Lecture (2 - 3): Numerical methods and Types of CFD codes

2-1 Numerical methods

- The continuous Initial Boundary Value Problems (IBVPs) are discretized into algebraic equations using numerical methods. Assemble the system of algebraic equations and solve the system to get approximate solutions
- **Numerical methods include:**

1. Discretization methods
2. Solvers and numerical parameters
3. Grid generation and transformation
4. High Performance Computation (HPC) and post-processing

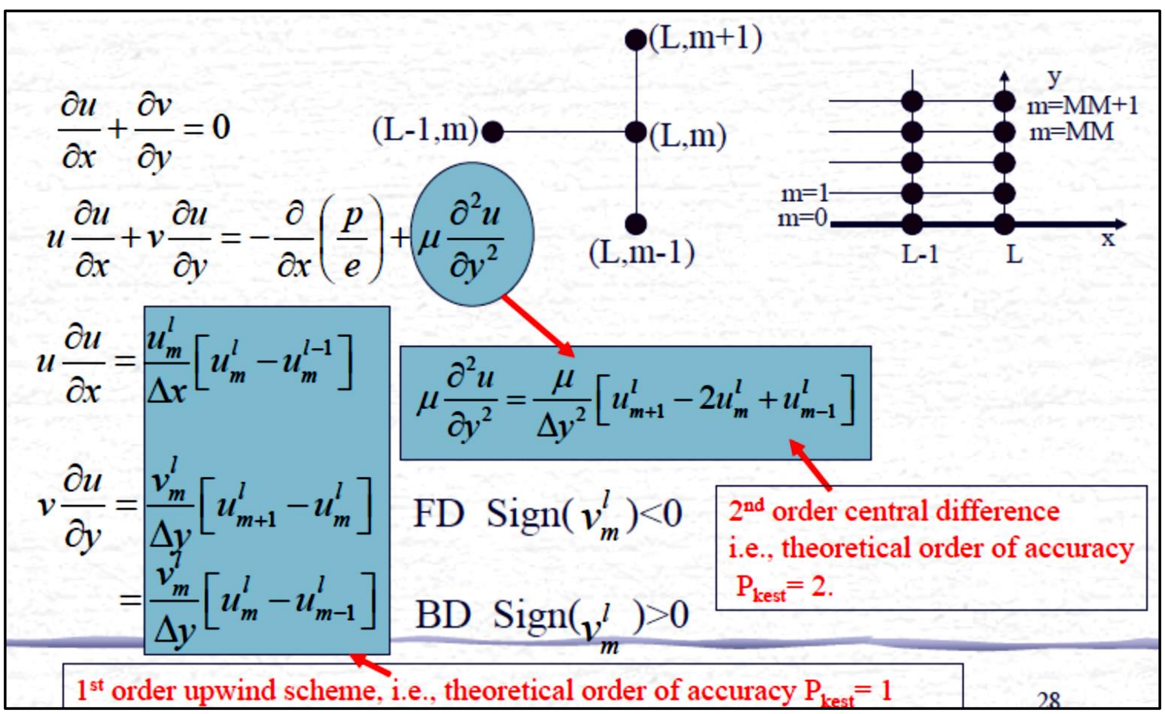
2.1.1 Discretization methods

- Finite difference methods (straightforward to apply, usually for regular grid) and finite volumes and finite element methods (usually for irregular meshes)
- Each type of methods above yields the same solution if the grid is fine enough. However, some methods are more suitable to some cases than others
- Finite difference methods for spatial derivatives with different order of accuracies can be derived using Taylor expansions, such as 2ndorder upwind scheme, central differences schemes, etc.
- Higher order numerical methods usually predict higher order of accuracy for CFD, but more likely unstable due to less numerical dissipation
- Temporal derivatives can be integrated either by the explicit method (Euler, Runge-Kutta, etc.) or implicit method (e.g. Beam-Warming method)

