



Solver Settings

Introductory FLUENT Training



Outline

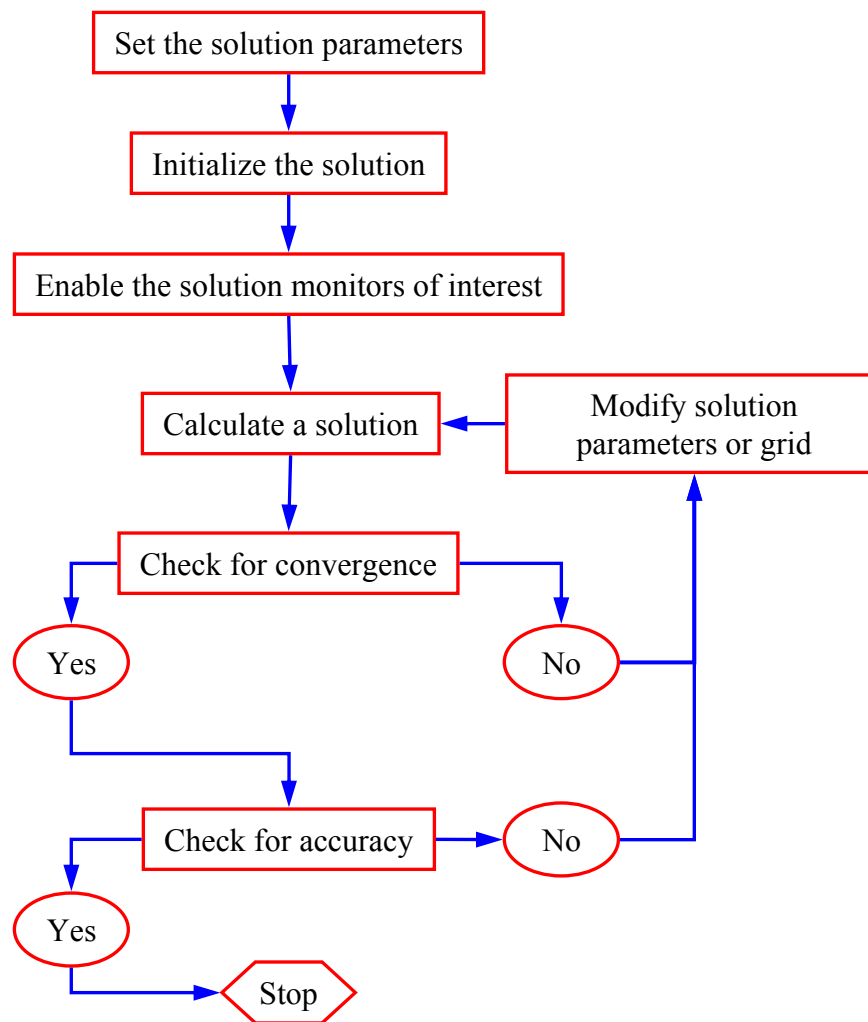
- ◆ Using the Solver
 - Setting Solver Parameters
 - Convergence
 - Definition
 - Monitoring
 - Stability
 - Accelerating Convergence
 - Accuracy
 - Grid Independence
 - Grid Adaption
 - Unsteady Flows Modeling
 - Unsteady-flow problem setup
 - Unsteady flow modeling options
 - Summary
 - Appendix

Outline

- ◆ Using the Solver (solution procedure overview)
 - Setting Solver Parameters
 - Convergence
 - Definition
 - Monitoring
 - Stability
 - Accelerating Convergence
 - Accuracy
 - Grid Independence
 - Grid Adaption
 - Unsteady Flows Modeling
 - Unsteady-flow problem setup
 - Unsteady flow modeling options
 - Summary
 - Appendix

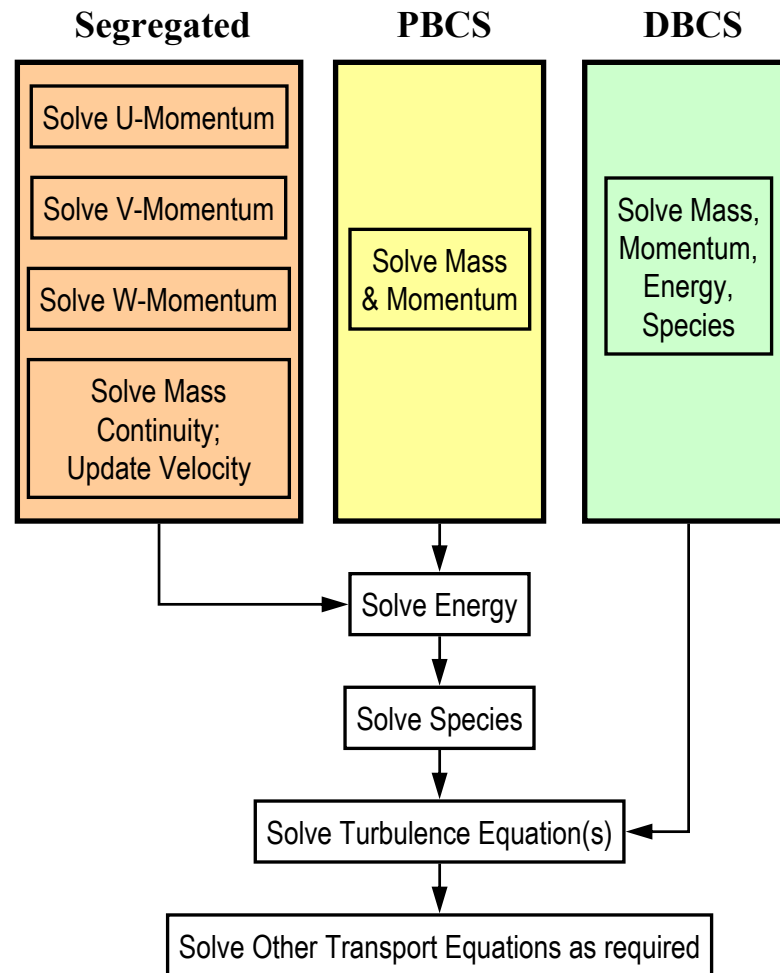
Solution Procedure Overview

- ◆ Solution parameters
 - Choosing the solver
 - Discretization schemes
- ◆ Initialization
- ◆ Convergence
 - Monitoring convergence
 - Stability
 - Setting Under-relaxation
 - Setting Courant number
 - Accelerating convergence
- ◆ Accuracy
 - Grid Independence
 - Adaption



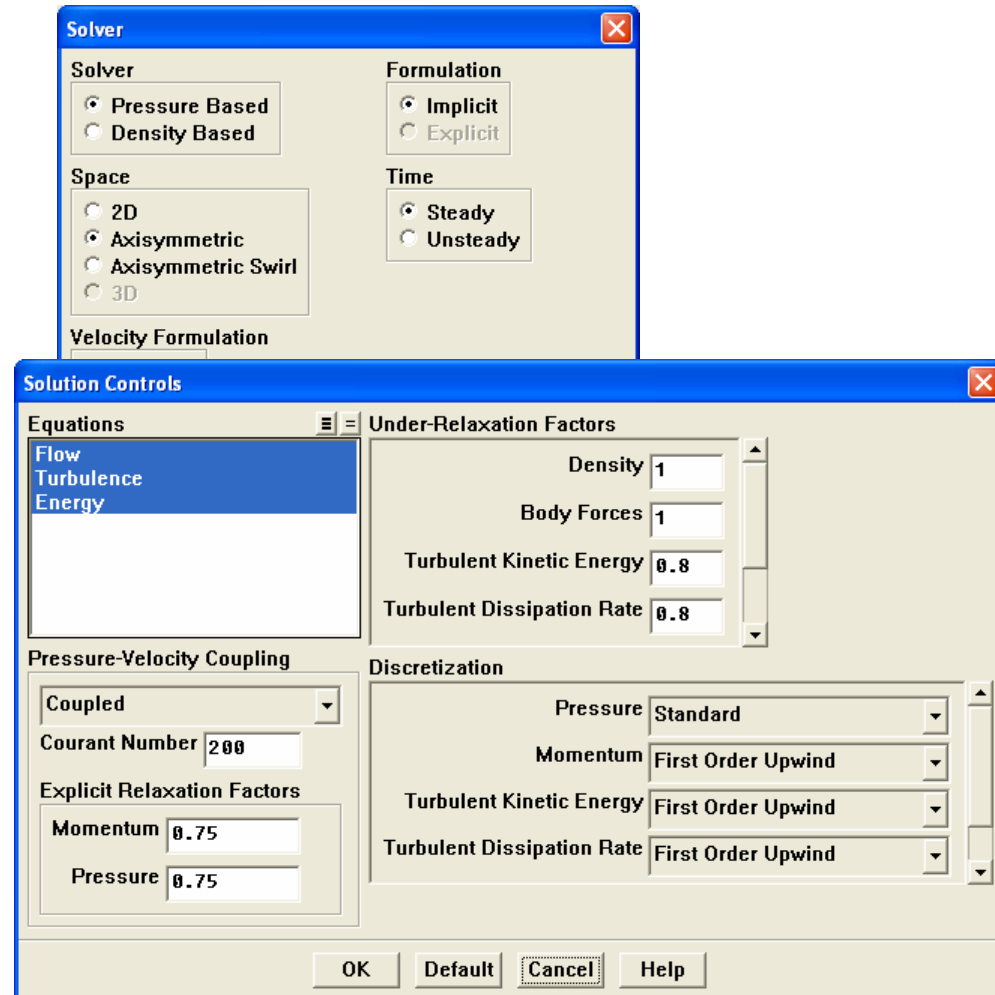
Available Solvers

- ◆ There are two kinds of solvers available in FLUENT.
 - Pressure-based solver
 - Density-based coupled solver (DBCS)
- ◆ The **pressure-based** solvers take momentum and pressure (or pressure correction) as the primary variables.
- ◆ Pressure-velocity coupling algorithms are derived by reformatting the continuity equation
- ◆ Two algorithms are available with the pressure-based solvers:
 - Segregated solver – Solves for pressure correction and momentum sequentially.
 - Coupled Solver (PBCS) – Solves pressure and momentum simultaneously.



