define SETMAX(a,b)((b)>(a)?(a=b):(a))
DEFINE_ON_DEMAND(scaled_temp)
{
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)
 }
}
```