- Explicit methods can be easily applied but yield conditionally stable Finite Different Equations (FDEs), which are restricted by the time step; Implicit methods are unconditionally stable, but need efforts on efficiency.
- Usually, higher-order temporal discretization is used when the spatial discretization is also of higher order.
- Stability: A discretization method is said to be stable if it does not magnify the errors that appear in the course of numerical solution process.
- Pre-conditioning method is used when the matrix of the linear algebraic system is ill-posed, such as multi-phase flows, flows with a broad range of Mach numbers, etc.
- Selection of discretization methods should consider efficiency, accuracy and special requirements, such as shock wave tracking

2.2. Discretization methods (example)

- 2D incompressible laminar flow boundary layer



$$\underbrace{\left[\frac{u_m^l}{\Delta x}+v_m^l\begin{matrix}-\frac{1}{\Delta y}FD\\ \hline \frac{1}{\Delta y}BD\end{matrix}-\frac{2\mu}{\Delta y^2}\right]}_{B_2}u_m^l+\underbrace{\left[\frac{\mu}{\Delta y^2}+\frac{v_m^l}{\Delta y}FD\right]}_{B_3}u_{m+1}^l+\underbrace{\left[\frac{\mu}{\Delta y^2}-\frac{v_m^l}{\Delta y}BD\right]}_{B_1}u_{m-1}^l$$

$$=\underbrace{\frac{u_m^l}{\Delta x}}_{B_4}u_m^{l-1}-\frac{\partial}{\partial x}(p/e)_m^l$$

$$B_1u_{m-1}^l+B_2u_m^l+B_3u_{m+1}^l=B_4u_m^{l-1}-\frac{\partial}{\partial x}(p/e)_m^l$$

$$\begin{bmatrix} B_2 & B_3 & 0 & 0 & 0 & 0 & 0 & 0\\ B_1 & B_2 & B_3 & 0 & 0 & 0 & 0 & 0\\ & & \bullet & \bullet & \bullet & \bullet & & \\ 0 & 0 & 0 & 0 & 0 & B_1 & B_2 & B_3\\ 0 & 0 & 0 & 0 & 0 & 0 & B_1 & B_2 \end{bmatrix}\times\begin{bmatrix} u_1^l\\ \bullet\\ \bullet\\ \bullet\\ u_{mm}^l\end{bmatrix}=\begin{bmatrix} B_4u_1^{l-1}-\frac{\partial}{\partial x}\left(\frac{p}{e}\right)_1^l\\ \bullet\\ \bullet\\ \bullet\\ B_4u_{mm}^{l-1}-\frac{\partial}{\partial x}\left(\frac{p}{e}\right)_{mm}^l\end{bmatrix}$$

Solve it using Thomas algorithm

To be stable, Matrix has to be Diagonally dominant.

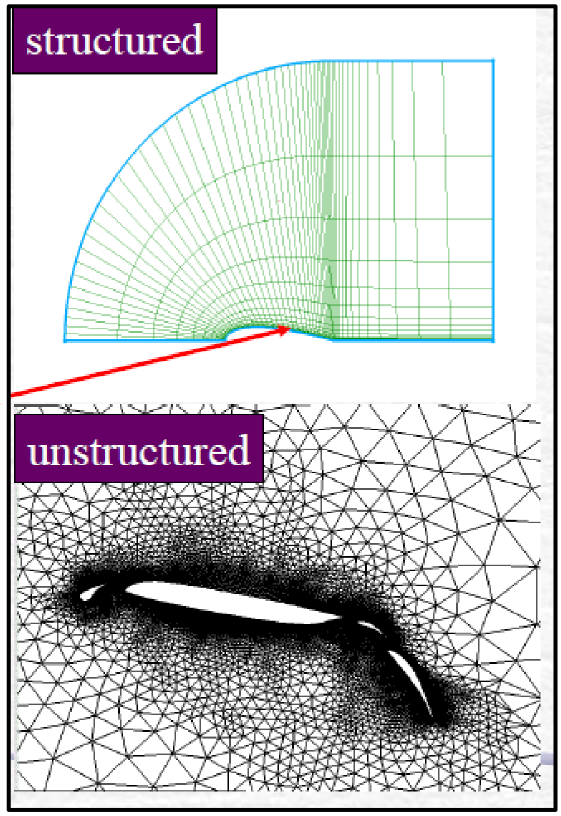
2.3 Solvers and numerical parameters

- **Solvers**include: tridiagonal, pentadiagonalsolvers, PETSC solver, solution-adaptive solver, multi-grid solvers, etc.
- **Solvers**can be either direct (Cramer’s rule, Gauss elimination, LU decomposition) or iterative (Jacobi method, Gauss-Seidel method, SOR method)
- **Numerical parameters**need to be specified to control the calculation. •Under relaxation factor, convergence limit, etc.
- Different numerical schemes
- Monitor residuals (change of results between iterations)
- Number of iterations for steady flow or number of time steps for unsteady flow
- Single/double precisions