Available Solvers

- ◆ **Density-Based Coupled Solver** – equations for continuity, momentum, energy, and species, if required, are solved in vector form. Pressure is obtained through the equation of state. Additional scalar equations are solved in a segregated fashion.
 - ◆ The density-based solver can use either an implicit or explicit solution approach:
 - Implicit – Uses a point-implicit Gauss-Seidel / symmetric block Gauss-Seidel / ILU method to solve for variables.
 - Explicit: uses a multi-step Runge-Kutta explicit time integration method
- Note:** the pressure-based solvers are implicit



Choosing a Solver

- ◆ The **pressure-based** solver is applicable for a wide range of flow regimes from low speed incompressible flow to high-speed compressible flow.
 - Requires less memory (storage).
 - Allows flexibility in the solution procedure.
- ◆ The **pressure-based** coupled solver (PBCS) is applicable for most single phase flows, and yields superior performance to the **pressure-based** (segregated) solver.
 - Not available for multiphase (Eulerian), periodic mass-flow and NITA cases.
 - Requires 1.5–2 times more memory than the segregated solver.
- ◆ The **density-based** coupled solver (DBCS) is applicable when there is a strong coupling, or interdependence, between density, energy, momentum, and/or species.
 - Examples: High speed compressible flow with combustion, hypersonic flows, shock interactions.
- ◆ The Implicit solution approach is generally preferred to the explicit approach, which has a very strict limit on time step size
- ◆ The explicit approach is used for cases where the characteristic time scale of the flow is on the same order as the acoustic time scale. (e.g.: propagation of high-Ma shock waves).

Discretization (Interpolation Methods)

- ◆ Field variables (stored at cell centers) must be interpolated to the faces of the control volumes.

$$\frac{\partial(\rho\phi)}{\partial t} V + \sum_f^{N_{\text{faces}}} \rho_f \mathbf{V}_f \phi_f \cdot \mathbf{A}_f = \sum_f^{N_{\text{faces}}} \Gamma_\phi \nabla \phi_f \cdot \mathbf{A}_f + S_\phi V$$

- ◆ Interpolation schemes for the convection term:
 - **First-Order Upwind** – Easiest to converge, only first-order accurate.
 - **Power Law** – More accurate than first-order for flows when $\text{Re}_{\text{cell}} < 5$ (typ. low Re flows)
 - **Second-Order Upwind** – Uses larger stencils for 2nd order accuracy, essential with tri/tet mesh or when flow is not aligned with grid; convergence may be slower.
 - **Monotone Upstream-Centered Schemes for Conservation Laws (MUSCL)** – Locally 3rd order convection discretization scheme for unstructured meshes; more accurate in predicting secondary flows, vortices, forces, etc.
 - **Quadratic Upwind Interpolation (QUICK)** – Applies to quad/hex and hybrid meshes, useful for rotating/swirling flows, 3rd-order accurate on uniform mesh

Interpolation Methods (Gradients)

- ◆ Gradients of solution variables are required in order to evaluate diffusive fluxes, velocity derivatives, and for higher-order discretization schemes.

$$\frac{\partial(\rho\phi)}{\partial t} V + \sum_f^{N_{\text{faces}}} \rho_f \mathbf{V}_f \phi_f \cdot \mathbf{A}_f = \boxed{\sum_f^{N_{\text{faces}}} \Gamma_\phi \nabla \phi_f \cdot \mathbf{A}_f} + S_\phi V$$

- ◆ The gradients of solution variables at cell centers can be determined using three approaches:
 - **Green-Gauss Cell-Based** – The default method; solution may have false diffusion (smearing of the solution fields).
 - **Green-Gauss Node-Based** – More accurate; minimizes false diffusion; recommended for tri/tet meshes.
 - **Least-Squares Cell-Based** – Recommended for polyhedral meshes; has the same accuracy and properties as Node-based Gradients.
- ◆ Gradients of solution variables at faces computed using multi-dimensional Taylor series expansion

Interpolation Methods for Face Pressure

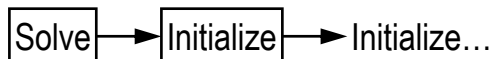
- ◆ Interpolation schemes for calculating cell-face pressures when using the segregated solver in FLUENT are available as follows:
 - **Standard** – The default scheme; reduced accuracy for flows exhibiting large surface-normal pressure gradients near boundaries (but should not be used when steep pressure changes are present in the flow – PRESTO! scheme should be used instead.)
 - **PRESTO!** – Use for highly swirling flows, flows involving steep pressure gradients (porous media, fan model, etc.), or in strongly curved domains
 - **Linear** – Use when other options result in convergence difficulties or unphysical behavior
 - **Second-Order** – Use for compressible flows; not to be used with porous media, jump, fans, etc. or VOF/Mixture multiphase models
 - **Body Force Weighted** – Use when body forces are large, e.g., high Ra natural convection or highly swirling flows

Pressure-Velocity Coupling

- ◆ Pressure-velocity coupling refers to the numerical algorithm which uses a combination of continuity and momentum equations to derive an equation for pressure (or pressure correction) when using the pressure-based solver.
- ◆ Four algorithms are available in FLUENT.
 - Semi-Implicit Method for Pressure-Linked Equations (SIMPLE)
 - The default scheme, robust
 - SIMPLE-Consistent (SIMPLEC)
 - Allows faster convergence for simple problems (e.g., laminar flows with no physical models employed).
 - Pressure-Implicit with Splitting of Operators (PISO)
 - Useful for unsteady flow problems or for meshes containing cells with higher than average skewness
 - Fractional Step Method (FSM) for unsteady flows.
 - Used with the NITA scheme; similar characteristics as PISO.

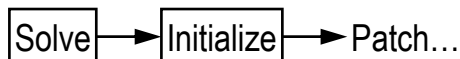
Initialization

- ◆ Iterative procedure requires that all solution variables be initialized before calculating a solution

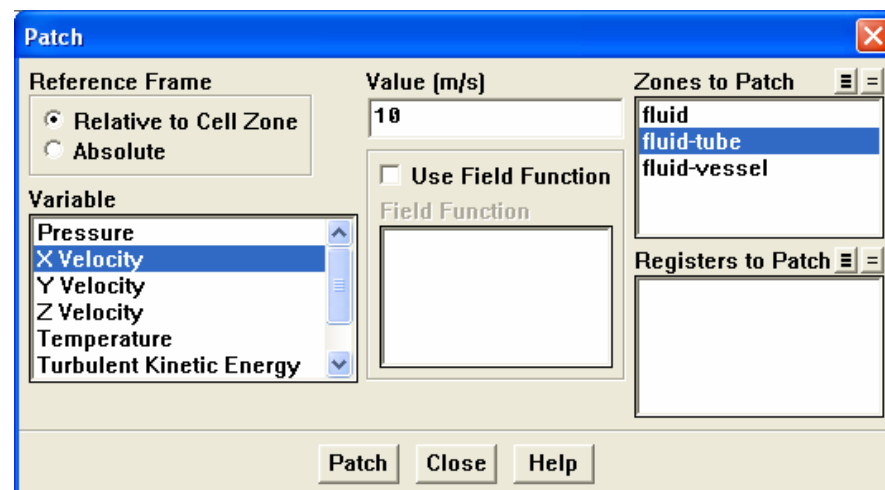
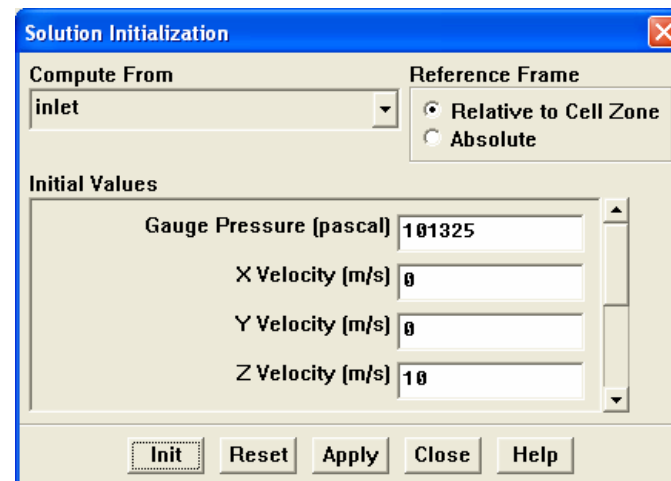


- Realistic guesses improves solution stability and accelerates convergence
- In some cases, a good initial guess is required.
 - Example: high temperature region to initiate chemical reaction.

