

Fluent Tips & Tricks

UGM 2004

Sutikno Wirogo
Samir Rida

2004 CFD SUMMIT

Fluent Training

The meeting of the minds in flow modeling

List of Tips and Tricks collected from:

- Solution database available through the Online Technical Support (OTS) portal

<http://www.fluentusers.com>

- Frequently Asked Questions
- Know-how of Fluent's technical staff

This presentation provides Tips and Tricks:

- For IO and Batch
- For Case Set-up and Mesh
- For Solving
- For Post-Processing
- For Reporting



This presentation provides Tips and Tricks:

- For IO and Batch
- For Case Set-up and Mesh
- For Solving
- For Post-Processing
- For Reporting

Miscellaneous on Parallel IO

- In parallel, the mesh is first read into the **host** and then distributed to the **compute nodes**
- If reading case into parallel solver takes **unusually long time**, do the following:
 - Merge as many zones as possible
 - Put the host and compute-node-0 on the same machine
 - Put the case and data files on a disk on the machine running the host and compute-node-0 processes
- Parallel Fluent will allocate buffers for exchanging messages while reading and building the grid
 - Default buffer size depends of the case that is being read
For 4 million cell case on 4 CPUs, the buffer size will be $4M/4 = 1M$
 - To query buffer size, use the following scheme command:
`(%query-parallel-io-buffers)`
 - If the host machine does not have enough memory, using a large buffer will slow down the read case stage and thus can limit the maximum buffer size using:
`(%limit-parallel-io-buffer-size 0)`
 - Needs to be executed before reading the case file
 - Will increase the total number of messages exchanged but may still give better performance
 - Once the case file has been read, can use the following to reset/free the buffers:
`(%allocate-parallel-io-buffers 10000000)`
`(%query-parallel-io-buffers)`

Miscellaneous on Parallel IO

- If the machine running the host process has enough memory, can speed up reading the case file by using the following Scheme command:

```
(rpsetvar 'parallel/parallel/fast-io? #t)
```

Fluent will first read the whole mesh into the host machine before distributing to the compute nodes

- When loading a **large mesh** into **parallel Fluent** and the process hangs, execute the following Scheme command:

```
(rpsetvar 'parallel/case-read-use-pack? #f)
```

The above will disable buffer packing and will change the way entities are packed into messages during the build process

- To check the time taken to read a case, can use:

```
(print-case-timer)
```

- Fluent 6.2 has significant improvement for parallel case read

- Two available settings:

```
(rpsetvar 'parallel/fast-read-case? #t)
```

```
(rpsetvar 'parallel/fast-read-section? #t)
```

Gives the similar performance as the fast-io option in Fluent 6.1 but with much less memory usage

- Turned on by default
- If turned off, will revert to the previous 6.1 methods

- Commands to run in batch mode:

fluent 3d -g < batch.jou >& out.trn &

for C-shell

fluent 3d -g < batch.jou > out.trn 2>&1 &

for Bourne/Korne-shell

fluent 3d -g -i batch.jou *-hidden*

for windows

For Windows, without *-hidden*, the GUI will still be started but iconified

- For Windows, there is no option for getting an output file
A transcript file should be started and then stopped before exiting the FLUENT session
- To receive mail notification upon the completion of a batch job, add the following line at the end of the batch journal:

!mail email-address < message.txt

where message.txt is a text file located inside the working directory and contains the message to be sent

- To enter a comment inside a journal file:
- To execute a shell command inside a journal file:

; This line is commented

! Shell-command-to-be-executed

- To generate animation files during a batch processing, add the following option to the start up command of the batch job:

-driver null

- Is it possible to read a journal file from another journal file ?
No. Nested journal files are not allowed in Fluent
- How to execute several batch processes sequentially ?
 - In **Unix/Linux** environments, if a shell command is not ended by an ampersand (&) then the OS will wait until the completion of the execution of that line before going to the next line. **This is not the case with Windows**
 - Suppose there are two directories corresponding to two cases where each has its own corresponding journal file. Then see the following shell scripts:

Unix/Linux

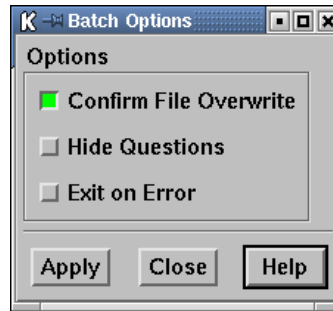
```
cd ./dir1  
fluent 3d -psmpi -t2 -g < batch1.jou  
cd ../dir2  
fluent 3d -psmpi -t2 -g < batch2.jou
```

Windows

```
call cd ./dir1  
call fluent 3d -wait -g -i batch1.jou  
call cd ../dir2  
call fluent 3d -wait -g -i batch2.jou
```

Miscellaneous on Batch Processing (3)

- Useful options for batch processing have been implemented in Fluent 6.1



- Confirm file overwrite is a must for batch processing
- Option to turn off question dialog boxes
- Must be set through GUI, no equivalent TUI/Scheme command
- Equivalent Scheme command to turn on exit on error is:

```
(set! *cx-exit-on-error* #t)
```
- Can be appended inside the journal file or .fluent file

Check Pointing (1)

- It is possible to tell the Fluent process to stop the iteration and write the case/data files without interacting with the GUI
- This procedure is useful to stop a batch process or when the GUI process has crashed, and can be used as long as **Fluent is still in the iteration loop**
- In Unix/Linux environment, the owner of the process can create a kill file inside the /tmp directory:

`unix> cd /tmp`

`unix> touch exit-fluent`

exit Fluent after writing files

or

`unix> touch check-fluent`

continue iteration after writing files

If either exit-fluent or check-fluent found in /tmp, Fluent will finish the current iteration, write the files, and then delete the exit-fluent or check-fluent file

- The files will be written to the **same directory** where the **original input files** were read and will have the same names but with appended current iteration number

By default, the case file will be written

- To write the **data file only**, execute the following Scheme command before iterating:

`(rpsetvar 'checkpoint/write-case? #f)`

Check Pointing (2)

- Make sure that sufficient disk space is available.
 - Check-pointing code calls the same file I/O routines used by the GUI or TUI and will produce the same error messages if disk space is insufficient
 - In such case, Fluent will not return to the iteration loop
- The **touch** command will produce file of zero length and is also available in Windows
- For Windows, the check point files need to be created at:
C:\temp\check-fluent.txt
C:\temp\exit-fluent.txt
- If the machine has **several Fluent sessions** running, **named check pointing** can be used to selectively stop a specific Fluent process
- A specific check point name can be added to the first line of the batch journal file, as shown below:
(set! checkpoint/exit-filename "/tmp/exit-fluent-job-1")
file/read-case-data sample.cas
solve/iterate 1000
file/write-case-data final.cas
- To stop this particular job, use the following command inside the /tmp directory:
unix> touch exit-fluent-job-1

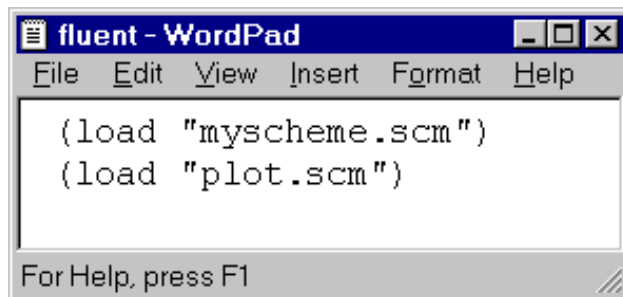


This presentation provides Tips and Tricks:

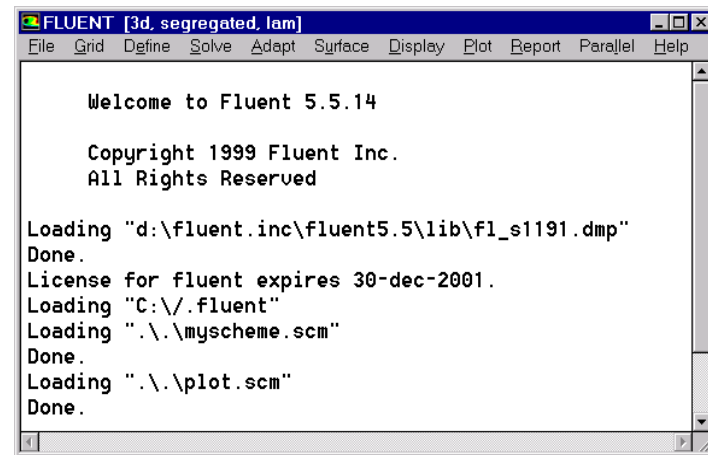
- For IO and Batch
- For Case Set-up and Mesh
- For Solving
- For Post-Processing
- For Reporting

The .fluent File (1)

- When FLUENT opens, it auto executes the .fluent file in the:
 - home directory of unix users
 - C:\ directory of Windows machines
- A .fluent file can contain any number of scheme file names to load
 - Note: unless full path is specified for each scheme file in .fluent file, FLUENT tries to locate the scheme files from the working directory



```
(load "myscheme.scm")
(load "plot.scm")
```



```
Welcome to Fluent 5.5.14

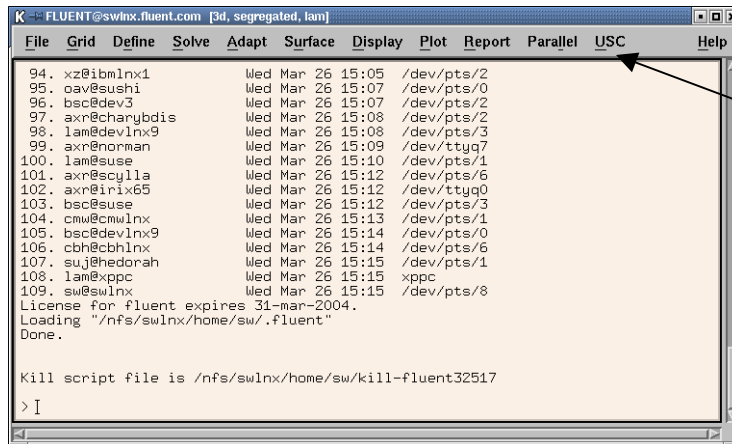
Copyright 1999 Fluent Inc.
All Rights Reserved

Loading "d:\fluent.inc\fluent5.5\lib\fl_s1191.dmp"
Done.
License for fluent expires 30-dec-2001.
Loading "C:\\.fluent"
Loading ".\myscheme.scm"
Done.
Loading ".\plot.scm"
Done.
```


The .fluent File (2)

- Scheme commands can also be appended inside the .fluent file itself

```
;; -----;;
;; Code to create USC panel
(let ((menu (cx-add-menu "USC" #\U)))
  (cx-add-item menu "User Services Center" #\O #f and (lambda ()
    (system "netscape http://www.fluentusers.com/ &"))))
;; Various useful stuff
(set! *cx-exit-on-error* #f)
;; -----;;
```



USC panel created by

The .fluent File (3)

- Other customization example:

```
(let ((old-rc client-read-case))
  (set! client-read-case
    (lambda args
      (apply old-rc args)
      (if (cx-gui?)
        (begin
          ;; Do your customization here
          (rpsetvar 'residuals/plot? #t)
          (rpsetvar 'residuals/settings '(
            (continuity #t 0 #f 0.001)
            (x-velocity #t 0 #f 0.001)
            (y-velocity #t 0 #f 0.001)
            (z-velocity #t 0 #f 0.001)
            (energy #t 0 #f 1e-06)
            (k #t 0 #f 0.001)
            (epsilon #t 0 #f 0.001)))
          (rpsetvar 'mom/relax 0.4)
          (rpsetvar 'pressure/relax 0.5)
          (rpsetvar 'realizable-epsilon? #t)
          (cxsetvar 'vector/style "arrow")
          ;; You can add more settings here
        )))))
```

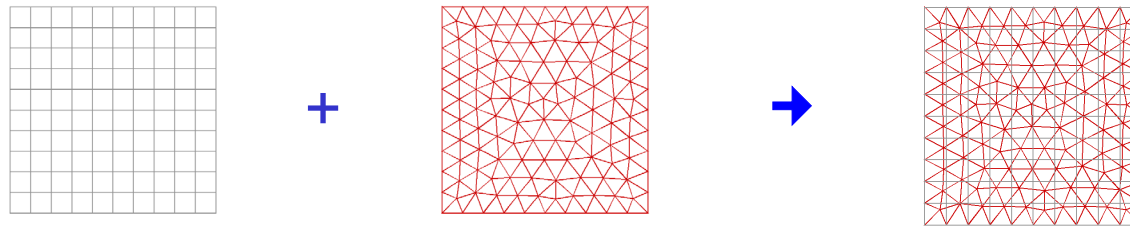
← Turning off convergence check

← Customize URFs

← Use arrow instead of harpoon for vector head

Creation of Non-conformal Interface (Solution 796)

- Intersected mesh is created using the two interface meshes and then used to 'replace' the original interface meshes.



- Creation of intersected mesh is delicate, if fails, try the alternative schemes by using either of the following Scheme commands:

(rpsetvar 'nonconformal/cell-faces 0)

(rpsetvar 'nonconformal/cell-faces 2)

The default is **(rpsetvar 'nonconformal/cell-faces 1)**

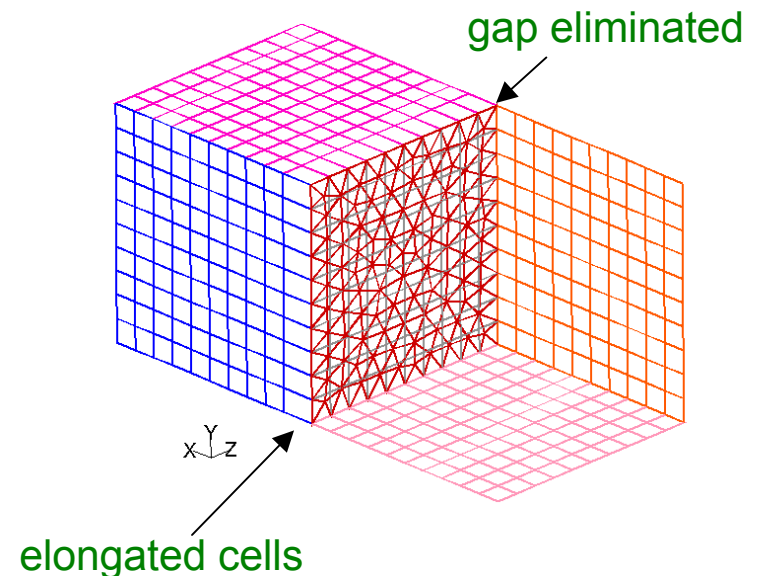
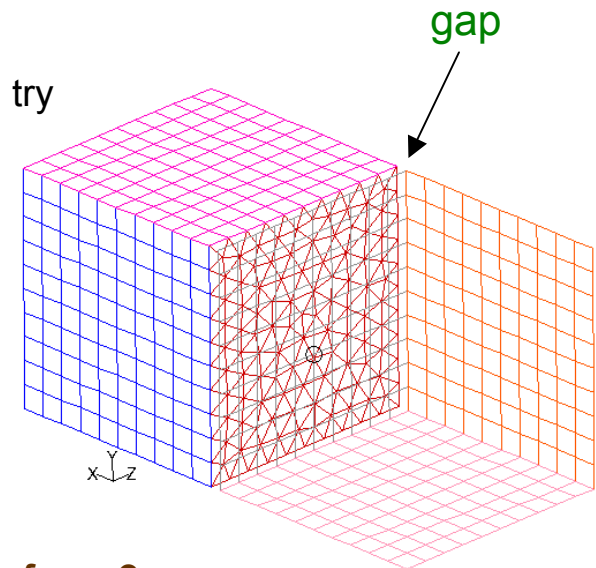
- If the interface zones are of different sizes, selecting the smaller interface as Zone 1 is usually more robust
- To create non-conformal interface which crosses periodic boundary, need to set the following Scheme command first:
(rpsetvar 'nonconformal/allow-interface-at-periodic-boundary 0)
- To improve accuracy and robustness:
 - Maintain similar face element sizes at both interfaces
 - Maintain good face mesh skewness at both interfaces

Creation of Non-conformal Interface (Solution 928)

- If the previous workaround for non-conformal interface creation still fails (e.g. due to too large of gap, etc), can try to project one interface to the other
- Write BC file from the current Fluent session.
- Read the case file into Tgrid
- Use the TUI command to project one interface to the other

