Solving Fluid Flow Equations

Pressure-Velocity Coupling

Hrvoje Jasak

hrvoje.jasak@fsb.hr

FSB, University of Zagreb, Croatia

Compressible Navier-Stokes Equations

Compressible Formulation of Navier-Stokes Equations

- Solution variables: density ρ, momentum $\rho\mathbf{u}$ and energy ρe
- Continuity equation:

$$\frac{\partial \rho}{\partial t} + \nabla_{\bullet}(\rho\mathbf{u}) = 0$$

  - Rate of change and convection: mass transport. The two terms are sometimes grouped into a **substantial derivative**
  - Mass sources and sinks would appear on the r.h.s.
  - Note the absence of a diffusion term: mass does not diffuse
  - Coupling with the momentum equation: rate of change of ρ depends on the divergence of momentum $\rho\mathbf{u}$
- Momentum equation:

$$\frac{\partial(\rho\mathbf{u})}{\partial t} + \nabla_{\bullet}(\rho\mathbf{u}\mathbf{u}) - \nabla_{\bullet}\left[\mu\left(\nabla\mathbf{u} + (\nabla\mathbf{u})^T\right)\right] = \rho\mathbf{g} - \nabla\left(P + \frac{2}{3}\mu\nabla_{\bullet}\mathbf{u}\right)$$

  - Non-linear convection term: $\nabla_{\bullet}(\rho\mathbf{u}\mathbf{u})$. This term provides the wealth of interaction present in fluid flows, e.g. vorticity, turbulence cascade
  - Diffusion term $\nabla_{\bullet}(\mu\,\nabla\mathbf{u})$ contains viscous effects

Compressible Navier-Stokes Equations

Compressible Formulation of Navier-Stokes Equations

- Energy equation:

$$\frac{\partial(\rho e)}{\partial t} + \nabla\bullet(\rho e\mathbf{u}) - \nabla\bullet(\lambda\nabla T) = \rho\mathbf{g}\bullet\mathbf{u} - \nabla\bullet(P\,\mathbf{u}) - \nabla\bullet\left(\frac{2}{3}\,\mu(\nabla\bullet\mathbf{u})\,\mathbf{u}\right) + \nabla\bullet\left[\mu\left(\nabla\mathbf{u} + (\nabla\mathbf{u})^T\right)\bullet\mathbf{u}\right] + \rho Q,$$

  - The diffusion term is given in terms of temperature T, not energy: for non-constant material properties, this may be problematic
  - r.h.s. contains a number of terms related to the work from the stress tensor
  - Weaker coupling to the rest of the system: e and T influence ρ and $\mathbf{u}$ through the equation of state

Compressible Navier-Stokes Equations

Compressible Formulation of Navier-Stokes Equations

- Equation of state:

$$\rho = \rho(P, T)$$

  - Relationship between density ρ and pressure P

- Transport coefficients λ and μ are also general functions of the thermodynamic state variables:

$$\lambda = \lambda(P, T),$$

$$\mu = \mu(P, T).$$

  - Properties of real gasses and liquids rarely used in tabular form. Instead, measured data is curve fitted be standard sources: JANAF, NIST, etc.
  - Variation of material properties is usually a smooth function and does not introduce significant non-linear problems. Issues sometimes occur when the state changes significantly in a single time-step. Here, the initial guess for the new state may be far away from the solution, causing excessive number of search iterations

Density-Based Algorithm

Block-Coupled Density-Based Solver

- Noting that all governing equations fit into the standard form and all variables are fully coupled, the compressible Navier-Stokes system can be written as:

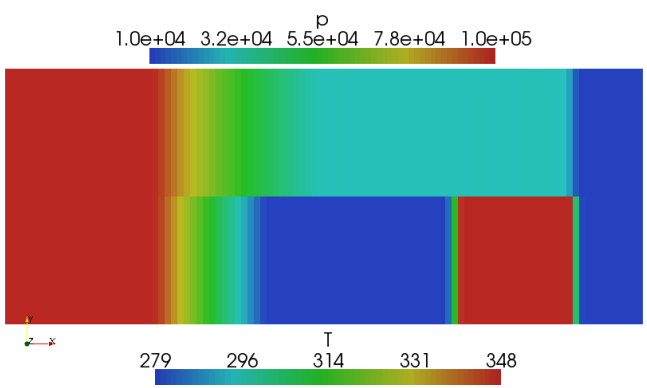
$$\frac{\partial U}{\partial t} + \nabla \bullet F - \nabla \bullet V = R$$