- ◆ Patch values for individual variables in certain regions.



- Free jet flows (high velocity for jet)
- Combustion problems (high temperature region to initialize reaction)
- Cell registers (created by marking the cells in the Adaption panel) can be used for patching values into various regions of the domain.



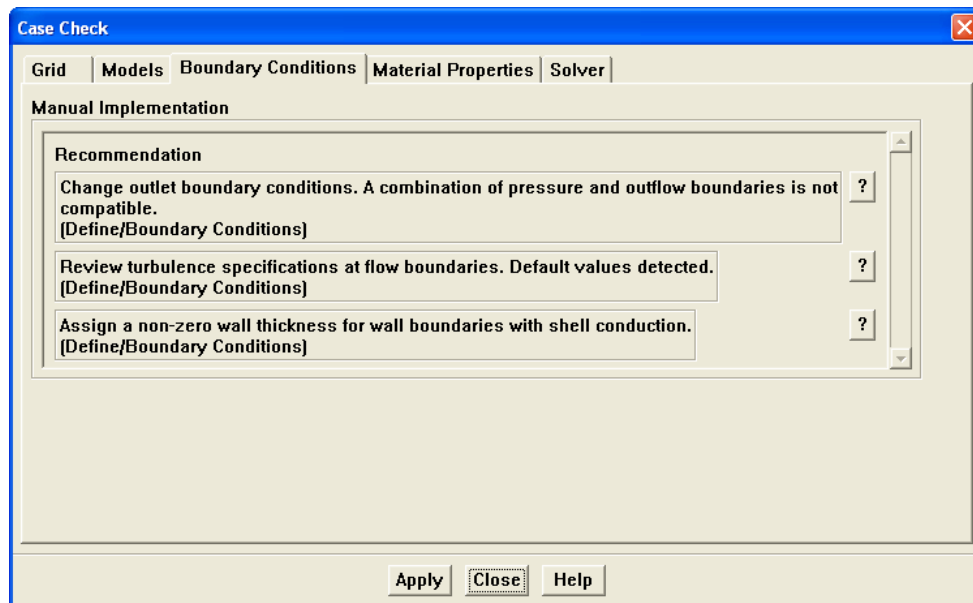
FMG Initialization

- ◆ Full Multigrid (FMG) Initialization can be used to create a better initialization of the flow field:
 - TUI command: `/solve/init/fmg-initialization`
- ◆ FMG is computationally inexpensive and faster. Euler equations are solved with first-order accuracy on the coarse-level meshes.
- ◆ It can be used with both pressure and density based solvers, but only in steady state.
- ◆ FMG uses the Full Approximation Storage (FAS) multigrid method to solve the flow problem on a sequence of coarser meshes, before transferring the solution onto the actual mesh.
 - Settings can be accessed by the TUI command:
`/solve/init/set-fmg-initialization`
- ◆ FMG Initialization is useful for complex flow problems involving large gradients in pressure and velocity on large domains (e.g.: rotating machinery, expanding spiral ducts)

Case Check

- ◆ Case Check is a utility in FLUENT which looks for common setup errors and provides guidance in selecting case parameters and models.
 - Uses rules and best practices
- ◆ Case check will look for compliance in:
 - Grid
 - Model Selection
 - Boundary Conditions
 - Material Properties
 - Solver Settings
- ◆ Tabbed sections contain recommendations
- ◆ Automatic recommendations: the utility will make the changes
- ◆ Manual recommendations: the user has to make the changes

Solve → Case Check...



Outline

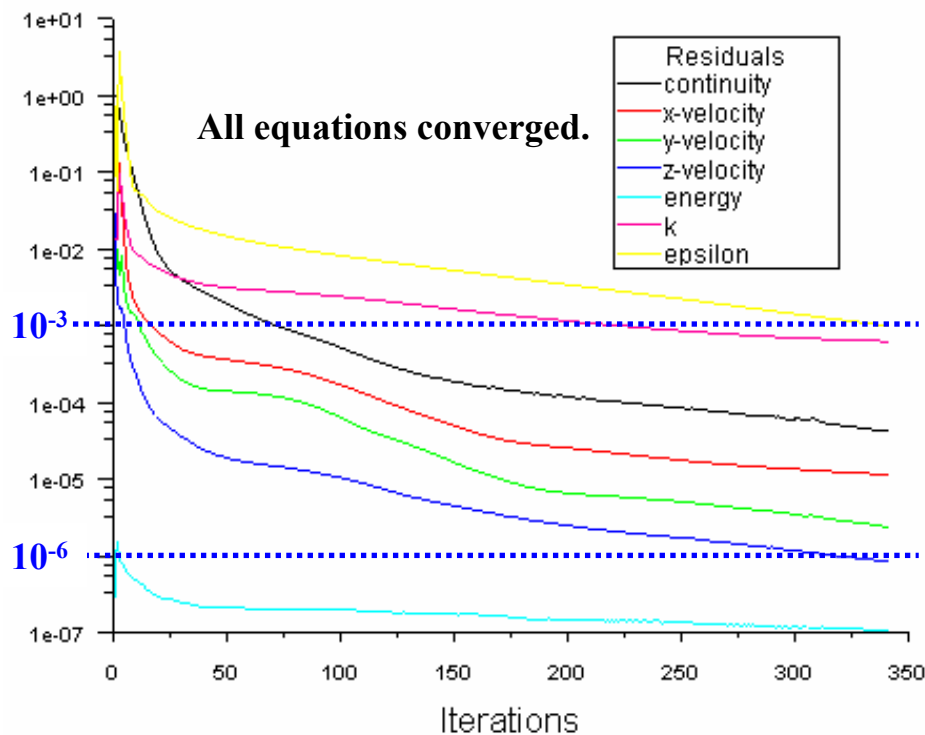
- ◆ Using the Solver
 - Setting Solver Parameters
 - **Convergence**
 - **Definition**
 - **Monitoring**
 - **Stability**
 - **Accelerating Convergence**
 - Accuracy
 - Grid Independence
 - Grid Adaption
 - Unsteady Flows Modeling
 - Unsteady-flow problem setup
 - Unsteady flow modeling options
 - Summary
 - Appendix

Convergence

- ◆ At convergence, the following should be satisfied:
 - All discrete conservation equations (momentum, energy, etc.) are obeyed in all cells **to a specified tolerance** OR the solution no longer changes with subsequent iterations.
 - Overall mass, momentum, energy, and scalar balances are achieved.
- ◆ Monitoring convergence using residual history:
 - Generally, a decrease in residuals by **three orders of magnitude** indicates at least qualitative convergence. At this point, the major flow features should be established.
 - Scaled energy residual must decrease to 10^{-6} (for the pressure-based solver).
 - Scaled species residual may need to decrease to 10^{-5} to achieve species balance.
- ◆ Monitoring quantitative convergence:
 - Monitor other relevant key variables/physical quantities for a confirmation.
 - Ensure that overall mass/heat/species conservation is satisfied.

Convergence Monitors: Residuals

- ◆ Residual plots show when the residual values have reached the specified tolerance.



Residual Monitors

Options: ☒ Print ☒ Plot

Storage: Iterations **1000**

Normalization: ☐ Normalize ☒ Scale

Convergence Criterion: absolute

Plotting: Window **0** Iterations **1000** Axes... Curves...