`boundary/project-face-zone` *interface-1* *interface-2*

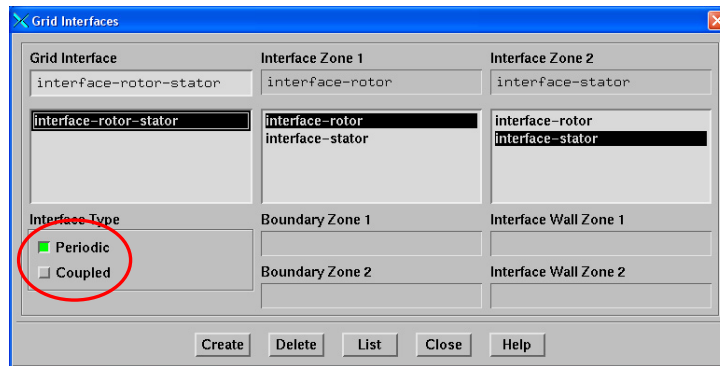
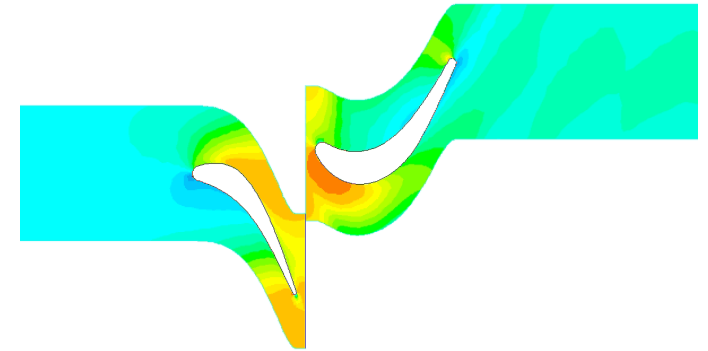
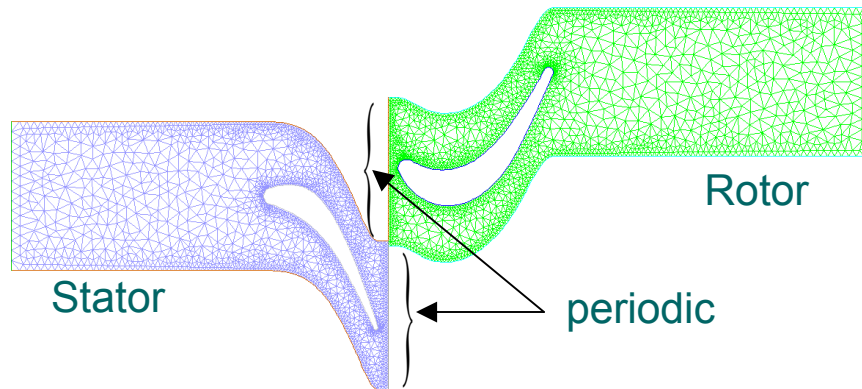
- Write a new mesh
- Use the BC file to resetup the case
- Recreate the non-conformal interface



Non-conformal Interface – Periodic Type (Overview and Example)

- The **periodic** option for the non-conformal interface is used if the non-overlapping portions of the interface is periodic

Example: Rotor-Stator interaction

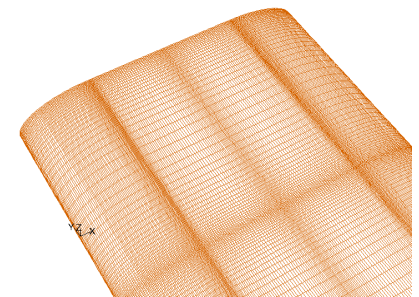
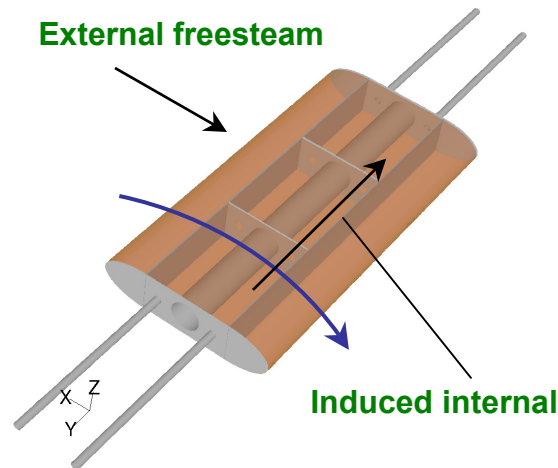
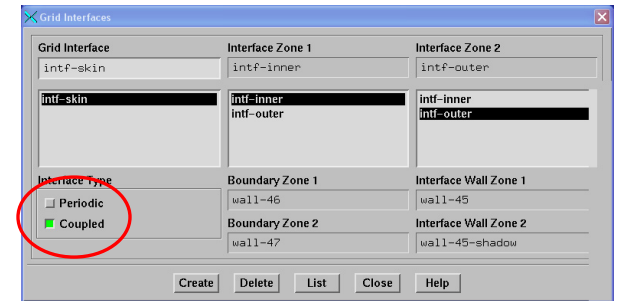


- This option is **not** for **non-conformal periodic**
 - Non-conformal periodic setup is done using the TUI commands

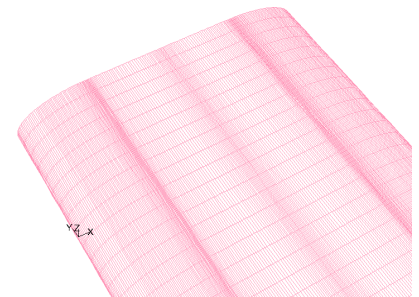
Non-conformal Interface – Coupled Type (Overview and Example)

- The ***coupled*** option for the non-conformal interface can be used to:
 - Couple non-matching **fluid** and **solid** interface meshes
 - Couple non-matching **fluid** and **fluid** interface meshes and insert a thin wall in between

Example: Canard Rotor Wing (CRW)



Inside interface (fluid)

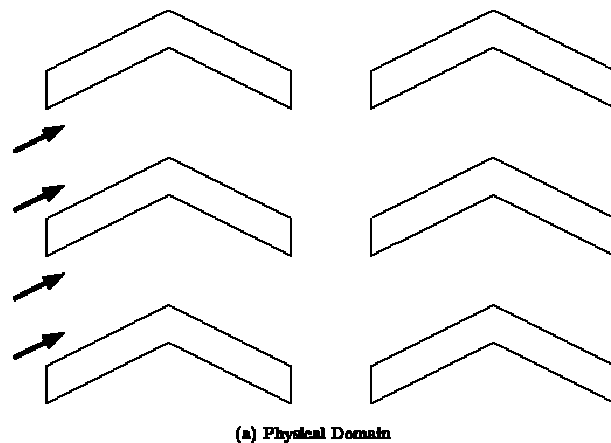


Outside interface (solid)

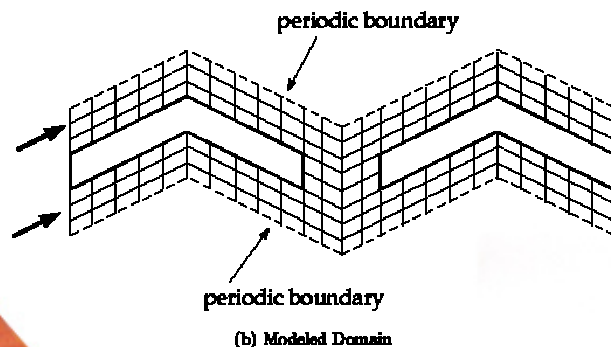
- Induced internal flow due to wing rotation
- Cooling heat transfer from the external flow
- Internal flow meshes must be fine along the span to resolve recirculations inside
- External flow meshes can be coarser along the span

Periodic Boundaries

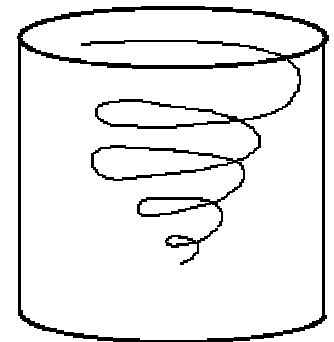
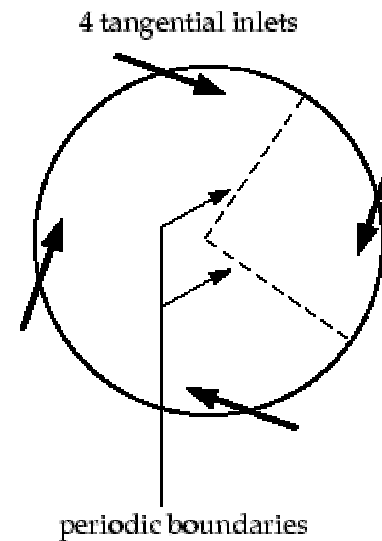
- Two types:
 - Pressure drop occurs across **translationally** periodic boundaries (e.g. tube bank)
 - No Pressure drop occurs across **rotationally** or **translationally** periodic boundaries



(a) Physical Domain



(b) Modeled Domain

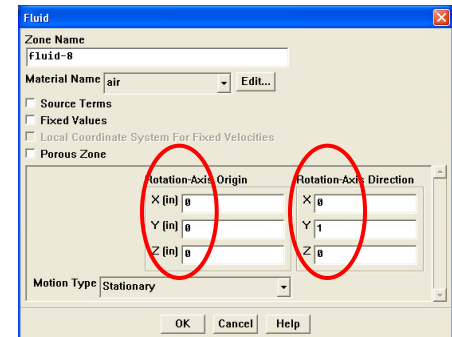
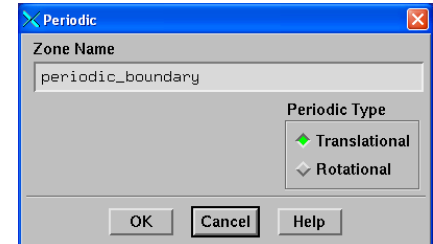


Periodic Boundaries: Conformal

- To create the periodicity in Fluent, use the TUI command:
grid/modify-zones/make-periodic
 - Can make the periodicity either translational or rotational
 - The periodicity type of the pair inside the boundary conditional panel will be updated accordingly

- For rotationally periodic cases, the periodicity axis is specified inside the fluid zone that contains the periodic pair

- Some Tips:
 - If creation fails because of **slightly non-matching** mesh nodes, increase the matching tolerance up to 0.5 (default is 0.05) using:
grid/modify-zones/match-tolerance
 - To repair corrupted periodic zones:
modify-zone/repair-periodic
 - For translationally periodic boundaries, the command computes an average translation distance and adjusts the node coordinates on the shadow face zone to match this distance
 - For rotationally periodic boundaries, the command prompts for an angle and adjusts the node coordinates on the shadow face zone using this angle and the defined rotational axis for the cell zone
 - To slit the periodic boundary into two symmetry boundaries use:
grid/modify-zones/slit-periodic

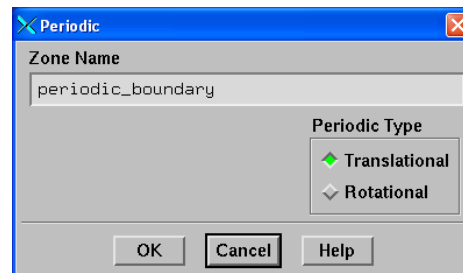


Periodic Boundaries: Non-conformal

- Set the type of non-conformal periodic zones to interface
- Input the correct axis origin and direction for the adjacent cell zone in the fluid panel
- Define the non-conformal periodic boundaries in the TUI

```
/define/grid-interfaces> make-periodic  
Periodic zone [()] interface-15  
Shadow zone [()] interface-2  
Rotational periodic? (if no, translational) [yes] yes  
Rotation angle (deg) [0] 40.0  
Create periodic zone? [yes] yes  
grid-interface name [] fan-periodic
```

- The right hand rule **must** be used when entering the rotation angle value
- With non-conformal periodic boundaries it is not required to specify the periodicity type in the periodic panel



- Ability to **extrude** a **boundary face zone** and extend the solution domain without having to exit the solver
 - A typical application of the extrusion capability is to extend the solution domain when recirculating flow is impinging on a flow outlet
- Current extrusion capability creates **prismatic** or **hexahedral** layers based on the shape of the face and normal vectors to the face zone's nodes.
- New fluid zone is created
- Implemented only in 3D
- Two options available in the TUI:
 - Extrude a face thread by specifying a list of displacements (in SI units).

`define/boundary-conditions/modify-zone/extrude-face-zone-delta`

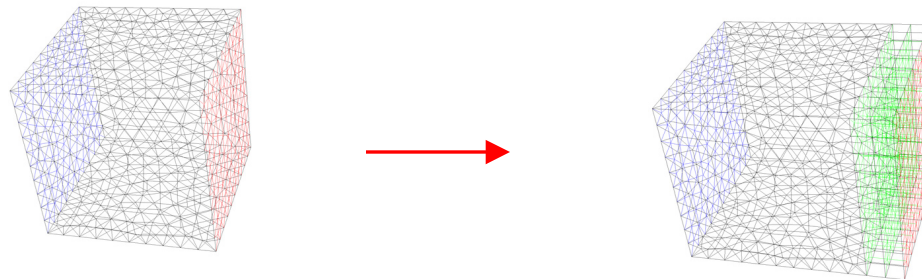
- Extrude a face thread by specifying a total distance (in SI units) and a list of parametric locations between 0 and 1 (e.g. 0., 0.1, 0.3, 0.75, 1.0).

`define/boundary-conditions/modify-zone/extrude-face-zone-para`

Face Extrusion: Example

- A cube (10x10x10) is extruded twice at the outlet by a distance of 1 m

```
/define/boundary-conditions/modify-zones> extrude-face-zone-delta  
Distance delta(1) [()] 1  
Distance delta(2) [()] 1  
Distance delta(3) [()]  
Extrude face zone? [yes]  
Moved original zone (outlet) to interior-7  
Created new prism cell zone fluid-10  
Created new prism cap zone pressure-outlet-11  
Created new prism side zone wall-12  
Created new prism interior zone interior-13
```



Miscellaneous on Mesh Modifications (1)

- How to repair left-handed faces in Fluent ?

/grid/modify-zones/repair-face-handedness

Left-handed cells usually occur with highly skewed cells and/or negative volumes.

- What is **tfilter** ?

A set of utilities used by Fluent/Gambit/Tgrid to perform mesh related modifications, such as:

- Convert mesh to Fluent format
- Convert mesh
- Merge meshes

In recent releases, tfilter has been renamed to utility

Utility can be invoked **manually**. To find all the options:

shell> utility - h

How to convert 2D mesh into 3D surface mesh ?

utility tconv -d2 sample2d.msh 3d-surface.msh

The z-coordinate is assigned as zero in the 3d surface mesh

- How to convert 3D surface mesh into 2d mesh ?

utility tconv -d3 3d-surface.msh sample2d.msh

The z-coordinate is ignored in the 3d surface mesh

- How to merge two meshes ?

utility tmerge3d -cl -p mesh1.msh mesh2.msh combined.msh

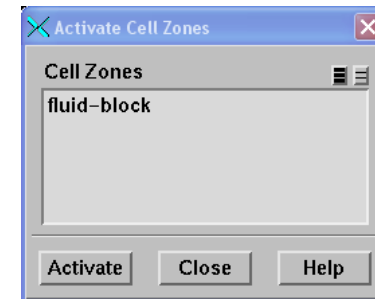
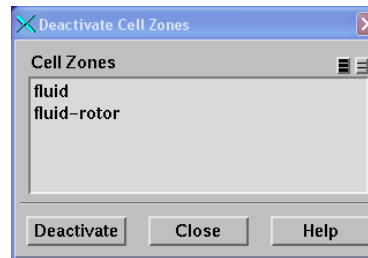
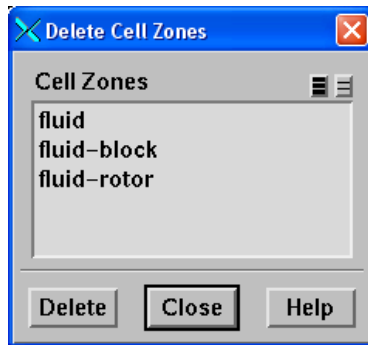
- How to convert CFX mesh to Fluent mesh format ?

utility fe2ram -cl -tCFX -dN -cl -group cfx.geo fl.cas

Miscellaneous on Mesh Modifications (2)

- How to delete, deactivate, and activate cell zone(s) from within Fluent ?