2.4 Numerical methods (grid generation)

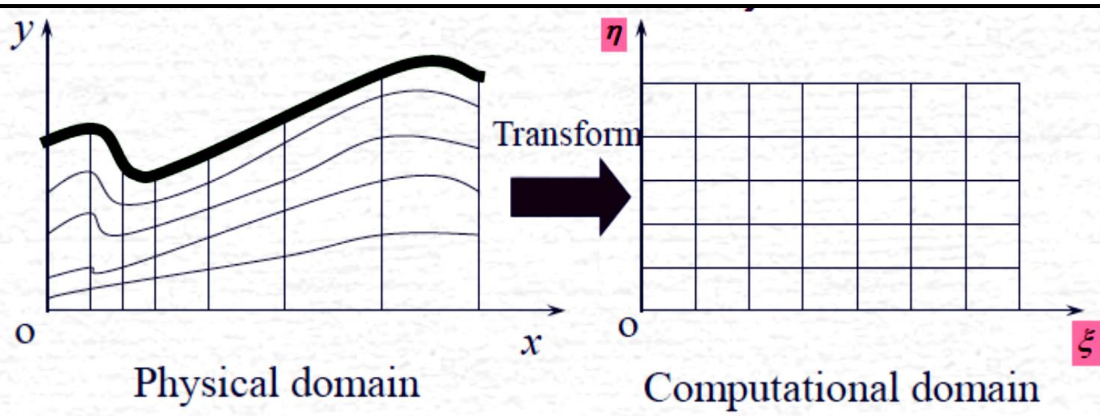
- Grids can either be structured (hexahedral) or unstructured (tetrahedral). Depends upon type of discretization scheme and application
- Scheme

- ○ Finite differences: structured
- ○ Finite volume or finite element: structured or unstructured
- Application



❑ Thin boundary layers best resolved with highly-stretched structured grids

□Unstructured grids useful for complex geometries
□Unstructured grids permit automatic adaptive refinement based on the pressure gradient, or regions interested (FLUENT)



•Transformation between physical (x,y,z) and computational (ξ,η,ζ) domains, important for body-fitted grids. The partial derivatives at these two domains have the relationship (2D as an example)

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial \xi}\frac{\partial \xi}{\partial x} + \frac{\partial f}{\partial \eta}\frac{\partial \eta}{\partial x} = \xi_x \frac{\partial f}{\partial \xi} + \eta_x \frac{\partial f}{\partial \eta}$$

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial \xi}\frac{\partial \xi}{\partial y} + \frac{\partial f}{\partial \eta}\frac{\partial \eta}{\partial y} = \xi_y \frac{\partial f}{\partial \xi} + \eta_y \frac{\partial f}{\partial \eta}$$

2.5 High performance computing

- CFD computations (e.g. 3D unsteady flows) are usually very expensive which requires parallel high performance supercomputers with the use of multi-block technique.
- As required by the multi-block technique, CFD codes need to be developed using the Massage Passing Interface (MPI) Standard to transfer data between different blocks.
- Emphasis on improving:
- Strong scalability, main bottleneck pressure Poisson solver for incompressible flow.
- Weak scalability, limited by the memory requirements.