Residual	Check	Absolute
continuity	☒	0.001
x-velocity	☒	0.001
y-velocity	☒	0.001
z-velocity	☒	0.001
energy	☒	1e-06

OK Plot Renorm Cancel Help

Convergence Monitors: Forces/Surfaces

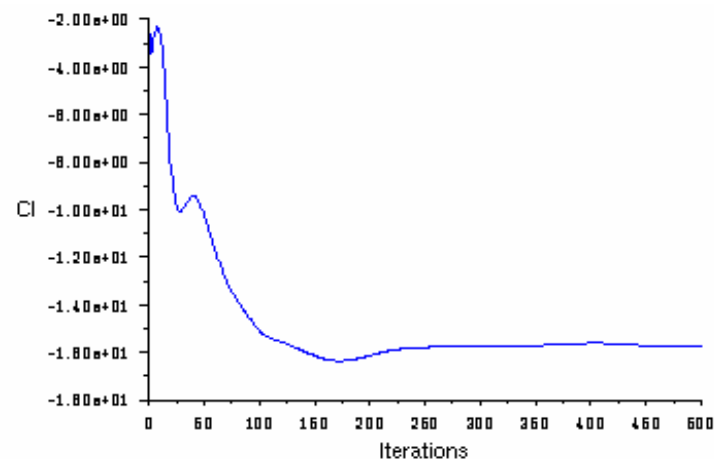
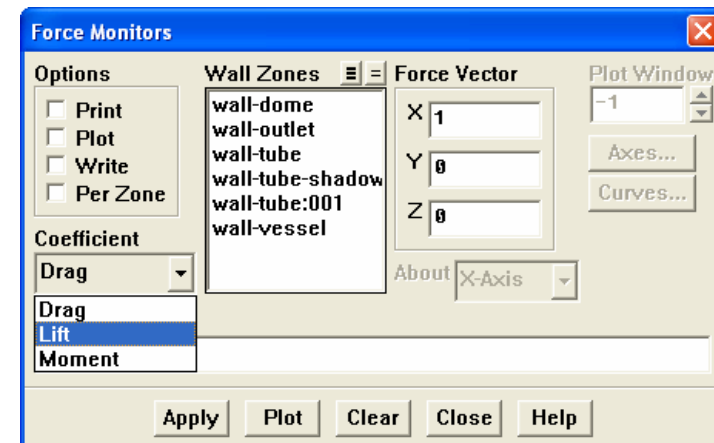
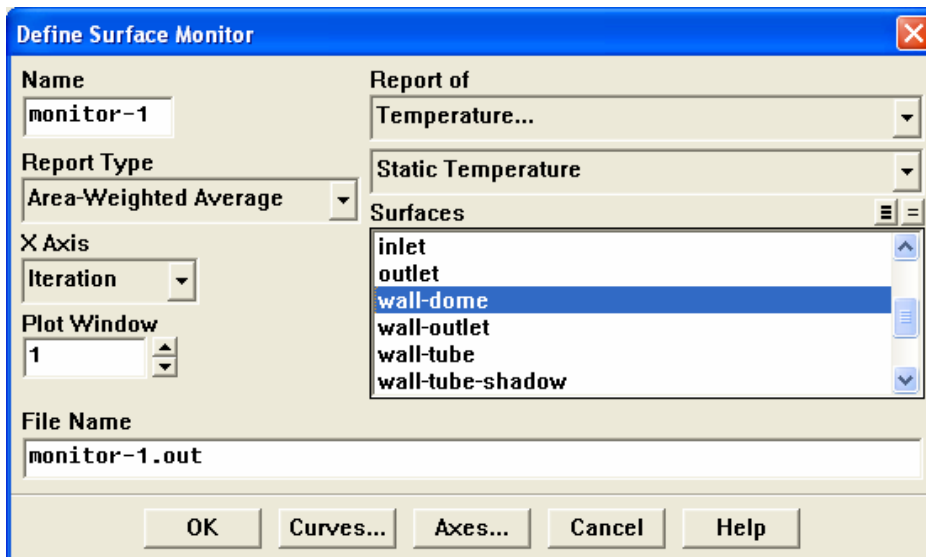
- ◆ In addition to residuals, you can also monitor:

- Lift, drag, or moment

Solve → Monitors → Force...

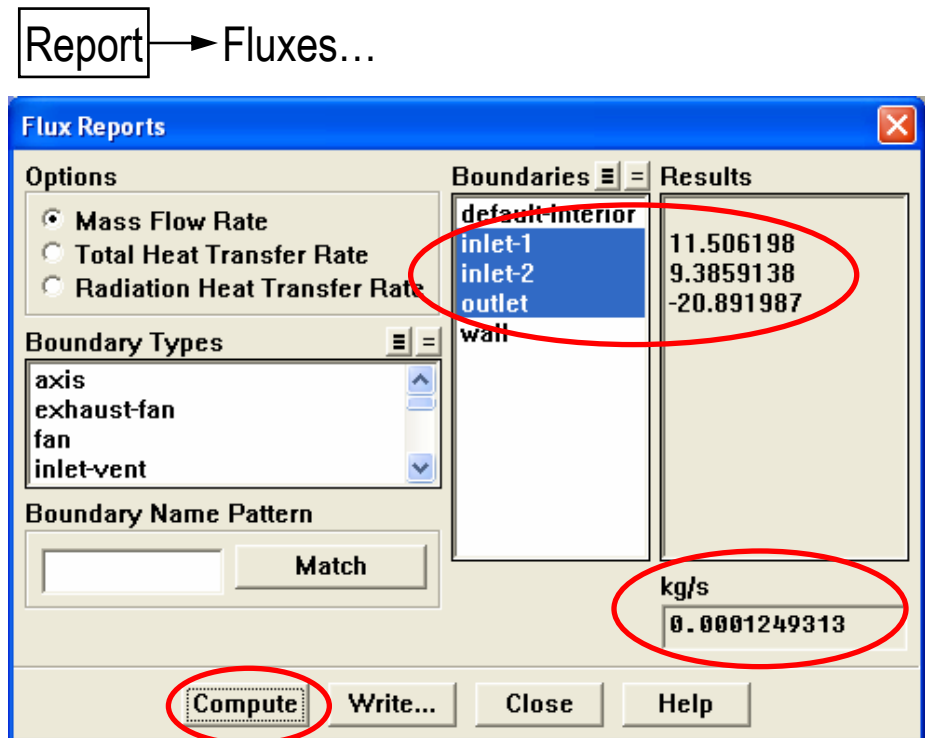
- Pertinent variables or functions (e.g., surface integrals) at a boundary or any defined surface.

Solve → Monitors → Surface...



Checking for Flux Conservation

- ◆ In addition to monitoring residual and variable histories, you should also check for overall heat and mass balances.
- ◆ The net imbalance should be less than 1% of the smallest flux through the domain boundary

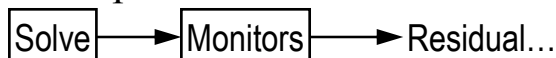


Tightening the Convergence Tolerance

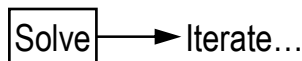
- ◆ If solution monitors indicate that the solution is converged, but the solution is still changing or has a large mass/heat imbalance, this clearly indicates the solution is not yet converged.

- ◆ In this case, you need to:

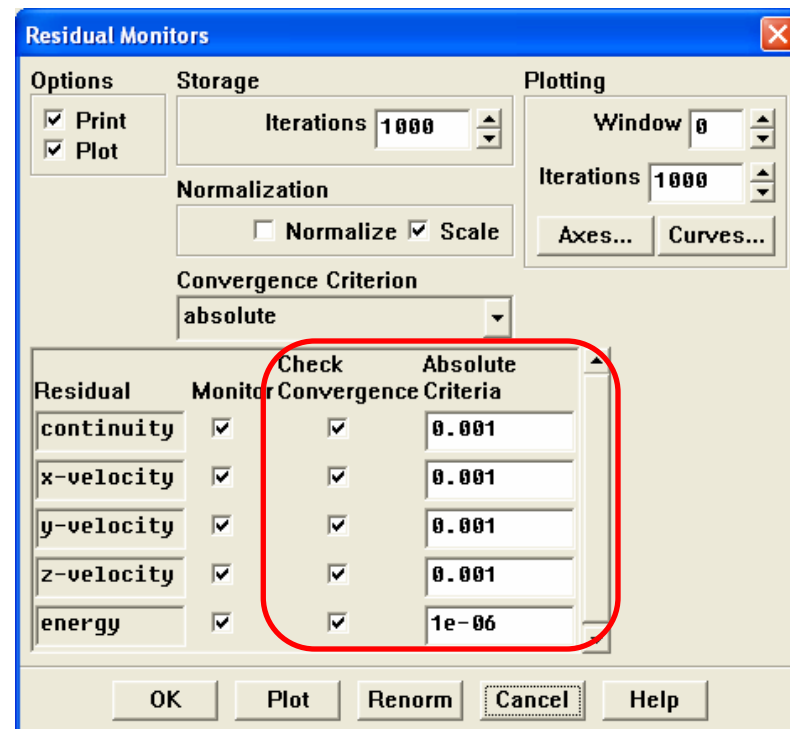
- Reduce values of Convergence Criterion or disable Check Convergence in the Residual Monitors panel.



- Continue iterations until the solution converges.