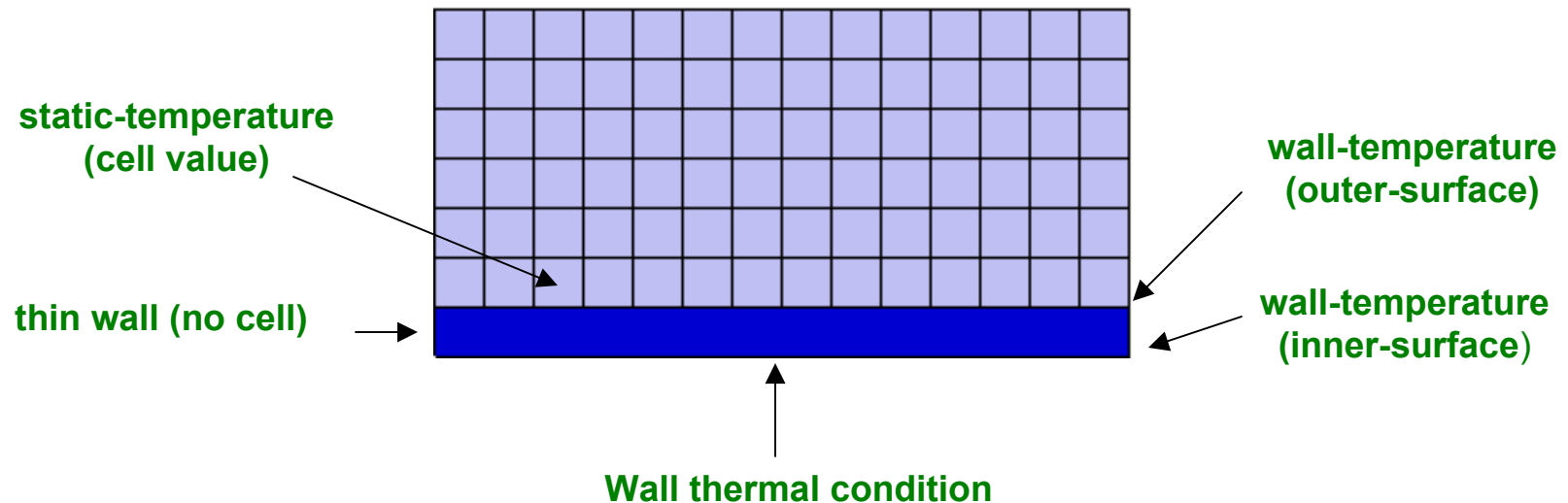
Grid → Zone → Delete ...
 → Deactivate
 → Activate



- The face boundaries of a deactivated cell zone will be changed to wall
- Works only in **serial** (not parallelized yet in Fluent 6.1)
- Useful to isolate highly skewed cells and remove them from computations
- Fluent 6.2 will have the functionality to **Add** new cell zones

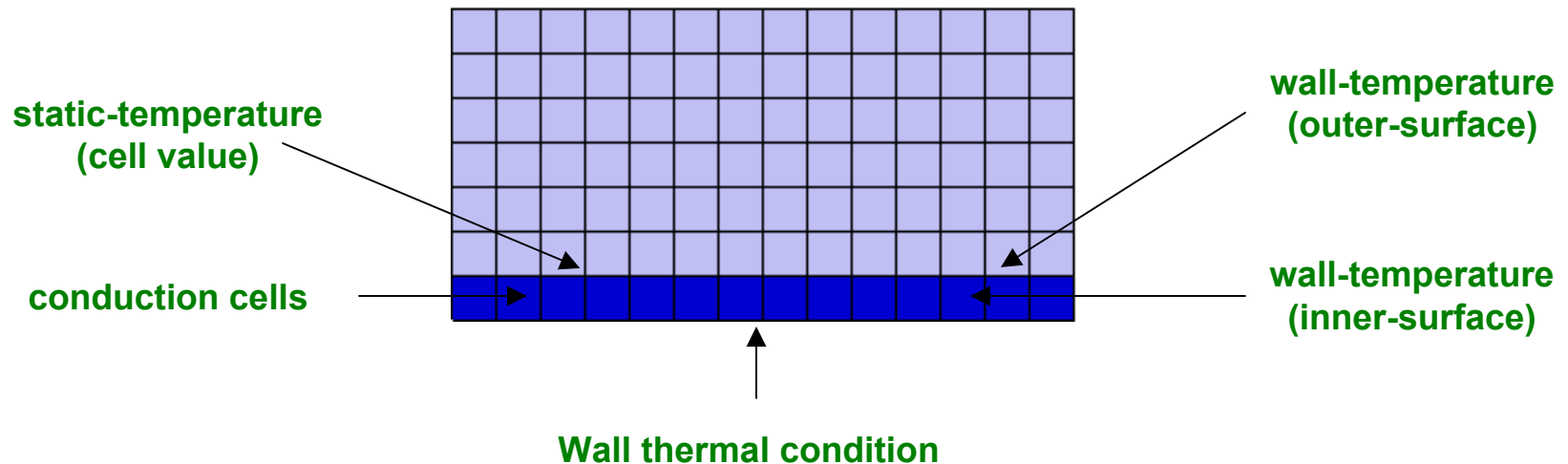
Temperature Definitions for Thin Wall Model

- **Thin wall** model applies **normal conduction** only (no in-plane conduction) and no actual cells are created
- Wall thermal boundary condition is applied at the outer layer



Temperature Definitions for Shell Conduction Zone

- A **thin wall** conduction with both **normal** and **in-plane** conductions
- Actual **conduction cells** are created but can not be displayed and cannot be accessed by UDFs
- Solid properties of the conduction zones must be constant and can not be specified as temperature dependent
- Case must be partitioned in the serial mode before loading into parallel (conduction zones need to be encapsulated)



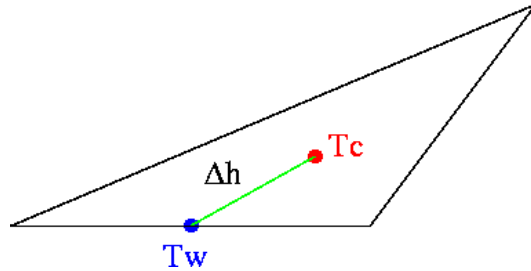
- If the planar conduction zone(s) has complex intersecting planar surfaces, unphysical high/low temperatures can happen at cells near the planar conduction zone. To avoid, use the following scheme command:

(rpsetvar 'temperature/shell-secondary-gradient? #f)

Will drop the accuracy of the diffusive flux computation in the shell conduction zone to first order, but should significantly improve the stability of the temperature field.

Alternate Formulation for Wall Temperature

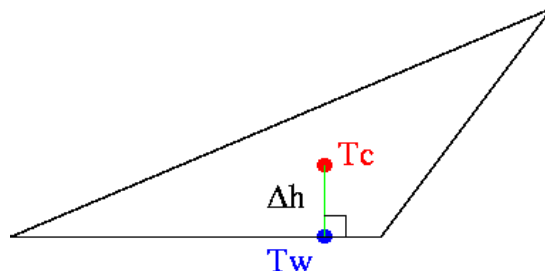
- solve/set/expert/use alternate formulation for wall temperature? **[no]**
 - FLUENT models the flux at the wall as follows:



$$q = k \nabla T \cdot \vec{n}$$

$$q = k \frac{(T_w - T_c)}{\Delta h} + f(\Delta T)$$

- The term $f(\Delta T)$ includes second order terms that need to be determined, which involves approximations
- solve/set/expert/use alternate formulation for wall temperature? **[yes]**
 - Assume face center is defined such as the vector between cell center and face center is perpendicular to the wall



$$q = k \nabla T \cdot \vec{n}$$

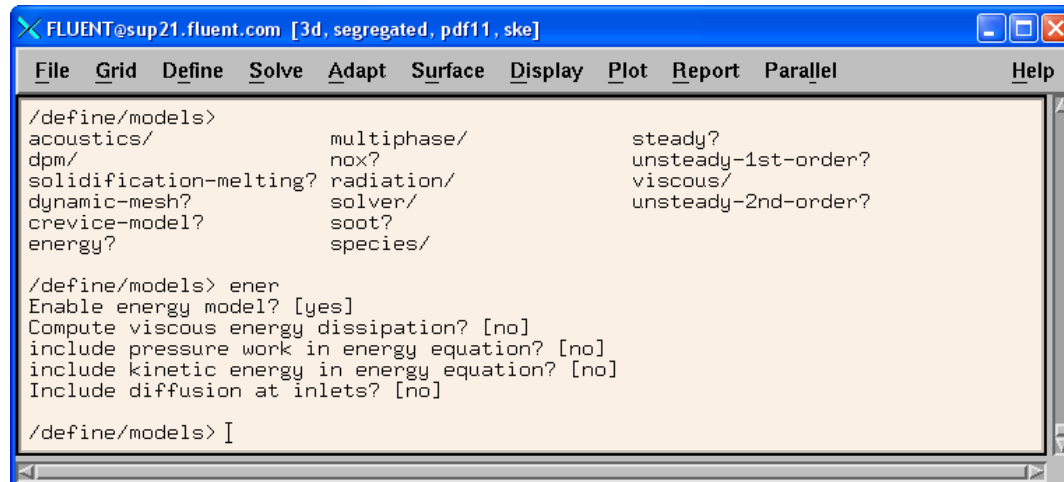
$$q = k \frac{(T_w - T_c)}{\Delta h}$$

- The term $f(\Delta T)$ vanishes and Δh changes
- This option doesn't impact the results if the wall cells are not skewed

define/models/energy?

▪ include diffusion at inlets?

- The net transport of energy at inlets consists of both the **convection** and **diffusion** components
- The convection component is fixed by the user specified inlet temperature
- The diffusion component, however, depends on the gradient of the computed temperature field, and thus is not specified a priori
- The **default** is to include the diffusion of energy at inlets
- To turn off inlet energy diffusion, answer [no] to include diffusion at inlets?
- Available only for the segregated solver



```
FLUENT@sup21.fluent.com [3d, segregated, pdf11, ske]
File  Grid  Define  Solve  Adapt  Surface  Display  Plot  Report  Parallel  Help
/define/models>
acoustics/          multiphase/          steady?
dpm/                nox?                unsteady-1st-order?
solidification-melting? radiation/          viscous/
dynamic-mesh?       solver/              unsteady-2nd-order?
crevice-model?      soot?
energy?             species/

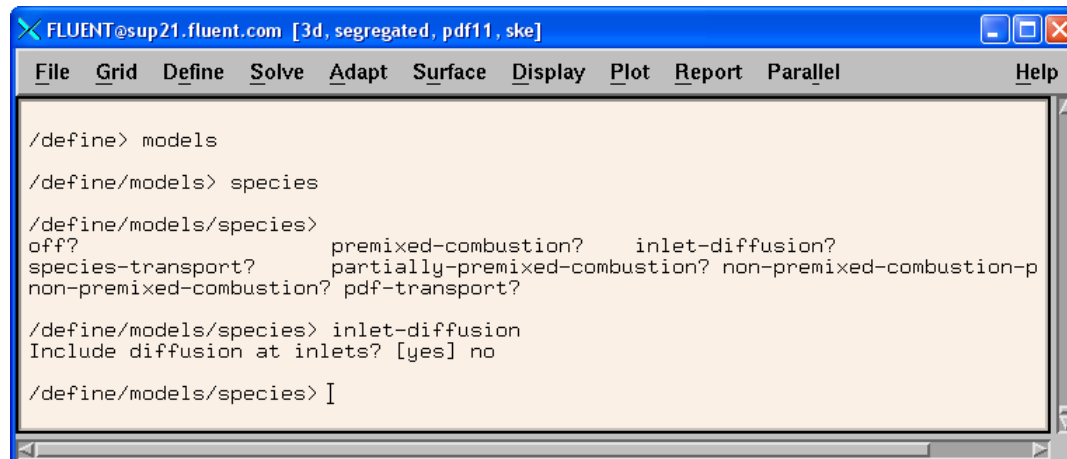
/define/models> ener
Enable energy model? [yes]
Compute viscous energy dissipation? [no]
include pressure work in energy equation? [no]
include kinetic energy in energy equation? [no]
Include diffusion at inlets? [no]

/define/models> I
```


define/models/species/inlet-diffusion?

■ inlet-diffusion?

- The net transport of species at inlets consists of both the **convection** and **diffusion** components
- The convection component is fixed by the user specified inlet species concentration
- The diffusion component, however, depends on the gradient of the computed species concentration field, and thus is not specified a priori
- The **default** is to include the diffusion flux of species at inlets
- To turn off inlet energy diffusion, answer [no] to include diffusion at inlets?
- Available only with the segregated solver



```

FLUENT@sup21.fluent.com [3d, segregated, pdf11, ske]
File  Grid  Define  Solve  Adapt  Surface  Display  Plot  Report  Parallel  Help

/define> models
/define/models> species
/define/models/species>
off?                premixed-combustion?    inlet-diffusion?
species-transport?  partially-premixed-combustion? non-premixed-combustion-p
non-premixed-combustion? pdf-transport?

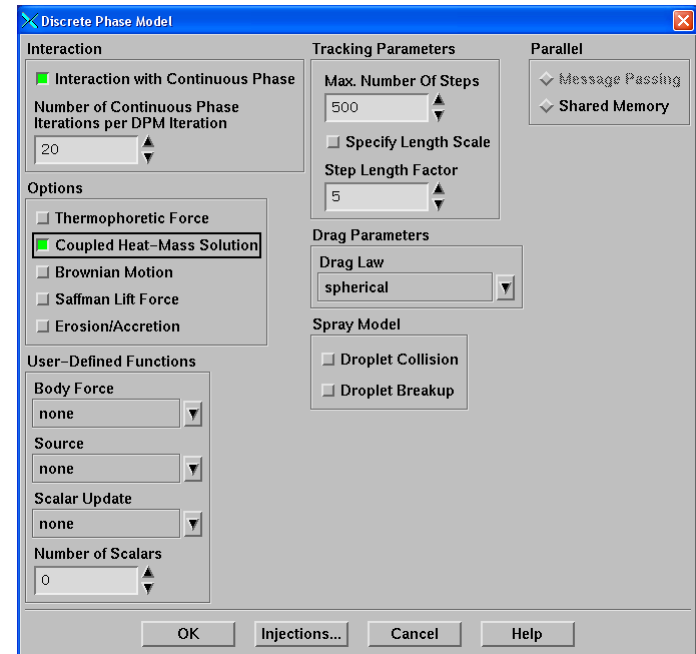
/define/models/species> inlet-diffusion
Include diffusion at inlets? [yes] no

/define/models/species> ]
  
```

TUI/GUI Command for DPM

■ coupled-heat-mass-update

- By default, the solution of the particle heat and mass equations are solved in a segregated manner
- With this option, FLUENT will solve this pair of equations using a stiff, coupled ODE solver with error tolerance control
- It doesn't affect nor accelerate the coupling of the particle source terms to the fluid equations
- This option gives a more accurate temperature and mass content for the particles when having a strong coupling between both equations
- The increased accuracy, however, comes at the expense of increased computational expense

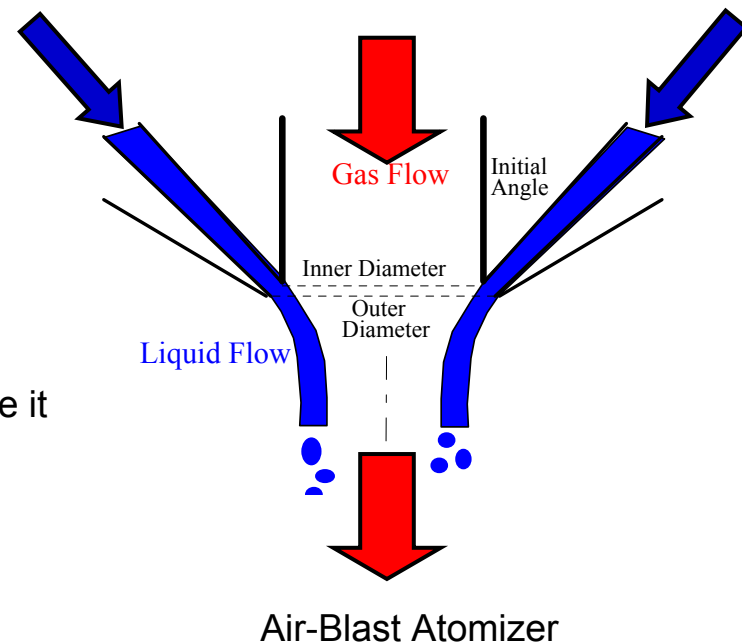


Modify Rosin-Rammler parameters for atomizers (Solutions 666 & 908)