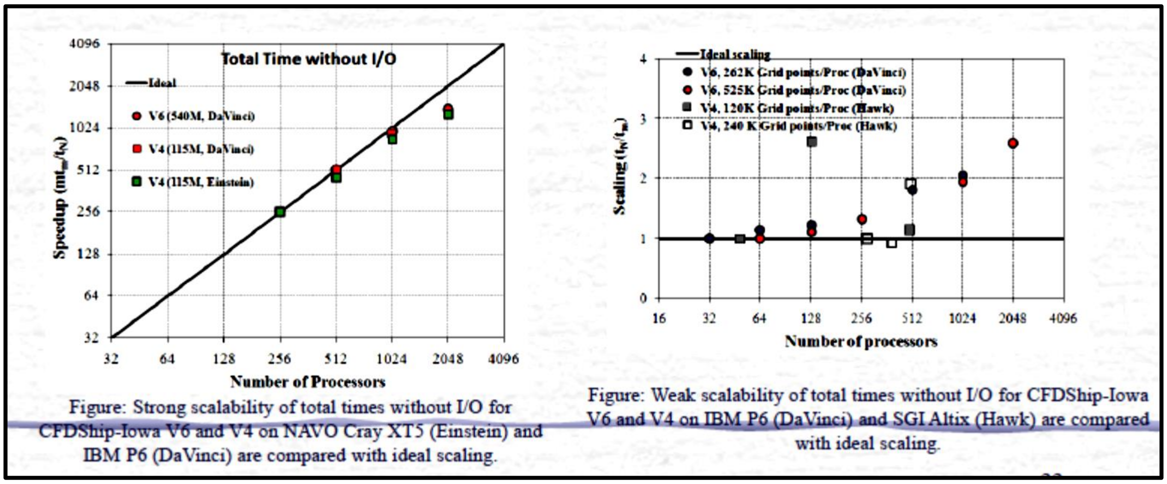


Figure: Strong scalability of total times without I/O for CFDShip-Iowa V6 and V4 on NAVO Cray XT5 (Einstein) and IBM P6 (DaVinci) are compared with ideal scaling.

Figure: Weak scalability of total times without I/O for CFDShip-Iowa V6 and V4 on IBM P6 (DaVinci) and SGI Altix (Hawk) are compared with ideal scaling.

2.5. Post-Processing

- Post-processing:

  1. Visualize the CFD results (contour, velocity vectors, streamlines, path lines, streak lines, and iso-surface in 3D, etc.), and
  2. CFD UA: verification and validation using EFD data (more details later)

- Post-processing usually through using commercial software

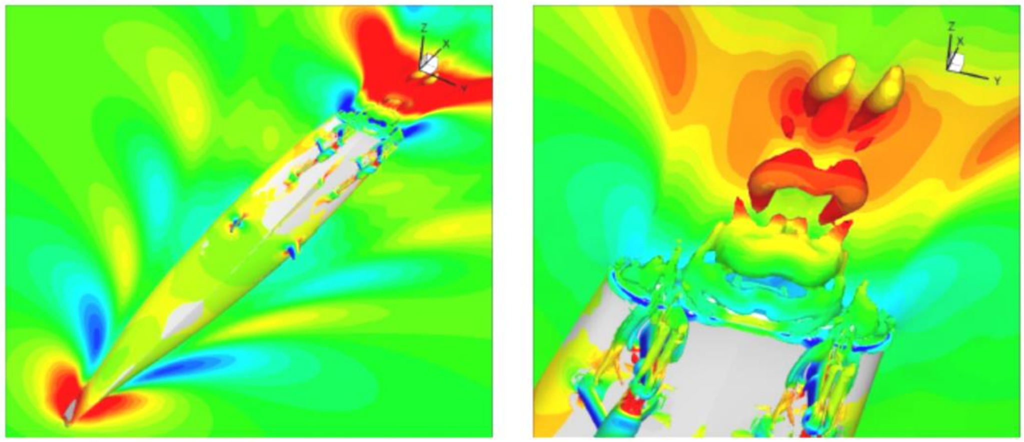


Figure: Isosurface of Q=300 colored using piezometric pressure. free=surface colored using z for fully appended Athena. Fr=0.25, $Re=2.9\times10^8$, Tecplot360 is used for visualization.