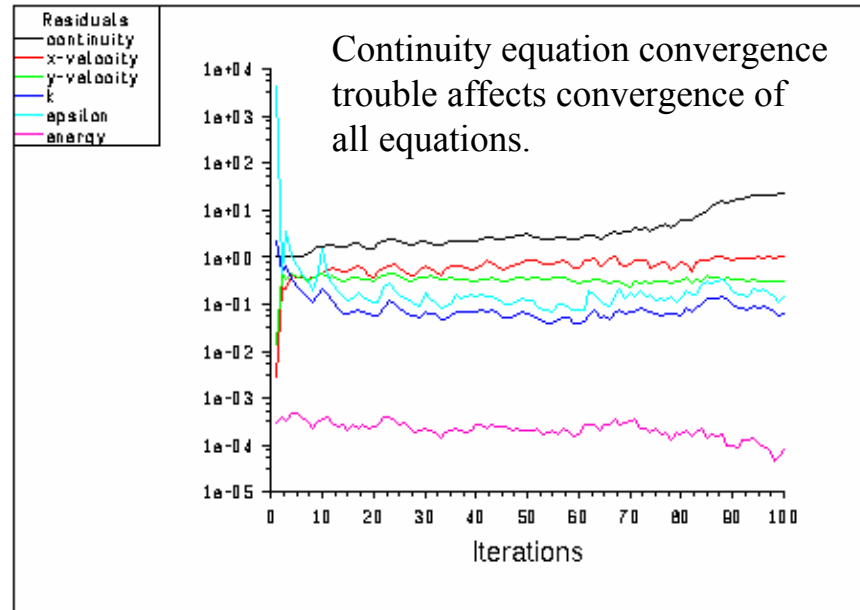


- ◆ Selecting none under Convergence Criterion will instruct FLUENT to not check convergence for any equations.



Convergence Difficulties

- ◆ Numerical instabilities can arise with an ill-posed problem, poor quality mesh, and/or inappropriate solver settings.
 - Exhibited as increasing (diverging) or “stuck” residuals.
 - Diverging residuals imply increasing imbalance in conservation equations.
 - Unconverged results are very misleading!
- ◆ Troubleshooting
 - Ensure that the problem is well-posed.
 - Compute an initial solution using a first-order discretization scheme.
 - Decrease under-relaxation factors for equations having convergence problems (pressure-based solver).
 - Decrease the Courant number (density-based solver)
 - Remesh or refine cells which have large aspect ratio or large skewness.



Modifying Under-Relaxation Factors

- ◆ Under-relaxation factor, α , is included to stabilize the iterative process for the pressure-based solver
- ◆ Use default under-relaxation factors to start a calculation.

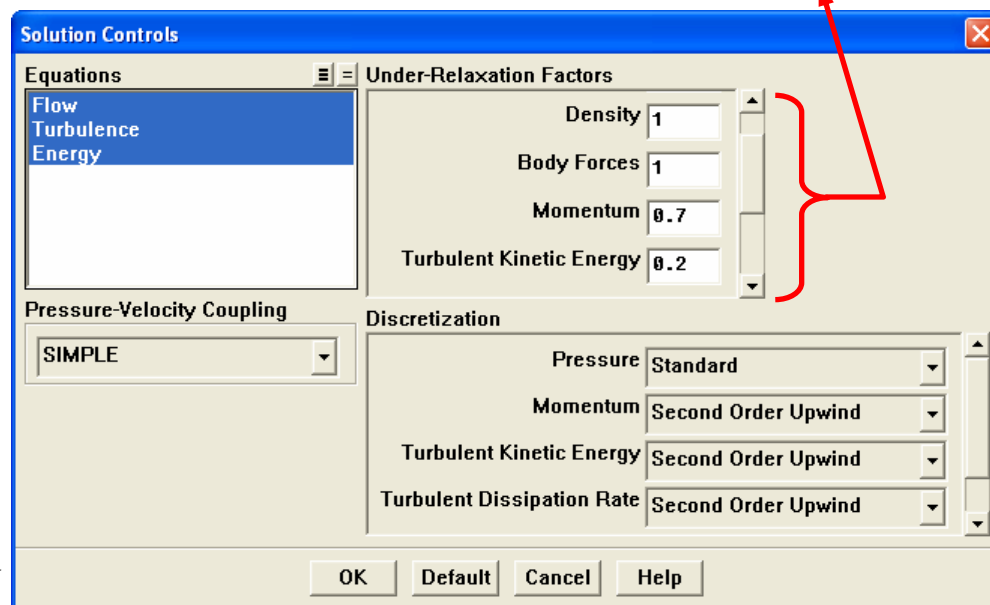
Solve

 →

Controls

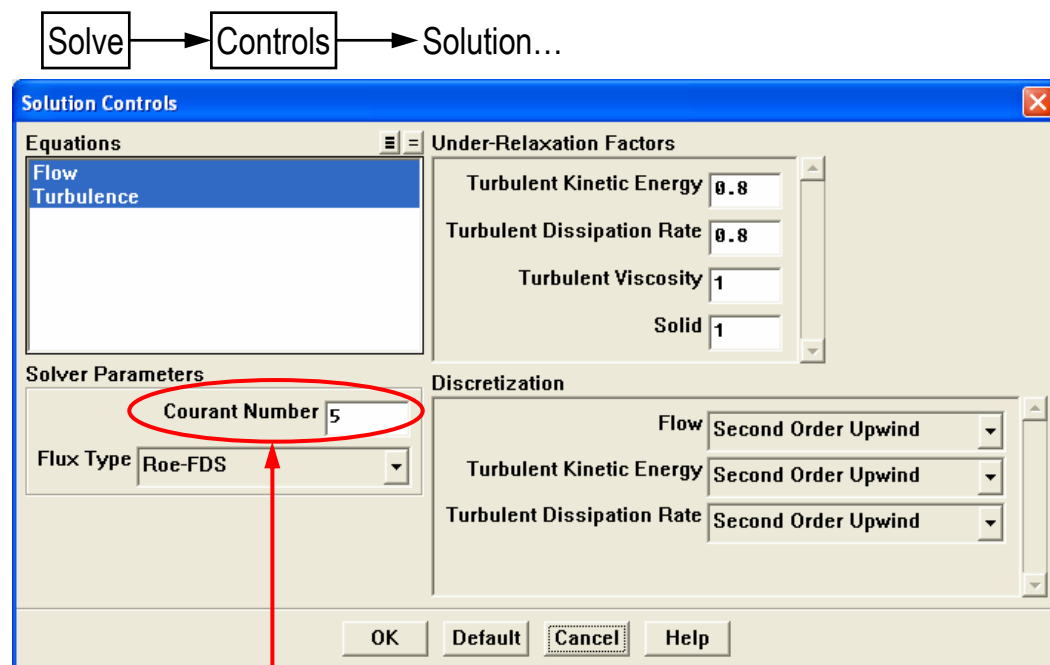
 → Solution...
- ◆ Decreasing under-relaxation for momentum often aids convergence.
 - Default settings are suitable for a wide range of problems, you can reduce the values when necessary
 - Appropriate settings are best learned from experience!
- ◆ For density-based solvers, under-relaxation factors for equations *outside* coupled set are modified as in the pressure-based solver.

$$\phi_p = \phi_{p,old} + \alpha \Delta\phi_p$$



Modifying the Courant Number

- ◆ A transient term is included in the density-based solver even for steady state problems.
 - The Courant number defines the time step size.
- ◆ For density-based explicit solver:
 - Stability constraints impose a maximum limit on the Courant number.
 - Cannot be greater than 2 (default value is 1).
 - Reduce the Courant number when having difficulty converging.
- ◆ For density-based implicit solver:
 - The Courant number is not limited by stability constraints.
 - Default value is 5.



$$\Delta t = \frac{(\text{CFL}) \Delta x}{u}$$

← Mesh size

← Appropriate velocity scale

Accelerating Convergence

- ◆ Convergence can be accelerated by:
 - Supplying better initial conditions
 - Starting from a previous solution (using file/interpolation when necessary)
 - Gradually increasing under-relaxation factors or Courant number
 - Excessively high values can lead to instabilities or convergence problems
 - Recommend saving case and data files before continuing iterations
 - Controlling multigrid solver settings (but default settings provide a robust Multigrid setup and typically do not need to be changed). See the Appendix for details on the Multigrid settings.

Starting from a Previous Solution

- ◆ Previous solution can be used as an initial condition when changes are made to problem definition.
 - Use solution interpolation to initialize a run (especially useful for starting fine-mesh cases when coarse-mesh solutions are available). File → Interpolate...
 - Once the solution is initialized, additional iterations always use the current data set as the starting point.
 - Some suggestions on how to provide initial conditions for some actual problems:

Actual Problem	Initial Condition
Heat Transfer	Isothermal
Natural convection	Low Rayleigh number
Combustion / reacting flow	Cold flow
Turbulence	Inviscid (Euler) solution

Outline

- ◆ Setting Solver Parameters
- ◆ Convergence
 - Definition
 - Monitoring
 - Stability
 - Accelerating Convergence
- ◆ Accuracy
 - Grid Independence
 - Grid Adaption
- ◆ Unsteady Flows Modeling
 - Unsteady-flow problem setup
 - Unsteady flow modeling options
- ◆ Summary
- ◆ Appendix

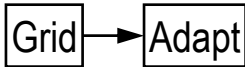
Solution Accuracy

- ◆ A converged solution is not necessarily a correct one!
 - Always inspect and evaluate the solution by using available data, physical principles and so on.
 - Use the second-order upwind discretization scheme for final results.
 - Ensure that solution is grid-independent:
 - Use adaption to modify the grid or create additional meshes for the grid-independence study
- ◆ If flow features do not seem reasonable:
 - Reconsider physical models and boundary conditions
 - Examine mesh quality and possibly remesh the problem
 - Reconsider the choice of the boundaries' location (or the domain): inadequate choice of domain (especially the outlet boundary) can significantly impact solution accuracy