- By default, the **spread parameter** for the Rosin Rammler distribution is 3.5 for all the atomizer models. Is it possible to change this parameter ?
- Can alter this spread parameter through the Text User Interface by executing:

```
(rpsetvar 'dpm/atomizer-spread-param 3.9)
```

- For **effervescent atomizer**, the spread parameter is hardwired, and can not be changed
- This rp-variable command will work for all atomizers in FLUENT 6.2
- What about the **dispersion angle** in the Flat-Fan atomizer model ?
- The dispersion angle cannot be altered
- It is hardwired to a value of 6 degrees
In FLUENT 6.2, there will be an rp-variable to change it



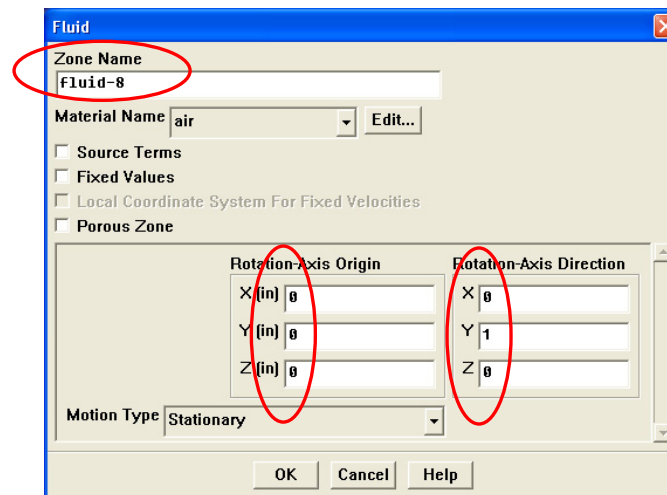


This presentation provides Tips and Tricks:

- For IO and Batch
- For Case Set-up and Mesh
- **For Solving**
- For Post-Processing
- For Reporting

Calculate Radial, Axial and Tangential velocities (Solution 562)

- Only velocities in **Cartesian** coordinates (**Ux,Uy,Uz**) are accessible through the UDF macros
- The **radial**, **tangential**, and **axial** velocities (**Ur, Ut, Ua**) within a fluid zone can be computed using a UDF
- An example: implement an energy source term that is a function of the radial velocity



- Each fluid zone will have a user specified **axis origin** and **axis direction** vectors
- These two vectors can be queried in UDF by using special macros
- Knowing these two vectors, the radial, tangential, and axial vector (**Er,Et,Ea**) can be computed and the cartesian velocities can be dotted with this vector to obtain the radial, tangential, and axial velocities within that fluid zone
- The selected fluid in the reference panel has no bearing on the macros**

Calculate Radial, Axial and Tangential velocities (Solution 562)

```
#include "udf.h"
#define FACTOR -950.0

DEFINE_SOURCE(cell_cold, cell, thread, dS, eqn)
{
    real NV_VEC(origin), NV_VEC(axis);
    real NV_VEC(V), NV_VEC(r), NV_VEC(R), NV_VEC(B);
    real NV_VEC(er), NV_VEC(et);
    real xc[ND_ND];
    real Bmag, rmag, source;
    real ua, ur, ut;

    /* Get origin vector of fluid region */
    NV_V(origin, =, THREAD_VAR(thread).cell.origin);
    /* Get axis of fluid region */
    NV_V(axis, =, THREAD_VAR(thread).cell.axis);
    /* Store the 3 Cartesian velocity components in vector V */
    N3V_D(V, =, C_U(cell,thread), C_V(cell,thread), C_W(cell,thread));

    /* Get current cell coordinate */
    C_CENTROID(xc, cell, thread);

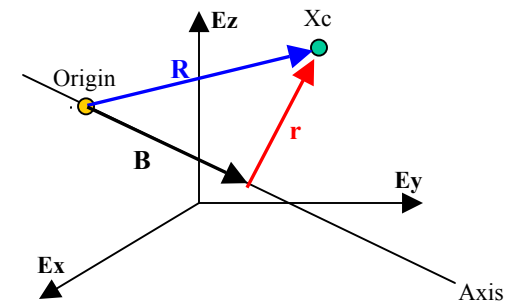
    /* Calculate (R) = (Xc)-(Origin) */
    NV_VV(R, =, xc, -, origin);
    /* Calculate |B| = (R) dot (axis) */
    Bmag = NV_DOT(R, axis);
    /* Calculate (B) = |B|*axis */
    NV_VS(B, =, axis, *, Bmag);
    /* Calculate (r) = (R)-(B) This is the local radial vector */
    NV_VV(r, =, R, -, B);
    /* Calculate |r| */
    rmag = NV_MAG(r);
```

```
    if (rmag != 0.)
    {
        NV_VS(er, =, r, /, rmag);
        NV_CROSS(et, axis, er);
        ur = NV_DOT(V, er);
        ut = NV_DOT(V, et);
        ua = NV_DOT(V, axis);
    }
    else
    {
        ur = 0.0;
        ut = 0.0;
        ua = NV_DOT(V, axis);
    }

    /*source term */
    source = FACTOR*ur;

    /* derivative of source term w.r.t. enthalpy */
    dS[eqn] = 0.0;

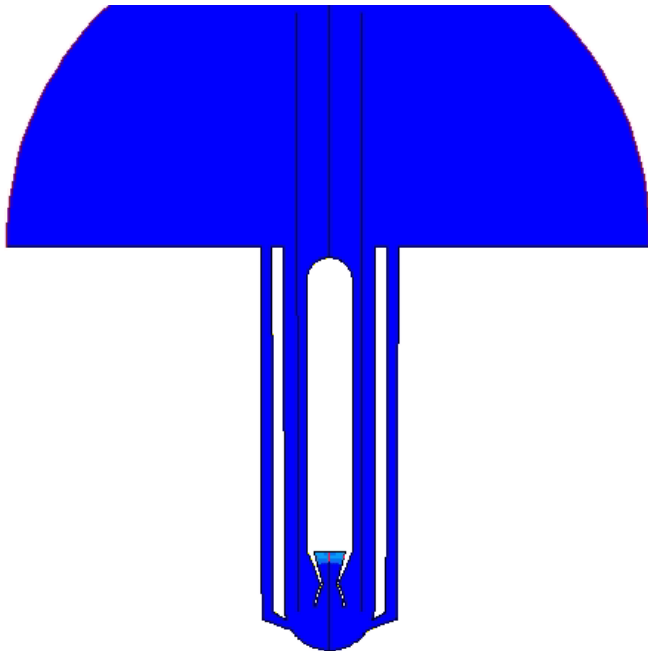
    return source;
}
```



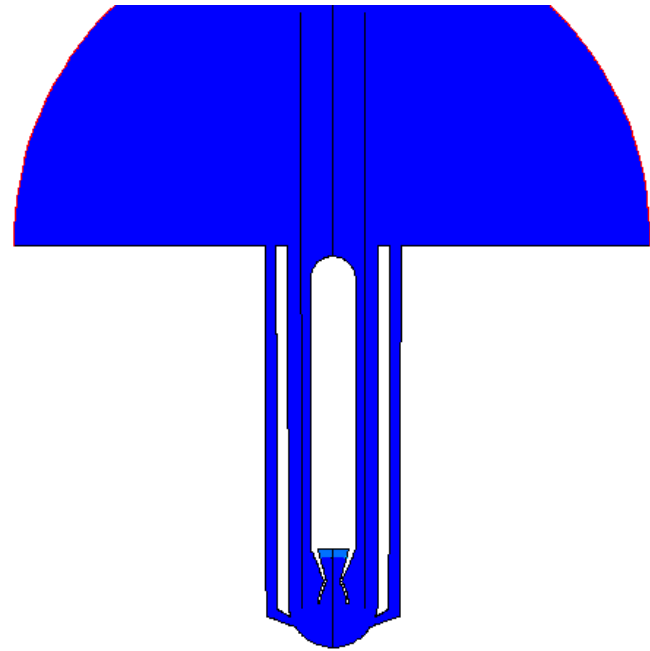
- **Unsteady** DPM injection releases the particle packets at the **beginning of every timestep**
 - For problem with large elapsed time and small timestep size, many particles will need to be tracked inside the computational domain
 - Tracking large number of particles is **computationally expensive** (RAM and CPU time)
- A process can be developed to inject particles at a certain **user specified time interval** so as to reduce the number of particles that need to be tracked inside the domain
- The process will **inject** the particles at the beginning of a timestep and then **store/accumulate** the particles' mass during the next timesteps when injections are turned off
- At the time of the next injection, the accumulated particles' mass will be injected
- **There will be fewer particles but the particles's mass will be preserved**

Staggering Releases of Particles for Unsteady Problems (Solution 1017)

Example: unsteady silo launch with solid particles injected at nozzle exhaust



Default Unsteady Injection



Staggered Unsteady Injection

Staggering Releases of Particles for Unsteady Problems (Solution 1017)

```
#include "udf.h"
#define RELEASE_STEP 5e-3
static real sum_inj = 0.0;

DEFINE_ADJUST (update, domain)
{
    real pflow;

    if ( first_iteration )
    {
        pflow = get_mdott_prt(tm);
        sum_inj += mdott*tmshps;
    }
}
```

Get the current particle flowrate & store for later release

```
DEFINE_DPM_INJECTION_INIT (init_prt_tm, I)
{
    Particle *p;
    real tm, tmshps, flowrate;

    tm = RP_Get_Real("flow-time");
    tmshps = RP_Get_Real("physical-time-step");
    flowrate = sum_inj/tmshps;

    loop(p,I->p_init)
    {
        p->flow_rate = flowrate;
    }

    if ((tm+tmshps) >= I->unsteady_start)
        I->unsteady_start += RELEASE_STEP;

    sum_inj = 0.0;
}
```

Move unsteady_start forward in time for next release

Reset storage for mass of particles

- For restart purpose, need to write the current values of **sum_inj** and **last injection time** to the case/data file
- Can use **DEFINE_RW_FILE**

Optimizing Timestep Size for Coupled Dynamic Mesh Problems (Solution 1018)

- **Coupled dynamic mesh** problem adjusts the motion of the moving body/surfaces based on the current computed aerodynamic load and applied external forces
- An optimum 'mean' timestep size that will apply generally for the duration of the coupled motion is difficult to obtain since the aerodynamic load is not known a priori
- **Continual** and **manual** adjustment of the timestep size is required to avoid using an excessively **conservative value** that will **increase running time** or **too large** value that will cause the **dynamic remeshing to fail**
- It is possible to implement a **user defined variable timestep size** using UDF
- The variable timestep size is computed subject to the following constraints:
 - User specifies **maximum allowable translational distance**
 - User specifies **maximum allowable timestep size value**
- At the end of each timestep:
 - The next timestep size and the corresponding velocity of the body are computed such that the body will move by the maximum allowable user specified translational distance
 - A cap on the timestep size is necessary to prevent an excessively large computed timestep size which can result in divergence
- **The maximum allowable translational distance can be varied as function of time, if necessary**

Optimizing Timestep Size for Coupled Dynamic Mesh Problems (Solution 1018)

- A relation is needed to solve for both the timestep size and velocity of the body which satisfy the constraint of the specified translational distance of the body
- This relation is obtained by solving a quadratic equation derived from the following two equations:

$$h_{max} = V^{n+1} \times dt$$

$$\frac{F}{m} = \frac{V^{n+1} - V^n}{dt}$$

h_{max} = maximum allowable distance traveled

$\frac{F}{m}$ = force_acting_on_body/mass_of_body

V^n = body velocity at previous timestep

V^{n+1} = body velocity at next timestep

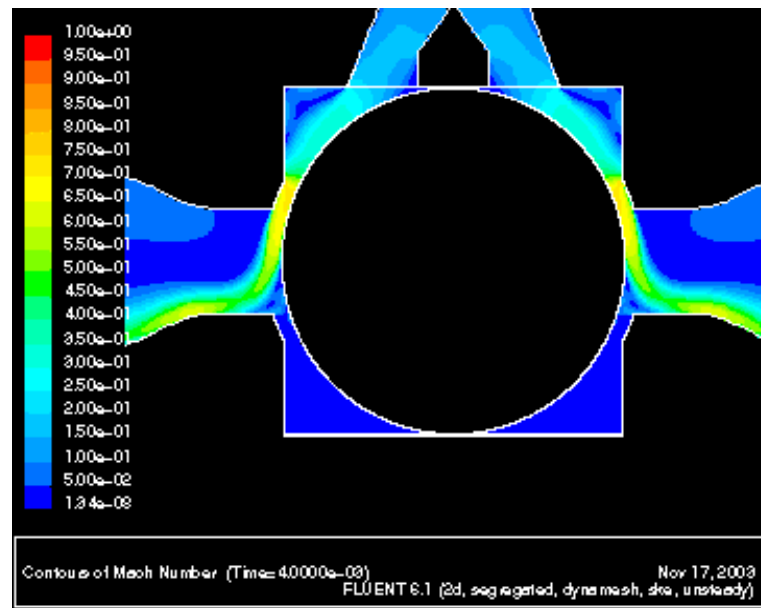
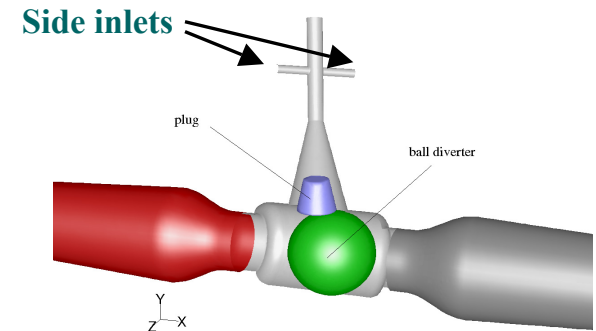
dt = next timestep size

- The above represents two equations which can be solved for the two unknowns, V^{n+1} and dt .

Optimizing Timestep Size for Coupled Dynamic Mesh Problems (Solution 1018)


Applied to Dynamic Ball Diverter Valve

- Dynamic ball diverter valve problem requires coupled dynamic mesh model
 - Motion of the ball is controlled by varying the massflow rates of the side inlets
 - Motion of the ball is not known apriori
- Optimum timestep size is difficult even to guess