3. Types of CFD codes

- Commercial CFD code: ANSYS FLUENT, Star-CCM+, CFDRC, CFX/AEA, etc.
- Research CFD code: CFDSHIP-IOWA
- Public domain software(PHI3D, HYDRO, and WinpipeD, etc.)
- Other CFD software includes the Grid generation software (e.g. Pointwise (gridgen), Gambit) and flow visualization software (e.g. Tecplot, Paraview, ANSYS EnSight, FieldView).

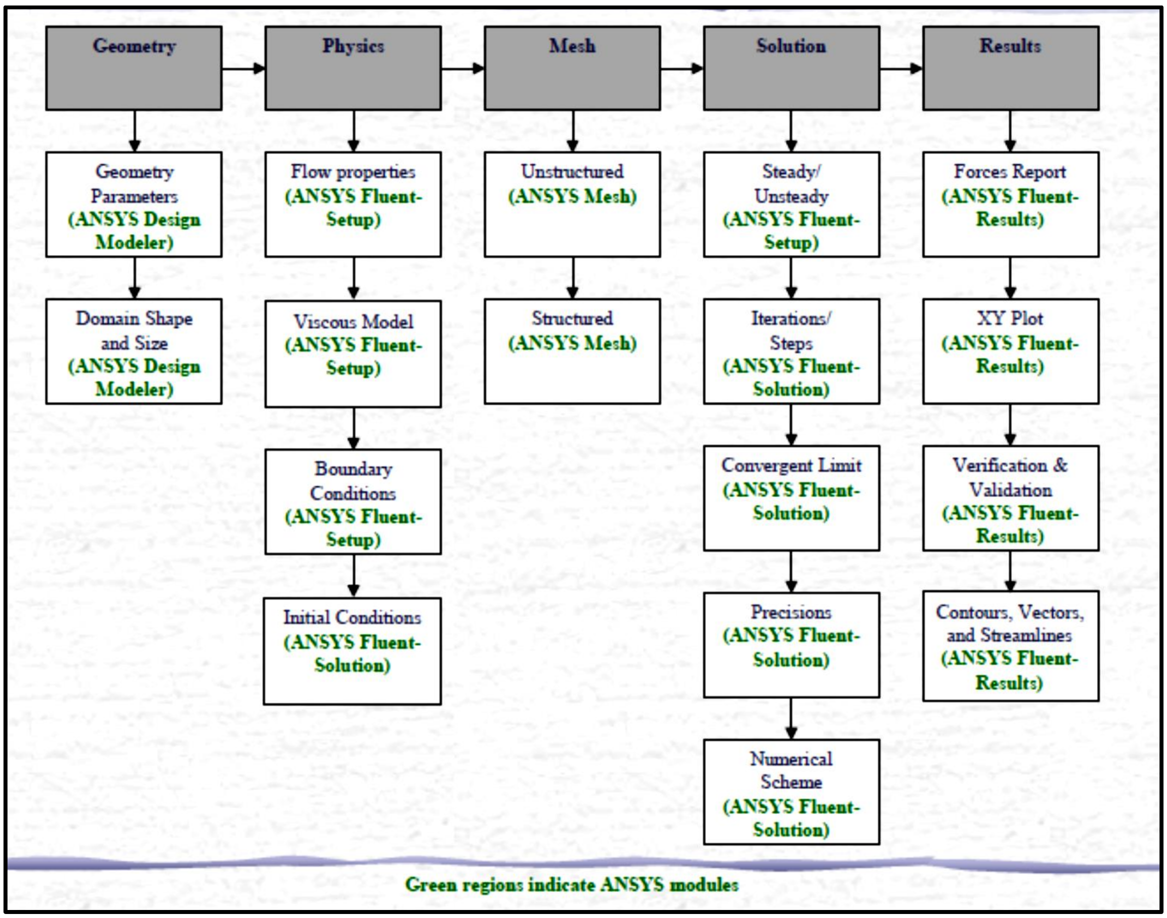
3.1 CFD process

- Purposes of CFD codes will be different for different applications: investigation of bubble-fluid interactions for bubbly flows, study of wave induced massively separated flows for free-surface, etc.
- Depend on the specific purpose and flow conditions of the problem, different CFD codes can be chosen for different applications (aerospace, marines, combustion, multi-phase flows, etc.)
- Once purposes and CFD codes chosen, “CFD process” is the steps to set up the IBVP problem and run the code:

  1. Geometry
  2. Physics (Setup)

3. Mesh
4. Solve
5. Results

3.2 CFD Process



3.3 Geometry

- Selection of an appropriate coordinate

- Determine the domain size and shape
- Any simplifications needed?
- What kinds of shapes needed to be used to best resolve the geometry? (lines, circular, ovals, etc.)
- For commercial code, geometry is usually created using commercial software (either separated from the commercial code itself, like Gambit)
- For research code, commercial software (e.g. pointwise) is used.