Mesh Quality and Solution Accuracy

- ◆ Numerical errors are associated with calculation of cell gradients and cell face interpolations.
- ◆ Ways to contain the numerical errors:
 - Use higher-order discretization schemes (second-order upwind, MUSCL)
 - Attempt to align grid with the flow to minimize the “false diffusion”
 - Refine the mesh
 - Sufficient mesh density is necessary to resolve salient features of flow
 - ◆ Interpolation errors decrease with decreasing cell size
 - Minimize variations in cell size in non-uniform meshes
 - ◆ Truncation error is minimized in a uniform mesh
 - ◆ FLUENT provides capability to adapt mesh based on cell size variation
 - Minimize cell skewness and aspect ratio
 - ◆ In general, avoid aspect ratios higher than 5:1 (but higher ratios are allowed in boundary layers)
 - ◆ Optimal quad/hex cells have bounded angles of 90 degrees
 - ◆ Optimal tri/tet cells are equilateral

Determining Grid Independence

- ◆ A “grid-independent” solution exists when the solution no longer changes with further grid refinement.
- ◆ Systematic procedure for obtaining a grid-independent solution:
 - Generate a new, finer mesh
 - Use the solution-based adaption feature in FLUENT. 
 - ◆ Save the original mesh before doing this.
 - ◆ If you know where large gradients should occur, you need to have a fine mesh in the original mesh for that region, e.g. use boundary layers and/or size functions.
 - ◆ Adapt the mesh
 - Data from the original mesh is interpolated onto the finer mesh.
 - FLUENT offers dynamic mesh adaption which automatically changes the mesh according to user-defined criteria.
 - Continue calculation until convergence.
 - Compare the results obtained on the different meshes.
 - Repeat the procedure if necessary.
- ◆ To use a different mesh on a single problem, use the TUI commands **file/write-bc** and **file/read-bc** to facilitate the setup of a new problem. Better initialization can be obtained via interpolation from existing case/data by using 

Outline

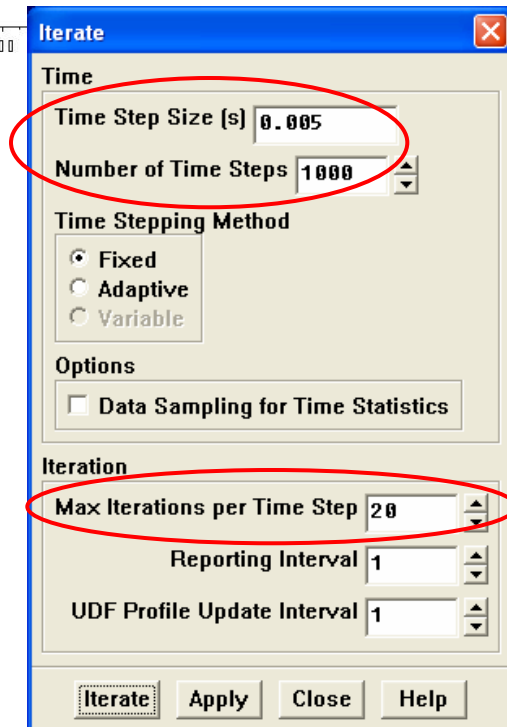
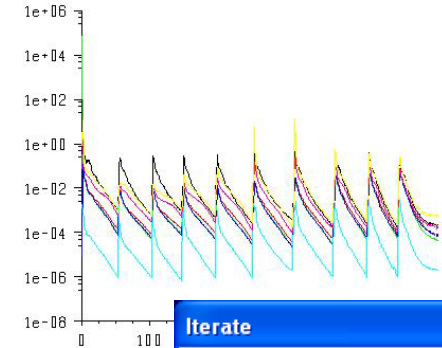
- ◆ Using the Solver
 - Setting Solver Parameters
 - Convergence
 - Definition
 - Monitoring
 - Stability
 - Accelerating Convergence
 - Accuracy
 - Grid Independence
 - Grid Adaption
 - Unsteady Flow Modeling
 - Unsteady flow problem setup
 - Unsteady flow modeling options
 - Summary
 - Appendix

Unsteady Flow Modeling

- ◆ Solver iterates to convergence within each time step, then advances to the next.
- ◆ Solution initialization defines the initial condition and it must be realistic.
- ◆ Non-iterative Time Advancement (NITA) is available for faster computation time (see the Appendix for details).
- ◆ For the pressure-based solver:
 - Time step size, Δt , is set in the Iterate panel
 - Δt must be small enough to resolve time-dependent features; make sure the convergence is reached within the number of Max Iterations per Time Step
 - The order of magnitude of an appropriate time step size can be estimated as:

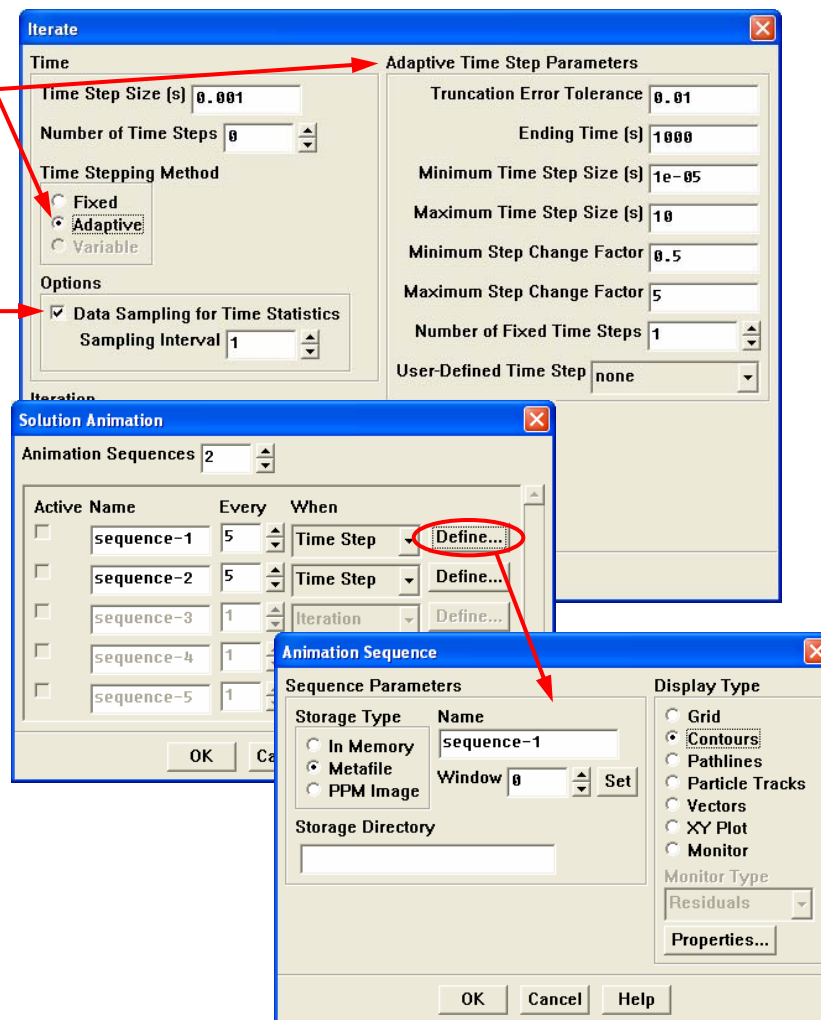
$$\Delta t \approx \frac{\text{Typical cell size}}{\text{Characteristic flow velocity}}$$

- Time step size estimate can also be chosen so that the unsteady characteristics of the flow can be resolved (e.g. flow within a known period of fluctuations)
- To iterate without advancing in time, use zero time steps
- The PISO scheme may aid in accelerating convergence for many unsteady flows



Unsteady Flow Modeling Options

- ◆ **Adaptive Time Stepping**
 - Automatically adjusts time-step size based on local truncation error analysis
 - Customization possible via UDF
- ◆ **Time-averaged statistics**
 - Particularly useful for LES turbulence modeling
- ◆ If desirable, animations should be set up before iterating (for flow visualization)
- ◆ For the density-based solver, the Courant number defines:
 - The global time step size for density-based explicit solver.
 - The pseudo time step size for density-based implicit solver
 - Real time step size must still be defined in the Iterate panel



Summary

- ◆ Solution procedure for both the pressure-based and density-based solvers is identical.
 - Calculate until you get a converged solution
 - Obtain a second-order solution (recommended)
 - Refine the mesh and recalculate until a grid-independent solution is obtained.
- ◆ All solvers provide tools for judging and improving convergence and ensuring stability.
- ◆ All solvers provide tools for checking and improving accuracy.
- ◆ Solution accuracy will depend on the appropriateness of the physical models that you choose and the boundary conditions that you specify.