Mach contour

Optimizing Timestep Size for Coupled Dynamic Mesh Problems (Solution 1018)



```
#include "udf.h"
#include "sg_mem.h"
#include "dynamesh_tools.h"

#define zoneID 28 /* zone ID for the ball */
#define b_mass 1.0 /* mass of the ball */
#define dtm_mx 0.005 /* maximum delt */
#define hmove 0.001 /* maximum distance */

real V_ball = 0.0;
real b_ctr = 0.0; /* center location of the ball */

DEFINE_EXECUTE_AT_END(exec_end)
{
    real dtm, Velx, Vn, Fm;
    real x_cg[3], f_glob[3], m_glob[3];
    Domain *domain = Get_Domain(1);
    Thread *tf = Lookup_Thread(domain,zoneID);

    /* Get the previous c.g. for the ball zone */
    x_cg[0] = b_ctr;

    /* Compute the forces on the ball */

    Compute_Force_And_Moment(domain,tf,
                             x_cg,f_glob,m_glob,TRUE);
```

```
/* Compute velocity and timestep size */

Vn = V_ball;
Fm = f_glob[0]/b_mass;
dtm = ( -fabs(Vn) +
        sqrt( Vn*Vn + 4.0*fabs(Fm)*hmove ) ) /
        ( 2.0*fabs(Fm) );
if ( dtm > dtm_mx ) dtm = dtm_mx;
Velx = Vn + Fm*dtm;

/* Update velocities, delt, and ball c.g. location */

tmsize = dtm;
V_ball = Velx;
b_ctr += V_ball*tmsize;

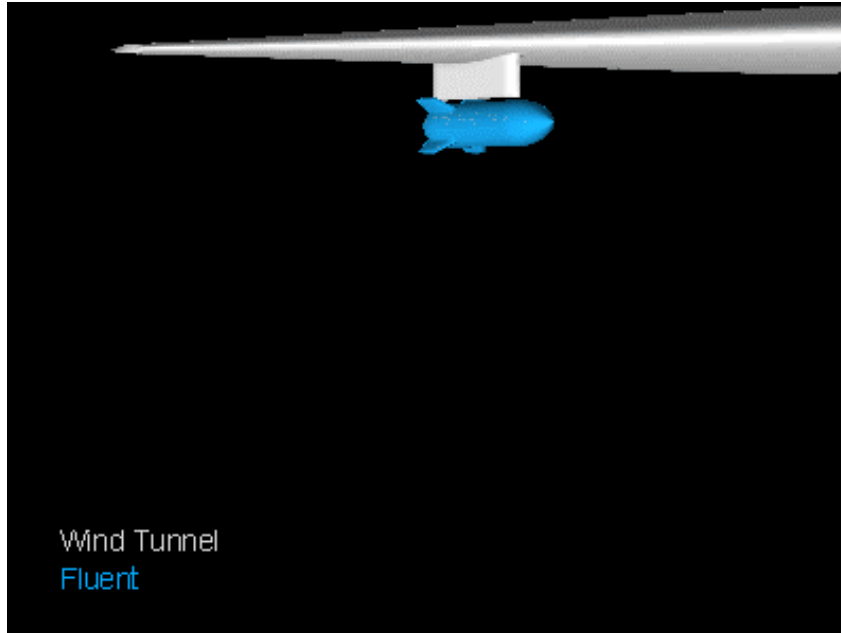
RP_Set_Real("physical-time-step",tmsize);
}

DEFINE_CG_MOTION(valve_motion, dt, cg_vel,
                 cg_omega, time, dtime)
{
    NV_S(cg_vel, =, 0.0);
    NV_S(cg_omega, =, 0.0);

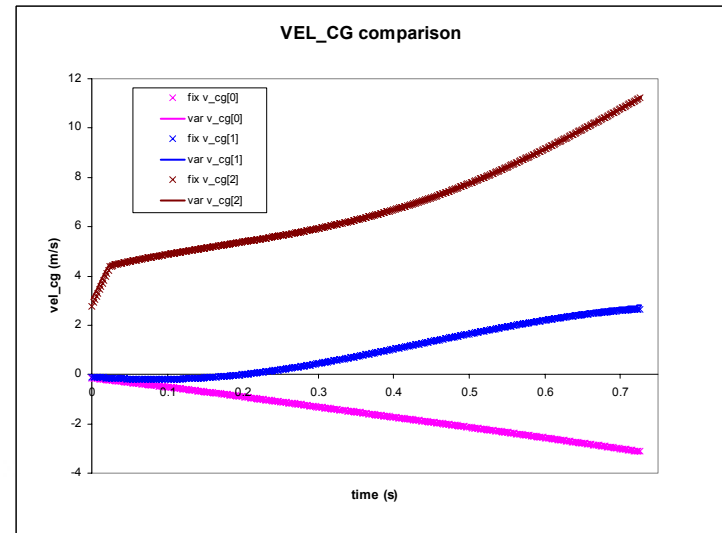
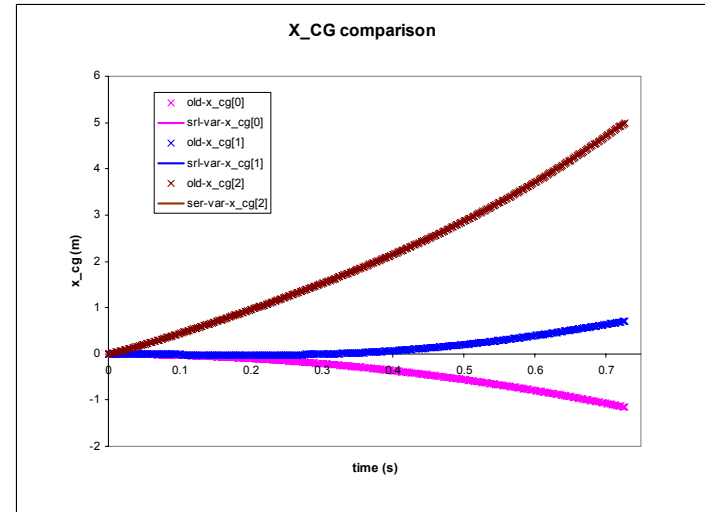
    cg_vel[0] = V_ball;
}
```

Optimizing Timestep Size for Coupled Dynamic Mesh Problems (Solution 1018)

Same approach as applied to the 6 DOF MDM UDF

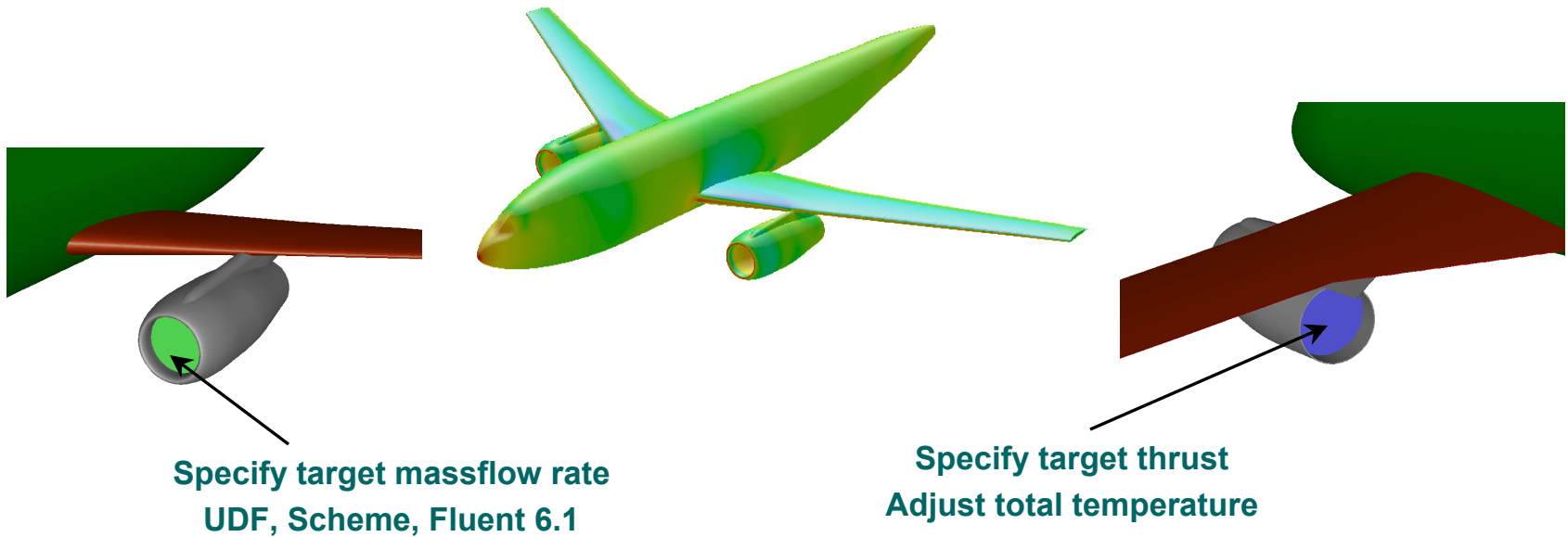


- Time to solution with fixed timestep size on a single CPU is **3 days**
- With variable timestep size is about **2 days**



Achieving a Target Thrust at Nozzle Exit (Solution 718)

- External aero configuration of a full aircraft with wing-body-pylon-nacelle requires the specifications of **target massflow rate** at the nacelle inlet and **target thrust** at the nozzle exit
- Target massflow rate is specified using massflow-outlet boundary type (UDF, Fluent 6.1).
- Target thrust can be achieved by **manually** adjusting the **inlet total-temperature**



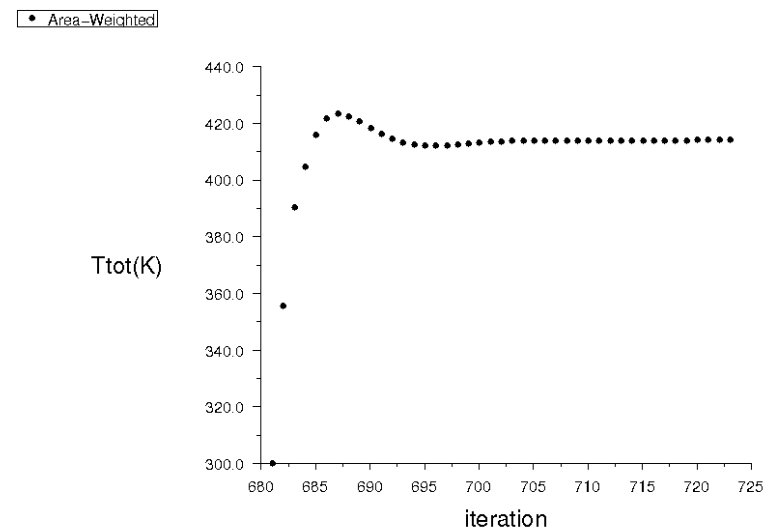
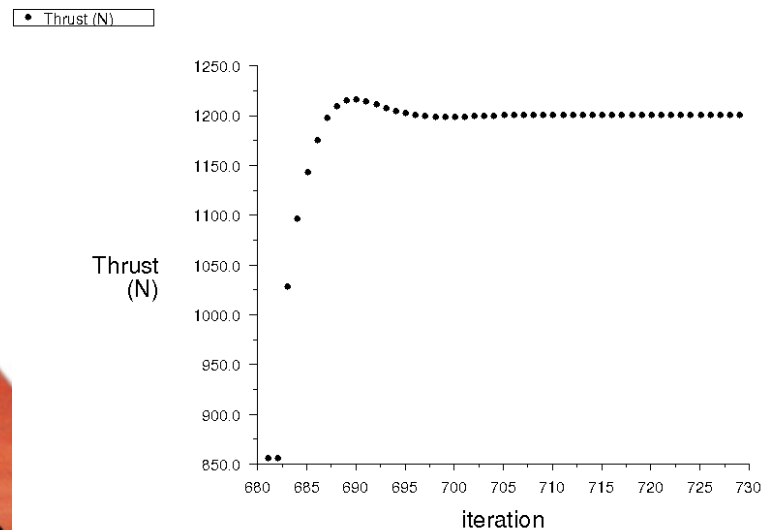
- The user manual adjustment can be automated by a UDF
 - Total temperature adjustment at each iteration is computed based on the equation relating the total temperature to the thrust

Achieving a Target Thrust at Nozzle Exit (718)

- A calculated update of the total temperature to achieve the target thrust is needed to ensure fast and robust procedure (as opposed to using a shooting method)

$$\left. \begin{aligned} Th &= \dot{m} V_e & V_e^2 &= \frac{Th}{\rho A} \\ \frac{T_t}{T} &= 1 + \frac{\gamma-1}{2} \frac{V_e^2}{a^2} \end{aligned} \right\} \rightarrow \frac{T_t}{T} = 1 + \frac{\gamma-1}{2} \frac{1}{\rho A a^2} Th \rightarrow \frac{dT_t}{T} = 1 + \frac{\gamma-1}{2} \frac{1}{\rho A a^2} dTh$$

- An example showing the convergence of the process starting with thrust of **880N** to achieve a target thrust of **1200N** (30 iterations)



Achieving a Target Thrust at Nozzle Exit (718)

```
#include "udf.h"
#define Gamma 1.4
real Ttot_new = 200.0;

DEFINE_ADJUST (Thrust_compute, domain)
{
    real Thrust_target, alpha, Ttot_min, Ttot_max;

    Thrust_target = 1200.0; /* Target thrust in Newton */
    alpha = 0.5; /* Relaxation */
    Ttot_min = 20.0; /* Min Ttot in K */
    Ttot_max = 1000.0; /* Max Ttot in K */

    zoneID = THREAD_ID(tf);

    Ttot_new = dTCompute(Thrust_target,alpha,
                        Ttot_min,Ttot_max,zoneID);
}

DEFINE_PROFILE (Thrust_fix, tf, position)
{
    face_t f;
    begin_f_loop(f,tf)
        F_PROFILE(f,tf,position) = Ttot_new;
    end_f_loop(f,tf)
}
```

```
real dTCompute(real Thrust_target, real alpha,
               real Ttot_min, real Ttot_max, int zoneID)
{
    face_t f;
    real Temp, Ttot, VsoundSq, rho, area, Thrust_present, dTot;
    Domain *domain = Get_Domain(1);
    Thread *tf = Lookup_Thread(domain,zoneID);

    /* Compute zone average values */
    begin_f_loop(f,tf)
    { .... }
    end_f_loop(f,tf)
    Temp = ... ;    Ttot = ... ;    VsoundSq = ... ;
    rho = ... ;    area = ... ;    Thrust_present = ... ;
    /* Finish */

    dTot = 1.0 + 0.5*(Gamma-1.0)/ ( rho*area*VsoundSq ) *
        (Thrust_target - Thrust_present )/
    dTtot = alpha*Temp*dTot;

    Ttot_new = Ttot + dTtot;

    if ( Ttot_new >= Ttot_max ) Ttot_new = Ttot_max;
    if ( Ttot_new <= Ttot_min ) Ttot_new = Ttot_min;

    return Ttot_new;
}
```

Miscellaneous on UDF (1)

- Define Adjust UDF is called at the beginning of every iteration even for unsteady solver.
- For unsteady runs, there are instances where it is desirable for the Adjust UDF to be called only **at the beginning of the first iteration of every time-step**.
- Can use the macro *first-iteration*, example:

```
DEFINE_ADJUST(myadjust, domain)
{

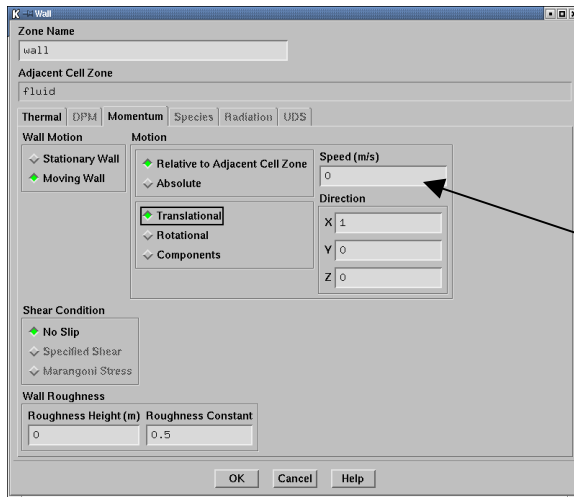
    if (first_iteration)
    {
        /* Do procedures to be executed only at the first
           iteration of every timestep */
    }

}
```

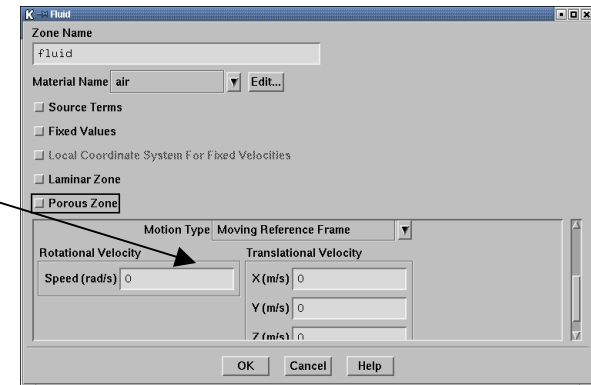
- An alternative is to use the **Execute At End** UDF which is called at the end of every iteration (steady) or timestep (unsteady).

Miscellaneous on UDF (2)

- Certain fields inside the FLUENT model panel do not have built-in UDF hookups, thus it is difficult to specify user specified time/iteration varying values
- Examples are the wall and fluid speeds
- Is there any workaround ?