where the solution variable U is:

$$U = \begin{bmatrix} \rho \\ \rho \mathbf{u} \\ \rho e \end{bmatrix}$$

- Pressure appears in the convective flux F and V is the diffusive flux:

$$F = \begin{bmatrix} \rho \mathbf{u} \\ \rho \mathbf{u}\mathbf{u} + p\mathbf{I} \\ \rho(e+p)\mathbf{u} \end{bmatrix} \quad V = \begin{bmatrix} 0 \\ \boldsymbol{\sigma} \\ \boldsymbol{\sigma} \bullet \mathbf{u} - \mathbf{q} \end{bmatrix}$$



Density-Based Algorithm

Block-Coupled Density-Based Solver

- Standard (Roe flux) compressible Navier-Stokes solver will evaluate F for each cell face directly from the state (U) left and right from the face, using approximate Riemann solver techniques
- Looking at the second row of the flux expression we can recognise the convective contribution and the pressure driving force (note $\nabla \bullet (p\mathbf{I}) = \nabla p$). In high-speed flows, the first component is considerably larger than the second
- Effects of the diffusive flux are more benign, apart from the region next to the wall where the viscous effects dominate: **boundary layer**
- In the low-speed limit, a pressure difference of $3 - 5\,\mathrm{Pa}$ can drive considerable flow; however, in this case, the pressure gradient will dominate. As shown before, this implies a density change of approximately $5 \times 10^{-5}\,\mathrm{kg/m^3}$ for the mean density of $1\mathrm{kg/m^3}$. Equivalent calculation for a liquid (water), would produce even more extreme result (due to the higher speed of sound)
- For the incompressible limit where $Ma = 0$, the system breaks down: density ρ is no longer a function of the pressure p!

Incompressible Flow Formulation

Incompressible Flow Equations

- Decoupling dependence of density on pressure, also resulting in the decoupling of the energy equation from the rest of the system
- Equations can be solved both in the velocity-density or velocity-pressure formulation
  - Velocity-density formulation does not formally allow for $Ma = 0$ (or $c = \infty$), but formally this is never the case. In practice, matrix preconditioning techniques are used to overcome zero diagonal coefficients
  - Velocity-pressure formulation does not suffer from low-Ma limit, but performs considerably worse at high Ma number

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla_{\bullet}(\mathbf{u}\mathbf{u}) - \nabla_{\bullet}\left(\nu \nabla \mathbf{u}\right) = -\nabla p$$

$$\nabla_{\bullet}\mathbf{u} = 0$$

Pressure and Velocity

Pressure – Momentum Interaction

- Counting the equations and unknowns, the system seems well posed: 1 vector and 1 scalar field governed by 1 vector and 1 scalar equation
- Linear coupling exists between the momentum equation and continuity. Note that $\mathbf{u}$ is a vector variable governed by the vector equation. Continuity equation imposes an additional criterion on velocity divergence ($\nabla_\bullet \mathbf{u}$). This is an example of a scalar constraint on a vector variable, as $\nabla_\bullet \mathbf{u}$ is a scalar
- Non-linear $\mathbf{u} - \mathbf{u}$ interaction in the convection is unlikely to cause trouble: use an iterative solution technique. In practice

$$\nabla_\bullet(\mathbf{u}\mathbf{u}) \approx \nabla_\bullet(\mathbf{u}^o\mathbf{u}^n)$$

where $\mathbf{u}^o$ is the currently available solution or an initial guess and $\mathbf{u}^n$ is the “new” solution. The algorithm cycles until $\mathbf{u}^o = \mathbf{u}^n$

Derivation of the Pressure Equation

Pressure Equation as a Schur Complement

- Consider a general block matrix system M, consisting of 4 block matrices, A, B, C and D, which are respectively $p \times p$, $p \times q$, $q \times p$ and $q \times q$ matrices and A is invertible:

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

- This structure will arise naturally when trying to solve a block system of equations

$$Ax + By = a$$

$$Cx + Dy = b$$

- The Schur complement arises when trying to eliminate x from the system using partial Gaussian elimination by multiplying the first row with A^{-1}:

$$A^{-1}Ax + A^{-1}By = A^{-1}a$$

and

$$x = A^{-1}a - A^{-1}By$$

Derivation of the Pressure Equation

Pressure Equation as a Schur Complement

- Substituting the above into the second row:

$$(D - CA^{-1}B)y = b - CA^{-1}a$$

- Let us repeat the same set of operations on the block form of the pressure-velocity system, attempting to assemble a pressure equation. Note that the operators in the block system could be considered both as differential operators and in a discretised form

$$\begin{bmatrix} [A_{\mathbf{u}}] & [\nabla(.)] \\ [\nabla\bullet(.)] & [0] \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ p \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

- Formally, this leads to the following form of the pressure equation:

$$[\nabla\bullet(.)][A_{\mathbf{u}}^{-1}][\nabla(.)][p] = 0$$

Here, $A_{\mathbf{u}}^{-1}$ represent the inverse of the momentum matrix in the discretised form, which acts as diffusivity in the Laplace equation for the pressure.

Derivation of the Pressure Equation

Pressure Equation as a Schur Complement

- From the above, it is clear that the governing equation for the pressure is a Laplacian, with the momentum matrix acting as a diffusion coefficient
  - While $[A_{\mathbf{u}}]$ is a sparse matrix, its inverse is likely to be dense
  - Discretised form of the divergence and gradient operator are sparse and well-behaved. However, a triple product with $[A_{\mathbf{u}}^{-1}]$ would result in a dense matrix, making it expensive to solve
- The above can be remedied be decomposing the momentum matrix before the triple product into the diagonal part and off-diagonal matrix:

$$[A_{\mathbf{u}}] = [D_{\mathbf{u}}] + [LU_{\mathbf{u}}],$$

where $[D_{\mathbf{u}}]$ only contains diagonal entries. $[D_{\mathbf{u}}]$ is easy to invert and will preserve the sparseness pattern in the triple product.

Derivation of the Pressure Equation

Pressure Equation as a Schur Complement

- Revisiting saddle momentum equation before the formation of the Schur complement and moving the off-diagonal component of $[A_{\mathbf{u}}]$ onto r.h.s. yields:

$$\begin{bmatrix} [D_{\mathbf{u}}] & [\nabla(.)] \\ [\nabla\bullet(.)] & [0] \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ p \end{bmatrix} = \begin{bmatrix} -[LU_{\mathbf{u}}][\mathbf{u}] \\ 0 \end{bmatrix}$$

  A revised formulation of the pressure equation via a Schur's complement yields:

$$[\nabla\bullet(.)][D_{\mathbf{u}}^{-1}][\nabla(.)][p] = [\nabla\bullet(.)][D_{\mathbf{u}}^{-1}][LU_{\mathbf{u}}][\mathbf{u}]$$

  In both cases, matrix $[D_{\mathbf{u}}^{-1}]$ is simple to assemble.

- It follows that the pressure equation is a Poisson equation with the diagonal part of the discretised momentum acting as diffusivity and the divergence of the velocity on the r.h.s.

Derivation of the Pressure Equation

Standard Derivation of the Pressure Equation

- We shall now rewrite the above equations to formally derive an equation for the pressure. This may be done in several ways: formally correct involves a Schur's complement of the block pressure-velocity system
- Start by discretising the momentum equation using the techniques described for a scalar transport equation. For the purposes of derivation, the pressure gradient term will remain in the differential form. For each CV, the discretised momentum equation yields:

$$a_P^{\mathbf{u}} \mathbf{u}_P + \sum_N a_N^{\mathbf{u}} \mathbf{u}_N = \mathbf{r} - \nabla p$$