Appendix

◆ Background

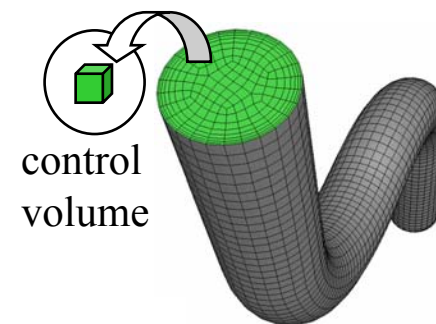
- Finite Volume Method
- Explicit vs. Implicit
- Segregated vs. Coupled
- Transient Solutions
- Flow Diagrams of NITA and ITA Schemes

The Finite Volume Method

- ◆ FLUENT solvers are based on the finite volume method.
 - Domain is discretized into a finite set of control volumes or cells.
- ◆ The general transport equation for mass, momentum, energy, etc. is applied to *each* cell and discretized.
- ◆ All equations are solved in order to render the flow field.

$$\underbrace{\frac{\partial}{\partial t} \int_V \rho \phi dV}_{\text{Unsteady}} + \underbrace{\oint_A \rho \phi \mathbf{V} \cdot d\mathbf{A}}_{\text{Convection}} = \underbrace{\oint_A \Gamma \nabla \phi \cdot d\mathbf{A}}_{\text{Diffusion}} + \underbrace{\int_V S_\phi dV}_{\text{Generation}}$$

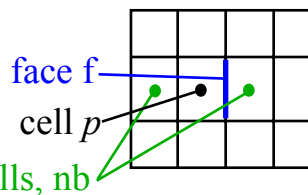
<u>Equation</u>	<u>Variable</u>
Continuity	ρ
X momentum	u
Y momentum	v
Z momentum	w
Energy	h



Fluid region of pipe flow discretized into finite set of control volumes (mesh).

The Finite Volume Method

- ◆ Each transport equation is discretized into algebraic form. For cell P,

$$\frac{(\rho\phi_p)^{t+\Delta t} - (\rho\phi_p)^t}{\Delta t} \Delta V + \sum_{\text{faces}} \rho_f \phi_f V_f A_f = \sum_{\text{faces}} \Gamma_f (\nabla\phi)_{\perp,f} A_f + S_\phi \Delta V$$


- ◆ Discretized equations require information at both cell centers and faces.
 - Field data (material properties, velocities, etc.) are stored at cell centers.
 - Face values are interpolated in terms of local and adjacent cell values.
 - Discretization accuracy depends on the “stencil” size.
- ◆ The discretized equation can be expressed simply as

$$a_p \phi_p + \sum_{nb} a_{nb} \phi_{nb} = b_p$$

- Equation is written for every control volume in the domain resulting in an equation set.

Linearization

- ◆ Equation sets are solved iteratively.
 - Coefficients a_p and a_{nb} are typically functions of solution variables (nonlinear and coupled).
 - Coefficients are written to use values of solution variables from the previous iteration.
 - Linearization removes the coefficients' dependence on ϕ .
 - Decoupling removes the coefficients' dependence on other solution variables.
 - Coefficients are updated with each outer iteration.
 - For a given inner iteration, coefficients are constant (frozen).
 - ◆ ϕ_p can either be solved explicitly or implicitly.

$$a_p \phi_p + \sum_{nb} a_{nb} \phi_{nb} = b_p$$

Explicit vs. Implicit Solution

- ◆ Assumptions are made about the knowledge of ϕ_{nb} .
 - Explicit linearization
 - Unknown value in each cell computed from relations that include only existing values (ϕ_{nb} assumed known from previous iteration).
 - ϕ_p is then solved explicitly using a Runge-Kutta scheme.
 - Implicit linearization
 - ϕ_p and ϕ_{nb} are assumed unknown and are solved using linear equation techniques.
 - Equations that are implicitly linearized tend to have less restrictive stability requirements.
 - The equation set is solved simultaneously using a second iterative loop (e.g., point Gauss-Seidel).

Pressure-Based vs. Density-Based Solver

◆ Pressure-based solver

- If the only unknowns in a given equation are assumed to be for a single variable, then the equation set can be solved without regard to the solution of other variables.
- Simply put, each governing equation is solved independently of the other equations).
- In this case, the coefficients a_p and a_{nb} are scalar values.

$$a_p \phi_p + \sum_{nb} a_{nb} \phi_{nb} = b_p$$

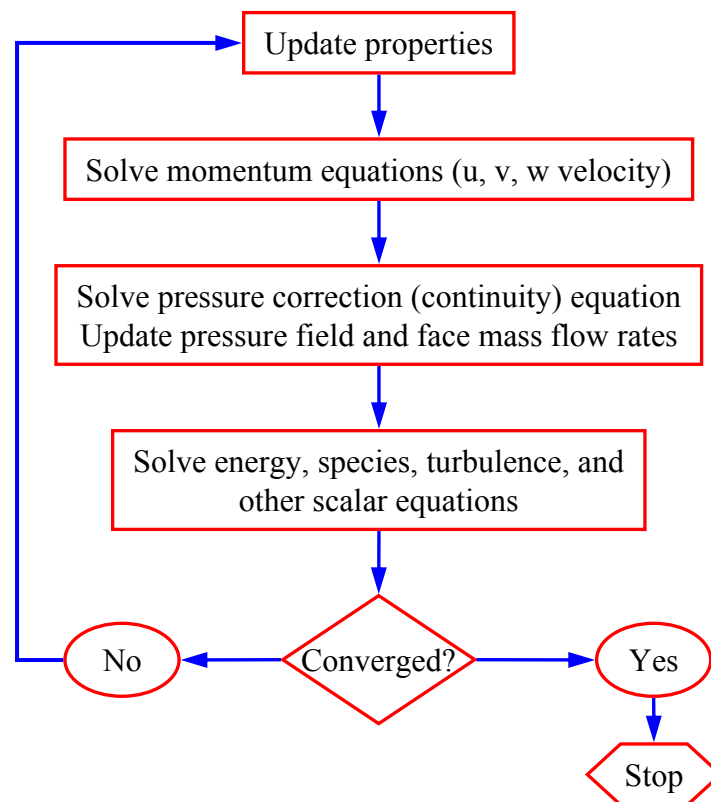
◆ Density-based solver

- If more than one variable is unknown in each equation, and each variable is defined by its own transport equation, then the equation set is coupled together.
- In this case, the coefficients a_p and a_{nb} are $N_{eq} \times N_{eq}$ matrices.
- ϕ is a vector of the dependent variables, $\{p, u, v, w, T, Y\}_T$

Pressure-Based Solver

- ◆ In the pressure-based solver, each equation is solved separately.
- ◆ The continuity equation takes the form of a pressure correction equation as part of Patankar's SIMPLE algorithm.
- ◆ Under-relaxation factors are included in the discretized equations.
 - Included to improve stability of iterative process.
 - An explicit under-relaxation factor, α , limits change in variable from one iteration to the next:

$$\phi_p = \phi_{p,\text{old}} + \alpha \Delta\phi_p$$

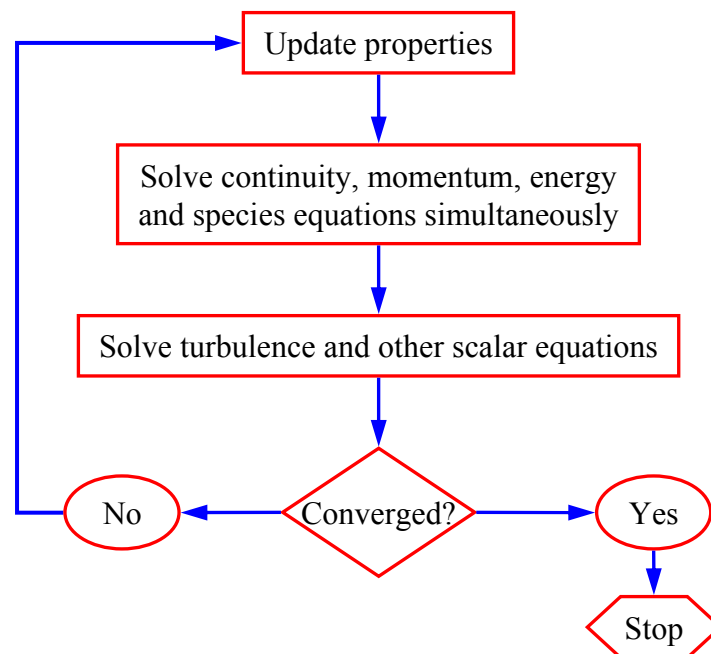


Density-Based Solver

- ◆ Continuity, momentum, energy, and species are solved simultaneously in the density-based solver.
- ◆ Equations are modified to resolve both compressible and incompressible flow.
- ◆ Transient term is always included.
 - Steady-state solution is formed as time increases and transients tend to zero.
- ◆ For steady-state problem, the “time step” is defined by the Courant number.
 - Stability issues limit the maximum time step size for the explicit solver but not for the implicit solver.