No UDF hookup



- Can write a **DEFINE_ADJUST** UDF and directly access the macro of the variable of interest (contact support engineer for variable names)

```
THREAD_VAR(tc).fluid.origin[0]  
THREAD_VAR(tc).fluid.origin[1]  
THREAD_VAR(tc).fluid.origin[2]
```

```
THREAD_VAR(tc).fluid.axis[0]  
THREAD_VAR(tc).fluid.axis[1]  
THREAD_VAR(tc).fluid.axis[2]
```

```
THREAD_VAR(t1).fluid.velocity[0]  
THREAD_VAR(t1).fluid.velocity[1]  
THREAD_VAR(t1).fluid.velocity[2]
```

```
THREAD_VAR(tc).fluid.omega
```

```
THREAD_VAR(tf).wall.origin[0]  
THREAD_VAR(tf).wall.origin[1]  
THREAD_VAR(tf).wall.origin[2]
```

```
THREAD_VAR(tf).wall.axis[0]  
THREAD_VAR(tf).wall.axis[1]  
THREAD_VAR(tf).wall.axis[2]
```

```
THREAD_VAR(tf).wall.translate_mag  
THREAD_VAR(tf).wall.translate_dir[0]  
THREAD_VAR(tf).wall.translate_dir[1]  
THREAD_VAR(tf).wall.translate_dir[2]
```

```
THREAD_VAR(tf).wall.omega
```

```
THREAD_VAR(tf).wall.u  
THREAD_VAR(tf).wall.v  
THREAD_VAR(tf).wall.w
```

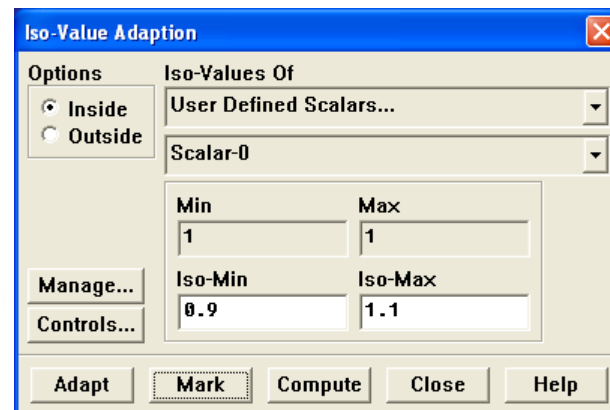
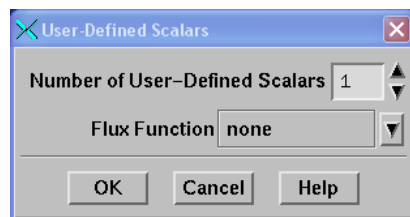


This presentation provides Tips and Tricks:

- For IO and Batch
- For Case Set-up and Mesh
- For Solving
- For Post-Processing
- For Reporting

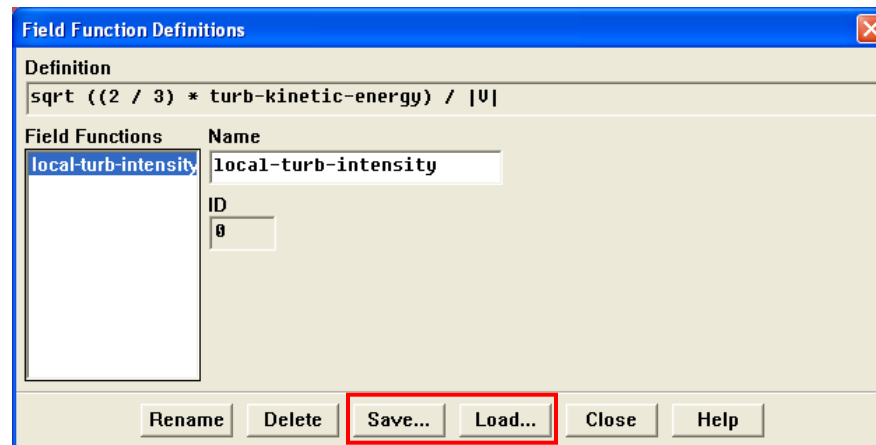
Saving Adaption Register for subsequent use (Solution 617)

- The adaption register information is **not stored** either in case or data files. The information is lost upon saving and reopening the case/data files
- There is no direct way to preserve the register for subsequent use
- To preserve the information in the register, create a User Defined Scalar (UDS) storage, initialize/patch the UDF value for all the cells in the domain with 0, and then patch the value of 1 in all the cells of the register. The overall procedure is:
 1. Generate the register by any means (Boundary, Gradient, Iso-Value, Region...)
 2. Define a UDS
 3. Patch the UDS with 0 for all cells in the domain
 4. Patch the UDS with 1 for all the registers that you want to keep (use the number 2, 3, 4... if you have a second, third, fourth... register, respectively)
 5. When the case is saved, the UDS will remain in the database
 6. Open the case again and regenerate the register by using **Iso-Value Adaption** with UDS values between 0.9 and 1.1



Custom Field Functions

- Custom field functions are **GUI based**, it is not possible to create them through journal files
- If a set of existing custom field functions are to be used for different cases:
 - Write the existing functions as a scheme file using:
 - **Define → Custom Field Functions → Manage → Save** GUI
 - **file/write-field-functions** TUI
 - Read the scheme file to the new case using:
 - **Define → Custom Field Functions → Manage → Load** GUI
 - **file/read-field-functions** TUI



Customizing the Color Map ? (1)

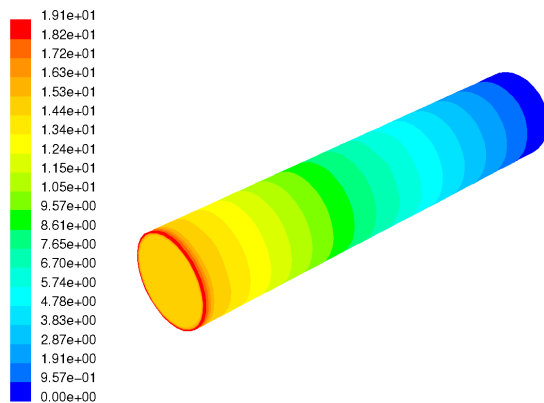
- It is possible to read a custom color map or write an existing one
- The procedure consists of:
 - Loading the Scheme file:


```
(load "rw-colormap.scm")
```
 - Reading a new color map:

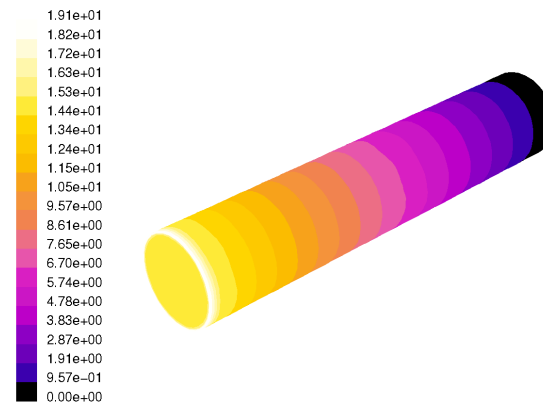

```
/file/read-colormap
```
 - Writing an existing color map:


```
/file/write-colormap
```
- Example: Static pressure variation for a simple duct.

	R	G	B
("thermacam"			
(0.0	0.000	0.000	0.000)
(0.083	0.055	0.000	0.467)
(0.167	0.306	0.000	0.592)
(0.250	0.545	0.000	0.616)
(0.333	0.725	0.016	0.584)
(0.417	0.824	0.114	0.455)
(0.500	0.898	0.267	0.098)
(0.583	0.945	0.404	0.012)
(0.667	0.973	0.545	0.000)
(0.750	0.996	0.702	0.000)
(0.833	0.996	0.847	0.047)
(0.917	1.000	0.945	0.455)
(1.000	1.000	1.000	0.976)
)			



Default FLUENT color map



Thermacam color map

Customizing the Color Map ? (2)

rw-colormap.scm

```
(define (write-cmap fn)
  (let ((port (open-output-file (cx-expand-filename
    fn))))
    (cmap (cxgetvar 'def-cmap)))
    (write (cons cmap (cx-get-cmap cmap)) port)
    (newline port)
    (close-output-port port)))

(define (read-cmap fn)
  (if (file-exists? fn)
    (let ((cmap (read (open-input-file (cx-expand-
      filename fn)))))
      (cx-add-cmap (car cmap) (cons (length (cdr
        cmap)) (cdr cmap)))
      (cxsetvar 'def-cmap (car cmap)))
    (cx-error-dialog
      (format #f "Macro file ~s not found." fn))))

(define (ti-write-cmap)
  (let ((fn (read-filename "colormap filename"
    "cmap.scm")))
    (if (ok-to-overwrite? fn)
      (write-cmap fn))))
```

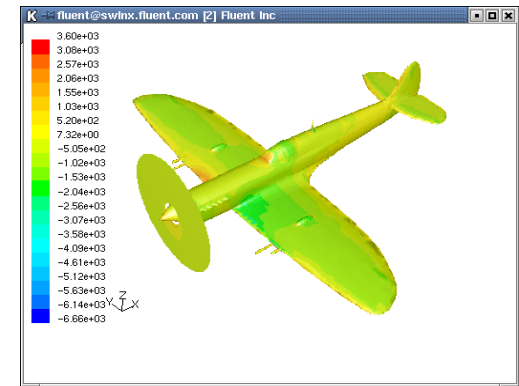
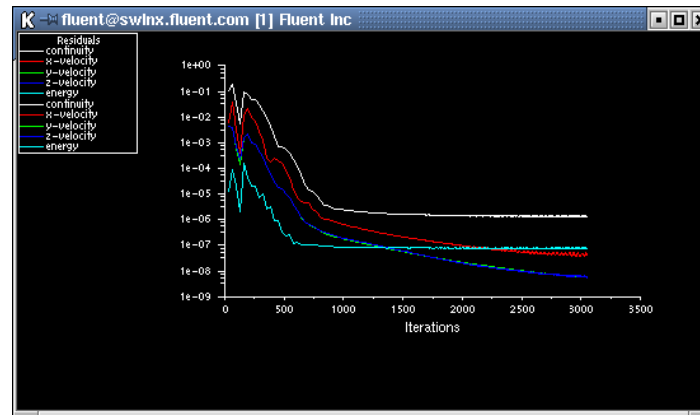
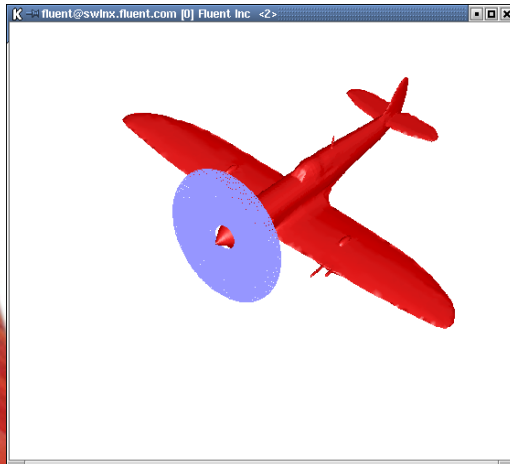
```
(define (ti-read-cmap)
  (read-cmap (read-filename "colormap filename"
    "cmap.scm")))

(ti-menu-insert-item!
  file-menu
  (make-menu-item "read-colormap" #t ti-read-cmap
    "Read a colormap from a file.))

(ti-menu-insert-item!
  file-menu
  (make-menu-item "write-colormap" #t ti-write-
    cmap
    "Write a colormap to a file.))
```

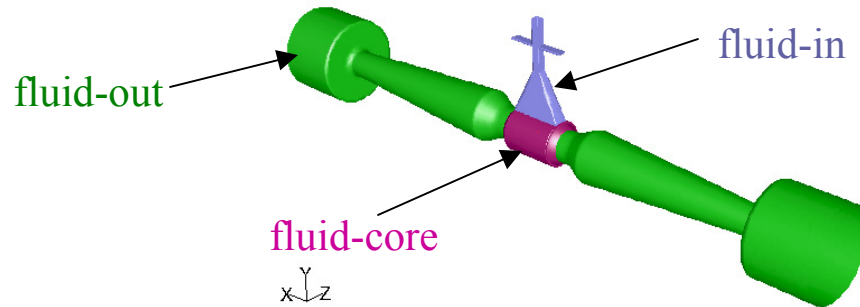
How to save graphics/GUI layout ?

- To save custom window layout, can use the GUI command:
File → Save Layout
- Sizes and locations of currently defined windows (0,1,2,...)** as well as **other windows** for iteration panel, display panel, etc that have been opened will be saved in **.cxlAYOUT** located at the same location as the **.fluent** file



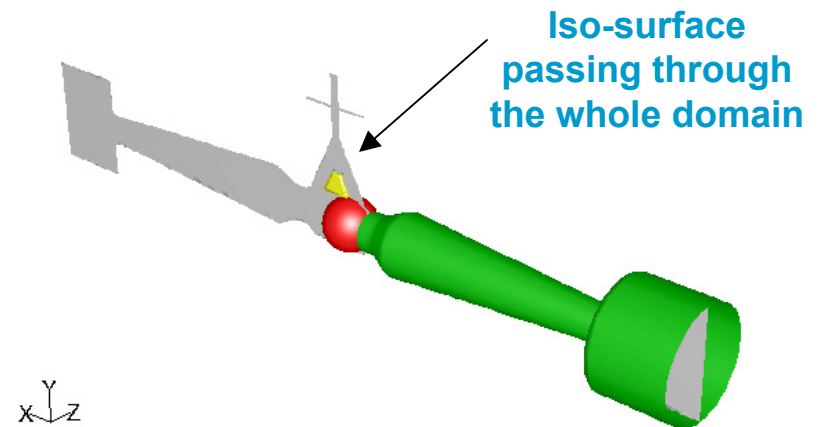
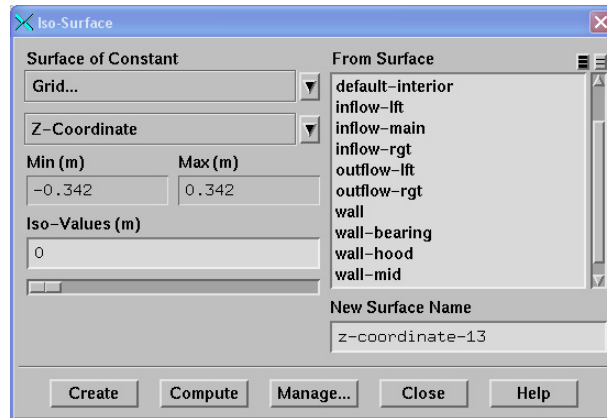
How to create an iso-surface that passes through a selected cell zone?

- The domain has several fluid zones and the objective is to create an iso-surface that passes through one specific fluid zone



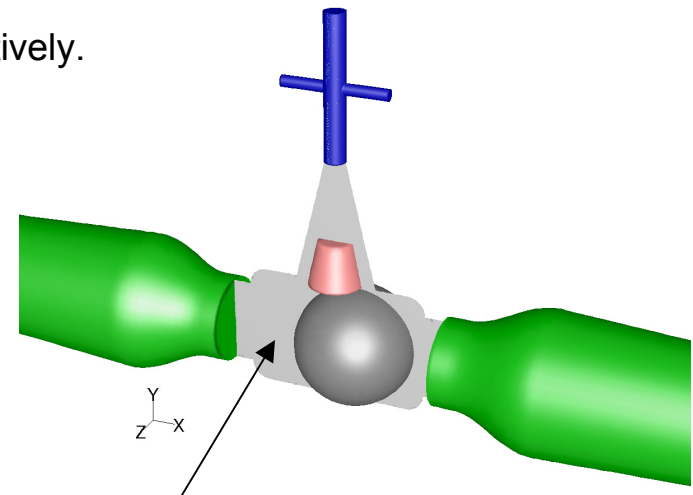
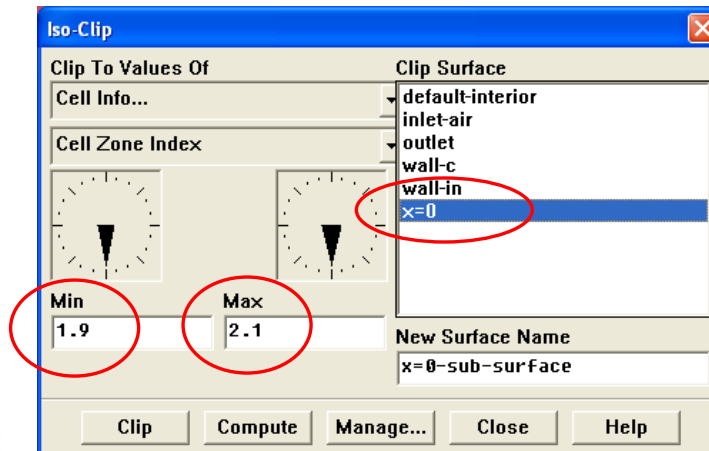
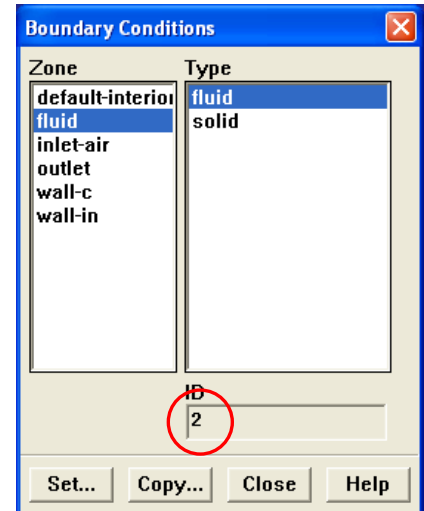
- Create an iso-surface that passes through the entire domain

Surface → Iso-Surface...