3.4 Physics (Setup)

Flow conditions and fluid properties

1. **Flow conditions**: inviscid, viscous, laminar, turbulent, single-phase, multi-phase, and phase change, etc.
2. **Fluid properties**: density, viscosity, surface tension, and thermal conductivity, etc.
3. Flow conditions and properties usually presented in dimensional form in industrial commercial CFD software, whereas in non-dimensional variables for research codes.

- Selection of models: different models usually fixed by codes, options for user to choose
- Initial and Boundary Conditions: not fixed by codes, user needs specify them for different applications.

3.5 Mesh

- Meshes should be well designed to resolve important flow features which are dependent upon flow condition parameters (e.g., Re), such as the grid refinement inside the wall boundary layer
- Mesh can be generated by either commercial codes (Pointwise/Gridgen, Gambit, etc.) or research code (using algebraic vs. PDE based, conformal mapping, etc.)

- The mesh, together with the boundary conditions need to be exported from commercial software in a certain format that can be recognized by the research CFD code or other commercial CFD software.

2.6 Solve

•Setup appropriate numerical parameters•Choose appropriate Solvers

•Solution procedure (e.g. incompressible flows)

Solve the momentum, pressure Poisson equations and get flow field quantities, such as velocity, turbulence intensity, pressure and integral quantities (lift, drag forces)

2.7 Results

- Reports the saved time history of the residuals of the velocity, pressure and temperature, etc.
- Report the integral quantities, such as total pressure drop, friction factor (pipe flow), lift and drag coefficients (airfoil flow), etc.
- XY plots could present the centerline velocity/pressure distribution, friction factor distribution (pipe flow), pressure coefficient distribution (airfoil flow).
- AFD or EFD data can be imported and put on top of the XY plots for validation

2.7.1 Analysis and visualization

- Calculation of derived variables
  - Vorticity
  - Wall shear stress
- Calculation of integral parameters: forces, moments
- Visualization (usually with commercial software)Simple 2D contours
  - 3D contour iso surface plots

- Vector plots and streamlines (streamlines are the lines whose tangent direction is the same as the velocity vectors)
- Animations

2.8 Results (Uncertainty Assessment)

Simulation error: the difference between a simulation result S and the truth T (objective reality), assumed composed of additive modeling δ_{SM} and numerical δ_{SN} errors:

$$\text{Error:} \quad \delta_S = S - T = \delta_{SM} + \delta_{SN} \qquad \text{Uncertainty:} \quad U_S^2 = U_{SM}^2 + U_{SN}^2$$

- Verification: process for assessing simulation numerical uncertainties USN and, when conditions permit, estimating the sign and magnitude Delta δ*SN of the simulation numerical error itself and the uncertainties in that error estimate USN

$$\delta_{SN} = \delta_I + \delta_G + \delta_T + \delta_P = \delta_I + \sum_{j=1}^{J} \delta_j \qquad U_{SN}^2 = U_I^2 + U_G^2 + U_T^2 + U_P^2$$

I: Iterative, G: Grid, T: Time step, P: Input parameters

Iterative convergence requires UI to be at least one order of magnitude smaller than U Gand UT.

- Validation: process for assessing simulation modeling uncertainty USM by using benchmark experimental data and, when conditions permit, estimating the sign and magnitude of the modeling error δ_{SM} itself.

$$U_V^2 = U_D^2 + U_{SN}^2$$

$$E = D - S = \delta_D - (\delta_{SM} + \delta_{SN}) \qquad |E| < U_V \quad \text{Validation achieved}$$

D: EFD Data; *UV:* Validation Uncertainty