$$\Delta t = \frac{(\text{CFL}) \Delta x}{U}$$

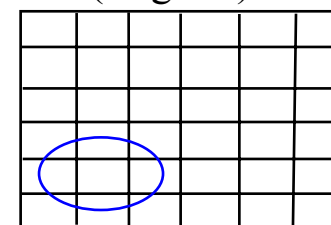
CFL = Courant-Friedrichs-Lewy-number
 u = appropriate velocity scale
 Δx = grid spacing



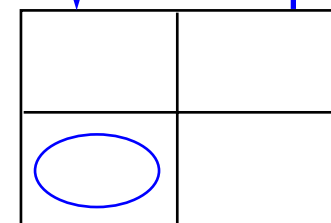
Multigrid Solver

- ◆ The Multigrid solver accelerates convergence by solving the discretized equations on multiple levels of mesh density so that the “low-frequency” errors of the approximate solution can be efficiently eliminated
 - Influence of boundaries and far-away points are more easily transmitted to interior of coarse mesh than on fine mesh.
 - Coarse mesh defined from original mesh
 - Multiple coarse mesh ‘levels’ can be created.
 - ◆ Algebraic Multigrid (AMG) – coarse mesh emulated algebraically
 - ◆ Full Approximate Storage Multigrid (FAS) – ‘cell coalescing’ defines new grid.
 - An option in the density-based explicit solver.
Final solution is for original mesh
 - Multigrid solver operates automatically in the background
- ◆ Consult the FLUENT User’s Guide for additional options and technical details

Fine (original) mesh



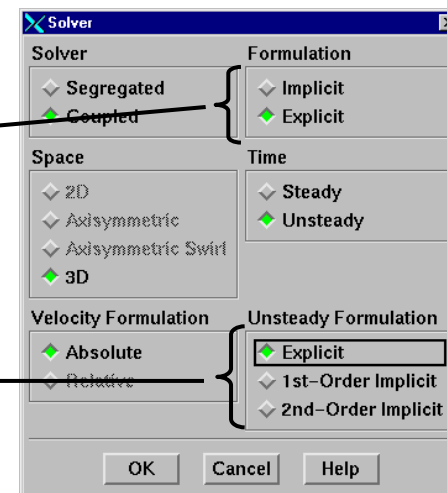
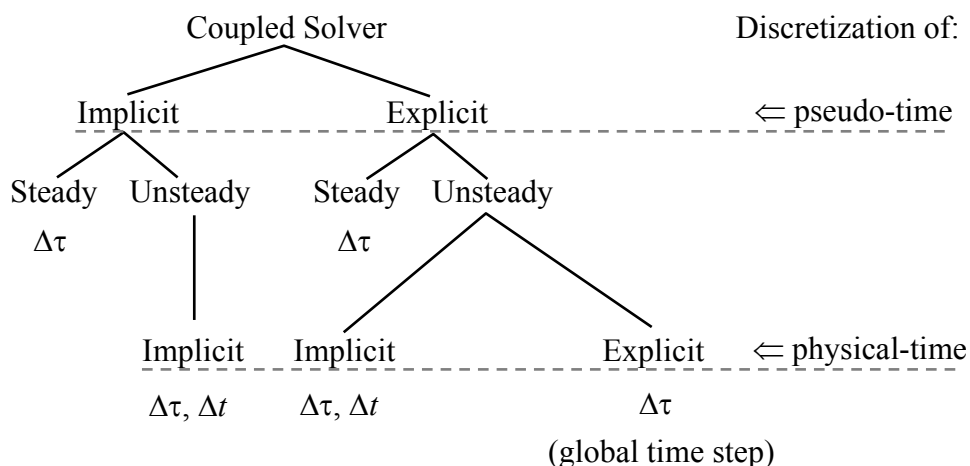
“Solution
Transfer”



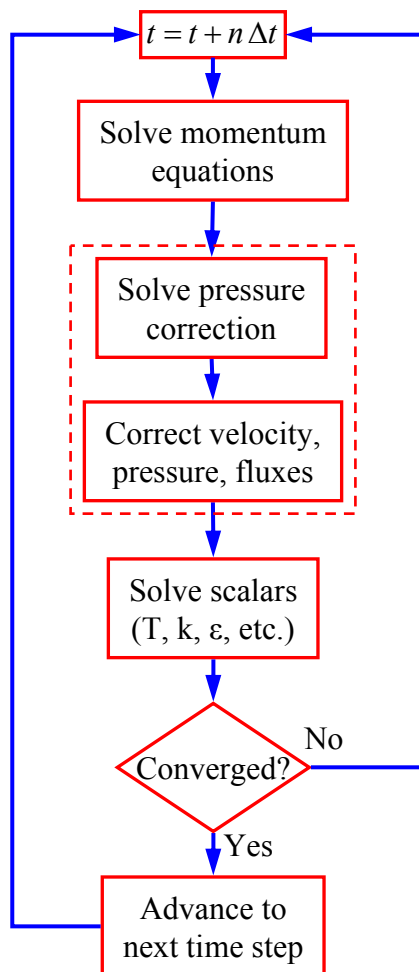
coarse mesh

Background: Coupled/Transient Terms

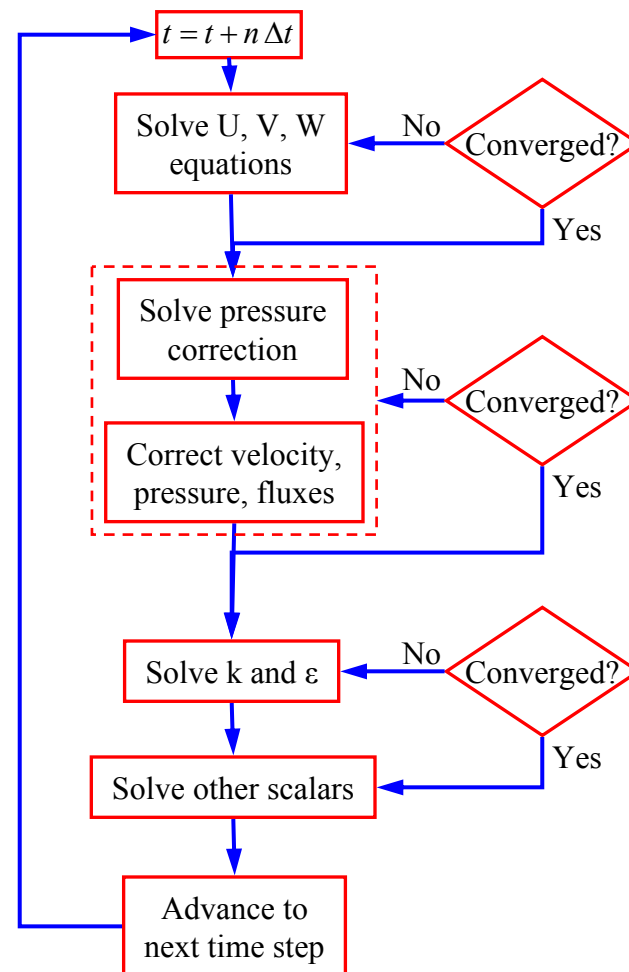
- ◆ Coupled solver equations always contain a transient term.
- ◆ Equations solved using the unsteady coupled solver may contain *two* transient terms:
 - Pseudo-time term, $\Delta\tau$.
 - Physical-time term, Δt .
- ◆ Pseudo-time term is driven to near zero at each time step and for steady flows.
- ◆ Flow chart indicates which time step size inputs are required.
 - Courant number defines $\Delta\tau$
 - Inputs to **Iterate** panel define Δt .



ITA versus NITA



Iterative Time Advancement (ITA)



Non-Iterative Time Advancement (NITA)

NITA Schemes for the Pressure-Based Solver

Overall time-discretization error
for 2nd-order scheme:
 $O(\Delta t^2)$

=

Truncation error:
 $O(\Delta t^2)$

+

Splitting error (due to eqn
segregation): $O(\Delta t^n)$

- ◆ Non-iterative time advancement (NITA) schemes reduce the splitting error to $O(\Delta t^2)$ by using sub-iterations (not the more expensive outer iterations to eliminate the splitting errors used in ITA) per time step.
- ◆ NITA runs about twice as fast as the ITA scheme.
- ◆ Two flavors of NITA schemes available in FLUENT 6.3:
 - PISO (NITA/PISO)
 - Energy and turbulence equations are still loosely coupled.
 - Fractional-step method (NITA/FSM)
 - About 20% cheaper than NITA/PISO on a per time-step basis.
- ◆ NITA schemes have a wide range of applications for unsteady simulations, such as incompressible, compressible (subsonic, transonic), turbomachinery flows, etc.
- ◆ NITA schemes are not available for multiphase (except VOF), reacting flows, porous media, and fan models, etc. Consult the FLUENT User's Guide for additional details.

NITA Solution Control and Monitoring

- ◆ Sub-iterations are performed for discretized equations till the Correction Tolerance is met or the number of sub-iterations has reached the Max Corrections
- ◆ Algebraic multigrid (AMG) cycles are performed for each sub-iteration. AMG cycles terminate if the default AMG criterion is met or the Residual Tolerance is satisfied for the last sub-iteration
- ◆ Relaxation Factor is used for solutions between each sub-iteration