How to create an iso-surface that passes through a selected cell zone?

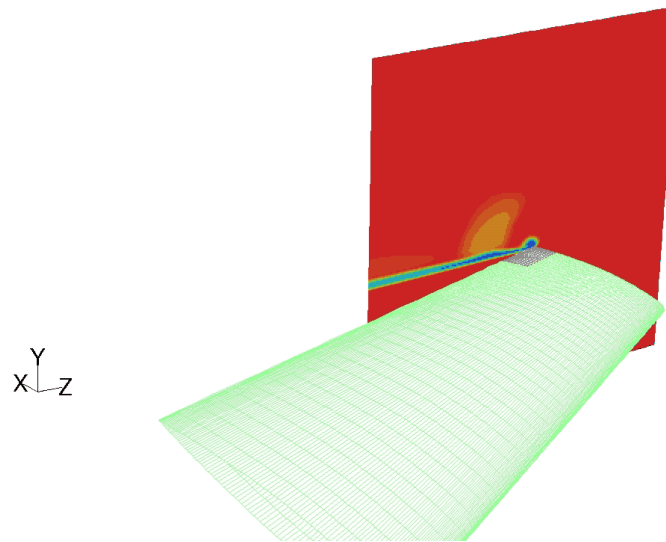
- From the boundary condition panel, determine the Cell Zone ID (same as the Cell Zone Index) where the iso-surface is to be retained. Let's say the Cell Zone ID is 2.
- Clip the iso-surface using to the desired ID:
 - Surface → Iso-Clip...**
 - Under "Clip To Values Of" select "Cell Info" and "Cell Zone Index"
 - Under "Clip Surface" select the surface created in first step
 - Enter 1.9 and 2.1 for "Min" and "Max", respectively.
 - Specify the name and click on "Clip"



iso-clip passing
through fluid-core
only

Creating Sweep Surface Animation (Solution 870)

- **Display → Sweep Surface...** allows a sweep surface to be defined along an axis and the resulting sweep animation displayed in the graphics window
- No functionality within this panel to save the animation as hardcopy files



Total pressure at tip wake for the M6 wing

- It is possible to create animation files for the sweep surface using **Scene Animation**

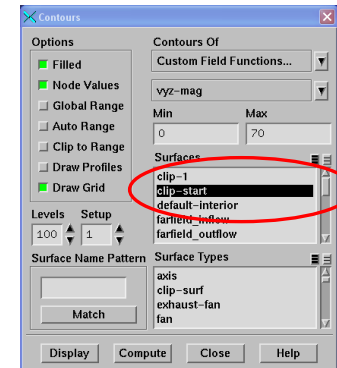
Creating Sweep Surface Animation (Solution 870)

- Create the surface where the animation sweep will start.
Can use iso-surface/iso-clip-surface but not bounded-plane-surface. Surface must be **normal** to any of the coordinate axis (e.g. x-axis or yz-plane).

- Display the contour/vector variable on this surface

Display → Contours...

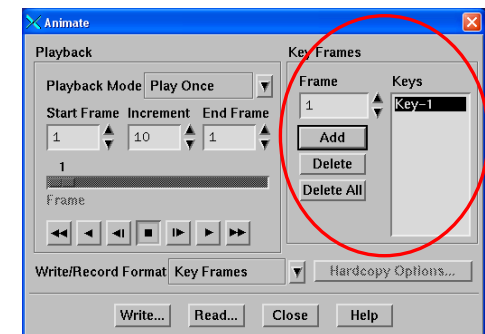
Set the appropriate contour range and grid display



- Set this contour/vector display as the first key frame

Display → Scene Animation...

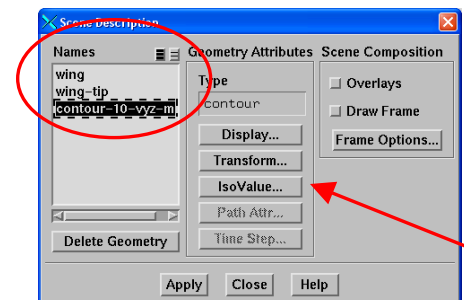
Click on the Add button to enter the frame



- Setup the final frame of the animation sweep

Display → Scene...

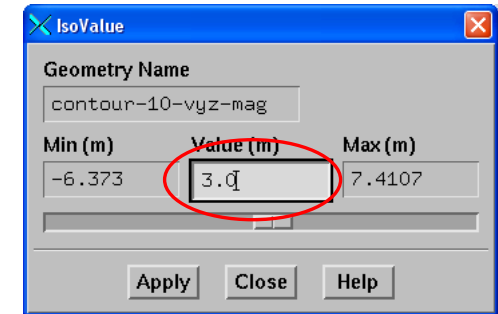
Under **Names**, select the contour and then click on the **IsoValue** button



Creating Sweep Surface Animation (Solution 870)

- Inside the IsoValue panel, enter the final value of the frame location along the chosen axis (e.g. x-axis)

Click on the Apply button and the surface will be automatically moved to the final location and the contour/vector display updated

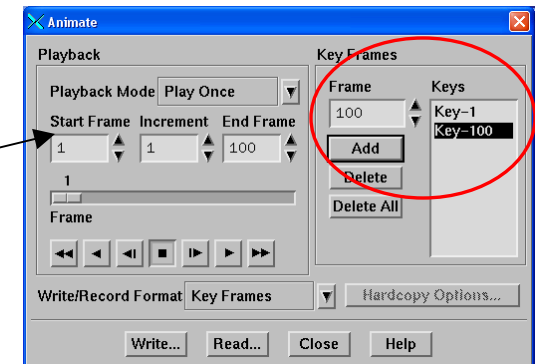


- Set this frame as the final frame

Display → Scene Animation...

Increase the number of frames to the desired value

Click on the **Add** button to enter the final frame

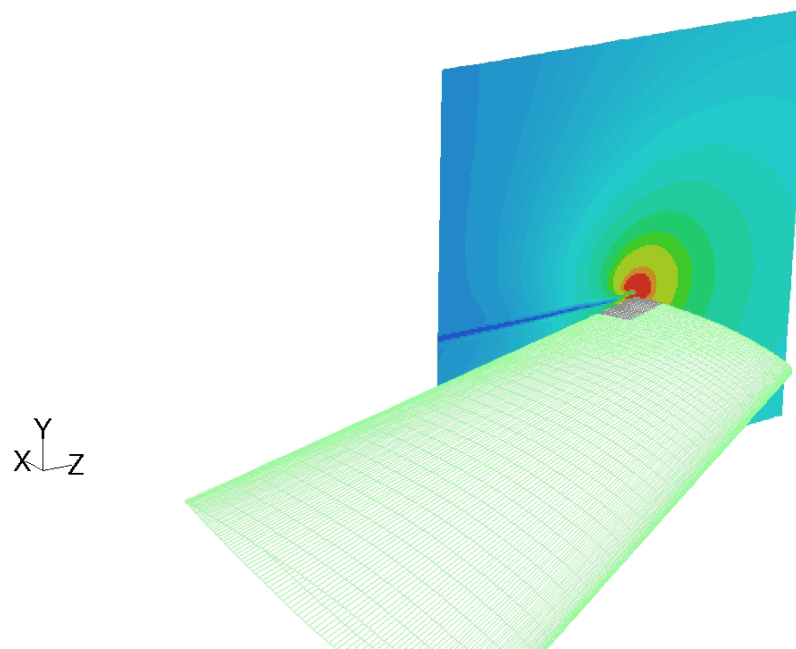


- Create the animation files

Write/Record Format

Option to create either animation files or graphic files (tif, postscript, etc)

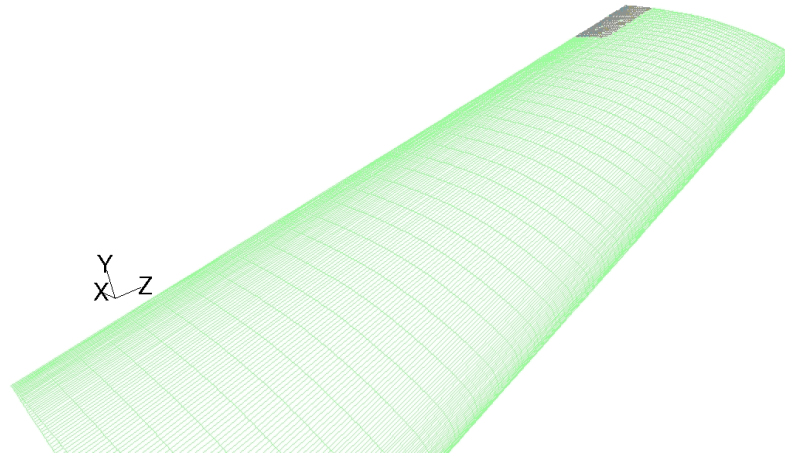
Creating Sweep Surface Animation (Solution 870)



In plane velocity at tip wake for the M6 wing

Creating Pathline Animation (Solution 90)

- **Display → Pathlines...** allows pathlines to be released and the resulting animation displayed in the graphics window
- No functionality within this panel to save the animation as hardcopy files



- It is possible to create animation files for pathlines using **Scene Animation**

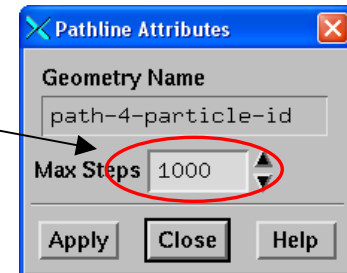
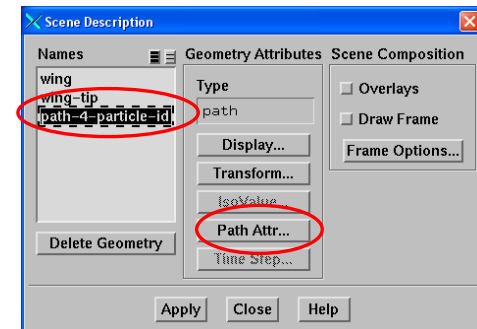
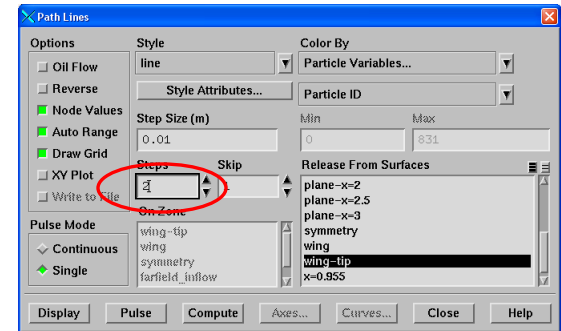
Creating Pathline Animation (Solution 90)

- Similar process as the sweep surface animation
- The first frame will have the pathlines advancing in a few steps only
Store the first frame inside the **Scene Animation** panel as before
- The final frame is set inside the **Scene Description** panel

Display → Scene...

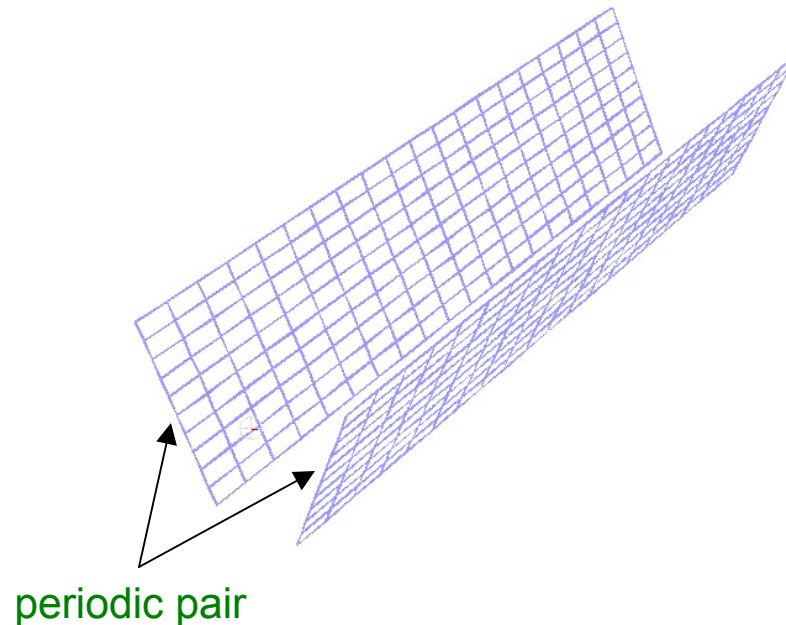
Under **Names**, select the particle scene and then click on the **Path Attr** button

- Inside the Path Attr panel, specify the maximum number of steps
- Record the final frame again inside the **Scene Animation** panel



Pathlines and Periodic Boundary

- Pathline display is compatible with periodic boundaries, either conformal or non-conformal
- As a pathline exits the domain at a face zone of a periodic pair, it will reenter the domain at the opposite zone of that periodic pair, at the same angle and velocities
- Particle tracks display also works

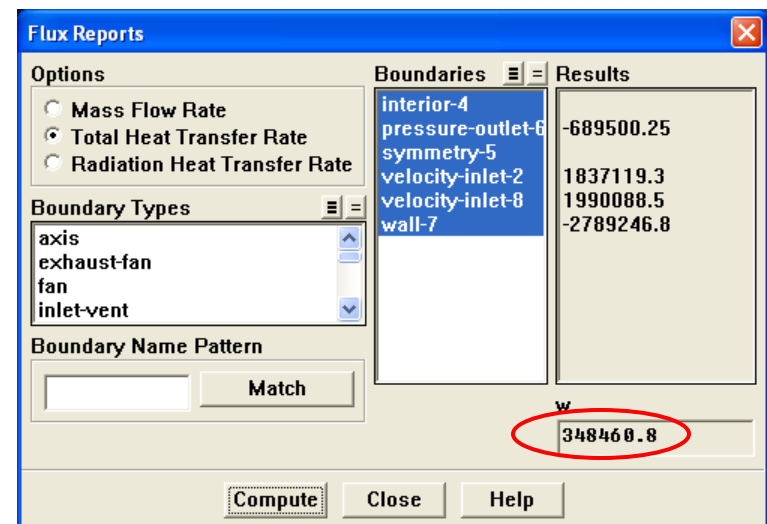
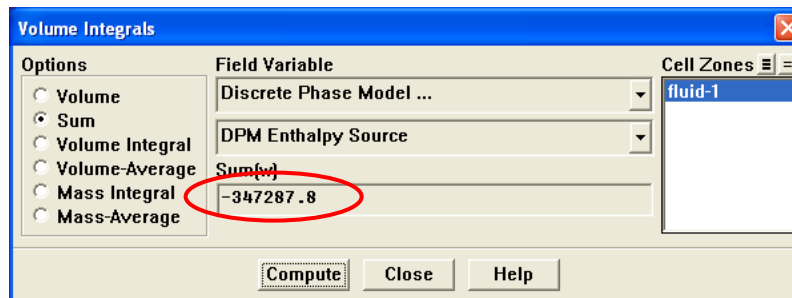




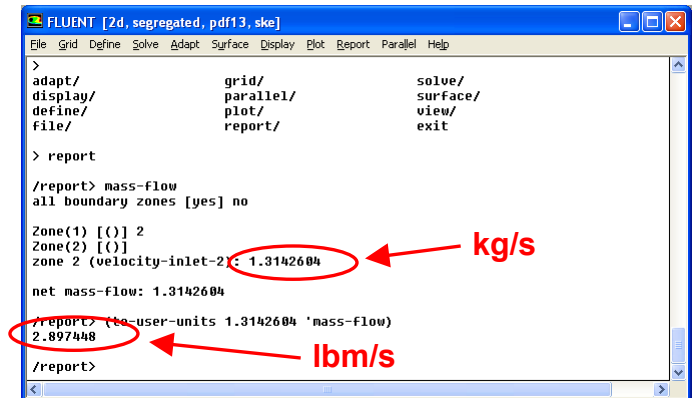
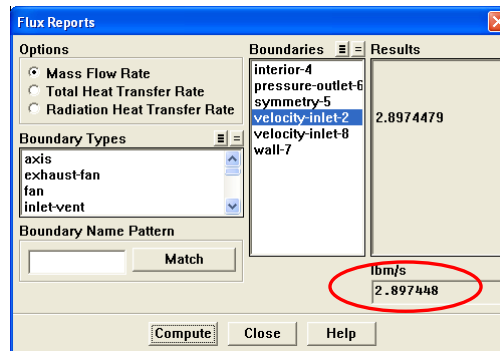
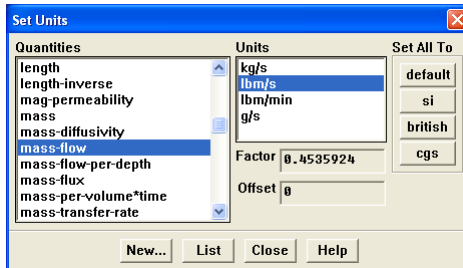
This presentation provides Tips and Tricks:

- For IO and Batch
- For Case Set-up and Mesh
- For Solving
- For Post-Processing
- For Reporting

- In DPM problems, the **Report → Fluxes...** panel will usually give a non-zero mass and energy balances
- The energy imbalance happens with cases involving **combustion or heat transfer from the particles/droplets to the gas phase**
- In these cases the user should go to:
Report → Volume Integrals → Sum → Discrete Phase Model... and calculate the sum for:
 - **DPM Mass Source**
 - **DPM Enthalpy Source**
- At convergence, the Sum of **DPM Mass Source** and **DPM Enthalpy Source** should be algebraically equal to the **Mass** and **Total Heat Transfer Rate** imbalances in the **Report → Fluxes...** panel
- For large combustion problems, this energy balance can take longer to achieve and is a better indication of convergence than low residuals



- The values of Surface Integrals and Flux Reports are always reported in **SI units** when accessed through the **Text User Interface (TUI)** while they are reported in the **user-specified units** in the **Graphic User Interface (GUI)**
- Workaround: Use a function which will convert the value from SI units to the user specified units for the desired variable



- In the example above, the mass-flow rate value is 1.3142 kg/s but the user is interested in lbm/s, which is the user specified unit in the GUI. The following function can be used:

(to-user-units 1.3142 'mass-flow)

- In order to use this function for other variables, “mass-flow” should be replaced by the desired variable (e.g. temperature, velocity, length, etc)

Convective Heat Transfer Coefficient (HTC)

Flux-based

- The HTC is defined as

$$q''_{total} = h''_{total} (T_w - T_{ref})$$

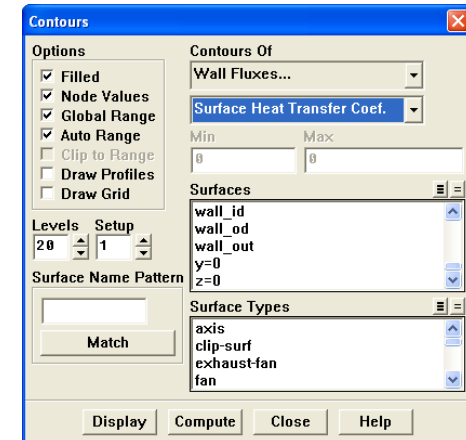
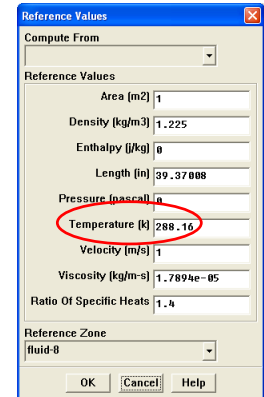
where

$$q''_{total} = q_{rad} + q_{conv}$$

- By default T_{ref} is the reference temperature from the Reference panel
- HTC is a field variable accessible through Surface Heat Transfer Coe. under Wall Fluxes
- Options for T_{ref}
 - T_{ref} is the local bulk temperature: Most “correct” approach, but calculating bulk temperature is not straightforward for complex geometry and UDF coding required
 - T_{ref} is fixed: Not correct everywhere
 - T_{ref} is adjacent cell temperature. It can be done using Custom Field Function of:

$$\frac{(Total\ Heat\ Surface\ Heat\ Flux - Radiation\ Heat\ Flux)}{(Wall\ Temperature\ (Outer\ Surface) - Static\ Temperature)}$$

and plotting without node values



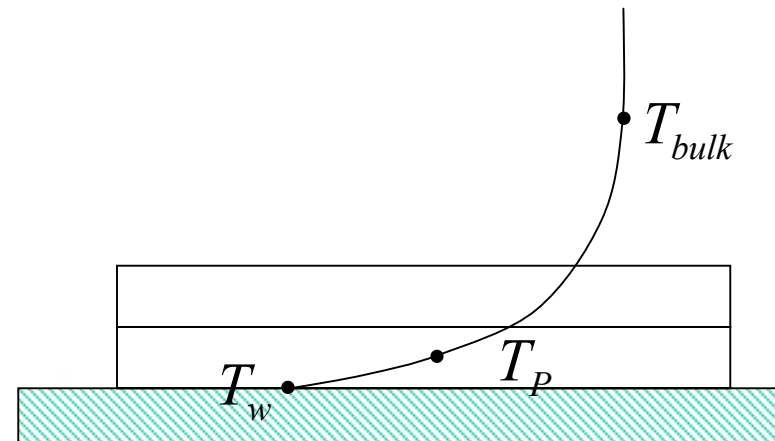
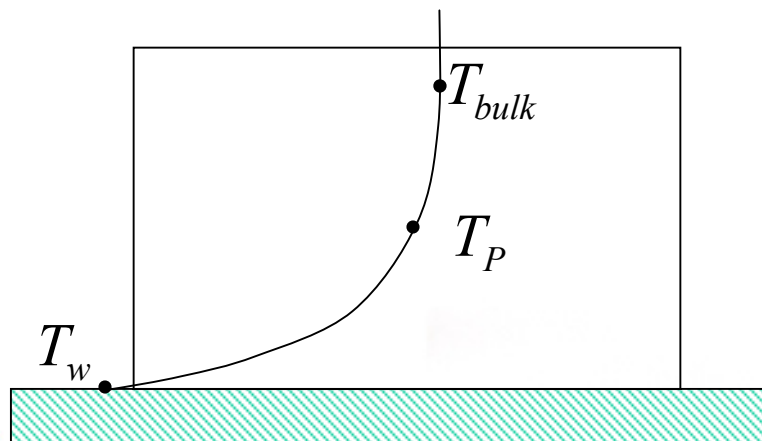
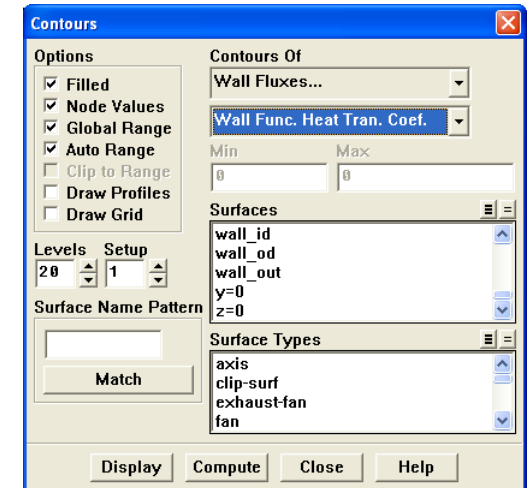
Convective Heat Transfer Coefficient (HTC)

- Based on wall functions

$$T^* \equiv \frac{(T_w - T_p) \rho c_p C_\mu^{1/4} k_p^{1/2}}{\dot{q}''} = \begin{cases} \text{Pr } y^* & (y^* < y_T^*) \\ \text{Pr}_t \left[\frac{1}{\kappa} \ln(Ey^*) + P \right] & (y^* > y_T^*) \end{cases}$$

$$h = \dot{q}'' / (T_w - T_p) = \frac{\rho c_p C_\mu^{1/4} k_p^{1/2}}{T^*}$$

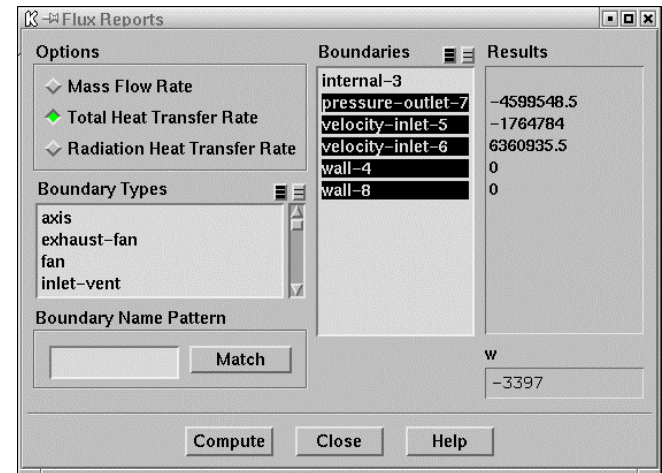
- This requires only to run turbulence equations
- The heat flux is \dot{q}'' is the convective flux only
- It would be equivalent to using the adjacent cell as reference temperature in the flux-based HTC
- Generally, for accurate h , we want T_p to be close to the bulk temperature
- This can be achieved only if y^+ is large



Reporting – Heat Flux and HTC

- Heat flux report:
 - It is recommended that you perform a heat balance check to ensure that your solution is truly converged
- Exporting Heat Flux Data:
 - It is possible to export heat flux data on wall zones (including radiation) to a generic file
 - Use the text interface:
 - `file/export/custom-heat-flux`
 - File format for each selected face zone:

```
zone-name nfaces
x_f  y_f  z_f  A  Q  T_w  T_c  HTC
...
```



Species and UDS reports

- **report/species-mass-flow**
 - Print list of species mass flow rate at inlets and outlets
 - Available after performing 1 iteration
- **report/uds-flow**
 - Print list of user-defined scalar flow rate at boundaries

```

FLUENT [axi, segregated, spe5, ske]
File Grid Define Solve Adapt Surface Display Plot Report Parallel Help

/report>
cell-volume
cell-sum
cell-vol-int
cell-average
cell-mass-int
cell-mass-avg
path-line-summary
heat-transfer
rad-heat-trans
mass-flow

species-mass-flow
uds-flow
print-histogram
reference-values/summary
surface-area
surface-avg
surface-facet-avg
surface-facet-min
surface-facet-max

surface-flow-rate
surface-int
surface-mass-rate
surface-mass-avg
surface-vertex-avg
surface-vertex-min
surface-vertex-max
wall-forces
wall-moments

/report> species-mass-flow
Zone 8 (air-inlet): (-6.8238079e-14 0.021393077 -1.6243408e-11 -1.3298383e-11)
Zone 6 (fuel-inlet): (0.0040948116 -1.4951428e-12 -4.0453518e-13 -3.3119053e-13)
Zone 2 (nozzle): (0 0 0)
Zone 7 (outer-wall): (0 0 0)
Zone 9 (pressure-outlet-9): (-3.4115728e-05 -0.0052453568 -0.011106912 -0.0090931635)

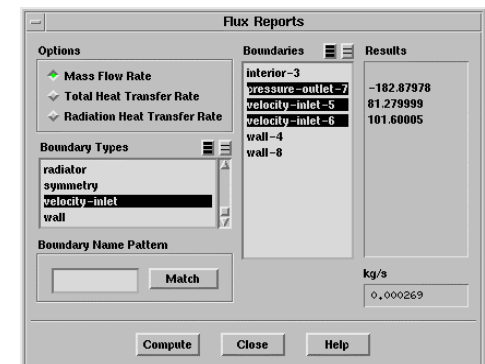
net species-mass-flow: (0.0040606959 0.016147721 -0.011106912 -0.0090931635)

/report>
  
```

- These options are **more accurate** than surface integrals at boundary zones since no interpolation is used and corresponds to:

Report → Fluxes...

in the GUI for mass and heat transfer rates



Pressure Force Calculation

- Force due to pressure is computed using the following equation in Fluent:

$$F_p = \text{Sum} [(p_g - p_{\text{ref}}) * \text{Area}]$$

where p_g is the gauge pressure and p_{ref} is the reference pressure specified in the reference value panel. The sum is over all the faces of the surface.

- For **closed surface** (closed body), it does not matter what the value of p_{ref} is since the total pressure force due to p_{ref} is zero:

$$\oint p_{\text{ref}} \vec{n} \cdot d\vec{A} = 0$$

- For **open surface**, p_{ref} must be set to the negative of the operating pressure so that the equation becomes:

$$\begin{aligned} F_p &= \text{Sum} [(p_g + p_{\text{op}}) * \text{Area}] \\ &= \text{Sum} [(p_{\text{abs}} * \text{Area})] \end{aligned}$$

Turbulent Intensity Definition

- The definition used by FLUENT when computing the turbulence intensity is:

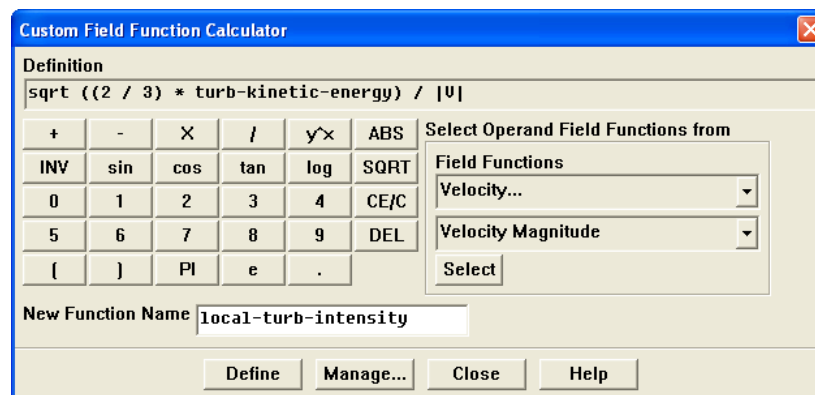
$$TI = \text{sqrt}((2/3)*k)/V_{\text{ref}}$$

where **Vref** is specified inside the **reference value panel**

- It is usually more instructive to use the local velocity magnitude for turbulence intensity calculation:

$$TI = \text{sqrt}((2/3)*k)/V_{\text{mag-local}}$$

- Can use a **custom field function** to define modified turbulent intensity



CPU Time for Serial/Parallel Solver

- For **parallel** run, one can use the **built-in parallel timer** via the GUI:

Parallel → Timer → Reset

Before iterating

Parallel → Timer → Usage

After completing iterations

- For **serial** run, you can use the following TUI command that is executed before and after the iteration period of interest:

(solver-cpu-time)

- The difference is the elapsed CPU time in seconds for the iteration period completed
- If used in the parallel session, the difference represents the sum of the CPU times of all the compute nodes

- An alternative is to use:

(benchmark `(iterate Niter))

- Niter** is the desired number of iteration
- Gives the solver time, cortex time, and elapsed time
- Can be used in serial or parallel session
- If used in parallel session, the solver time is the sum from all compute nodes

```

Parallel FLUENT@sup13.fluent.com [3d, segregated, rke]
File  Grid  Define  Solve  Adapt  Surface  Display  Plot  Report  Parallel  Help
288 7.5247e-05 5.8046e-09 5.8668e-09 4.6605e-08 4.3429e-08 9.4674
289 7.4982e-05 5.6254e-09 5.7932e-09 4.2515e-08 4.3471e-08 9.5126
290 7.5175e-05 5.7177e-09 5.7987e-09 4.5732e-08 4.2951e-08 9.4835
291 7.5542e-05 5.7493e-09 5.8709e-09 4.7452e-08 4.3397e-08 9.5114
292 7.5827e-05 5.8310e-09 5.9311e-09 4.3281e-08 4.3497e-08 9.4384
293 7.4984e-05 5.6931e-09 5.8743e-09 4.3260e-08 4.3087e-08 9.4576
294 7.5385e-05 5.7477e-09 5.8181e-09 4.7523e-08 4.3438e-08 9.4397
295 7.5815e-05 5.8361e-09 5.8260e-09 4.7568e-08 4.3877e-08 9.4462
296 7.6071e-05 5.8647e-09 5.8690e-09 4.4068e-08 4.3240e-08 9.4649
297 7.5539e-05 5.7101e-09 5.8255e-09 4.4684e-08 4.3078e-08 9.4744
298 7.5336e-05 5.7623e-09 5.8647e-09 4.7132e-08 4.3427e-08 9.4346
iter continuity x-velocity y-velocity z-velocity energy
299 7.5459e-05 5.6866e-09 5.8528e-09 4.2782e-08 4.3949e-08 9.4105
300 7.5379e-05 5.7014e-09 5.8199e-09 4.2890e-08 4.3297e-08 9.4390
cpu-time: cortex=0.5, solver=52.07
elapsed-time: 28.782145
()
  
```




Thank you !

Thinking CFD ... Think Fluent !
Care to use the very best in CFD