- For simplicity, we shall introduce the $\mathbf{H}(\mathbf{u})$ operator, containing the off-diagonal part of the momentum matrix and any associated r.h.s. contributions:

$$\mathbf{H}(\mathbf{u}) = \mathbf{r} - \sum_N a_N^{\mathbf{u}} \mathbf{u}_N$$

Derivation of the Pressure Equation

Standard Derivation of the Pressure Equation

- Using the above, it follows:

$$a_P^{\mathbf{u}} \mathbf{u}_P = \mathbf{H}(\mathbf{u}) - \nabla p$$

and

$$\mathbf{u}_P = (a_P^{\mathbf{u}})^{-1}(\mathbf{H}(\mathbf{u}) - \nabla p)$$

- Substituting the expression for $\mathbf{u}_P$ into the incompressible continuity equation $\nabla \bullet \mathbf{u} = 0$ yields

$$\nabla \bullet \left[(a_P^{\mathbf{u}})^{-1} \nabla p\right] = \nabla \bullet ((a_P^{\mathbf{u}})^{-1} \mathbf{H}(\mathbf{u}))$$

  This is the form of the pressure equation for incompressible fluid

- Note the implied decomposition of the momentum matrix into the diagonal and off-diagonal contribution, where $a_P^{\mathbf{u}}$ is an coefficient in $[D_{\mathbf{u}}]$ matrix and $\mathbf{H}(\mathbf{u})$ is the product $[LU_{\mathbf{u}}][\mathbf{u}]$

Derivation of the Pressure Equation

Assembly of Conservative Fluxes

- Pressure equation is derived from continuity: divergence-free velocity
- Looking at the discretised form of the continuity equation

$$\nabla \bullet \mathbf{u} = \sum_f \mathbf{s}_f \bullet \mathbf{u} = \sum_f F$$

where F is the face flux

$$F = \mathbf{s}_f \bullet \mathbf{u}$$

- Conservative face flux should be created from the solution of the pressure equation. Substituting expression for $\mathbf{u}$ into the flux equation, it follows:

$$F = -(a_P^{\mathbf{u}})^{-1} \mathbf{s}_f \bullet \nabla p + (a_P^{\mathbf{u}})^{-1} \mathbf{s}_f \bullet \mathbf{H}(\mathbf{u})$$

Derivation of the Pressure Equation

Assembly of Conservative Fluxes

- A part of the above, $(a_P^{\mathbf{u}})^{-1}\mathbf{s}_f \bullet \nabla p$ appears during the discretisation of the Laplacian, for each face:

$$(a_P^{\mathbf{u}})^{-1}\mathbf{s}_f \bullet \nabla p = (a_P^{\mathbf{u}})^{-1}\frac{|\mathbf{s}_f|}{|\mathbf{d}|}(p_N - p_P) = a_N^p(p_N - p_P)$$

  Here, $a_N^p = (a_P^{\mathbf{u}})^{-1}\frac{|\mathbf{s}_f|}{|\mathbf{d}|}$ is equal to the off-diagonal matrix coefficient in the pressure Laplacian

- Note that in order for the face flux to be conservative, assembly of the flux must ba completely consistent with the assembly of the pressure equation (e.g. non-orthogonal correction)

SIMPLE Algorithm

- This is the earliest pressure-velocity coupling algorithm: Patankar and Spalding, 1972 (Imperial College London): Semi-Implicit Algorithm for Pressure-Linked Equations
- Sequence of operations:
  1. Guess the pressure field p^*
  2. Solve the momentum equation using the guessed pressure. This step is called **momentum predictor**

$$a_P^{\mathbf{u}} \mathbf{u}_P = \mathbf{H}(\mathbf{u}) - \nabla p^*$$

  3. Calculate the new pressure based on the velocity field. This is called a **pressure correction** step

$$\nabla \bullet \left[(a_P^{\mathbf{u}})^{-1} \nabla p \right] = \nabla \bullet ((a_P^{\mathbf{u}})^{-1} \mathbf{H}(\mathbf{u}))$$

  4. Based on the pressure solution, assemble conservative face flux F

$$F = \mathbf{s}_f \bullet \mathbf{H}(\mathbf{u}) - a_N^p (p_N - p_P)$$

  5. Repeat to convergence

Under-Relaxation

Stability of the SIMPLE Iteration Sequence

- The algorithm in its base form produces a series of corrections on $\mathbf{u}$ and p. Unfortunately, in the above form it will diverge!
- Divergence is due to the fact that pressure correction contains both the pressure as a physical variable and a component which forces the discrete fluxes to become conservative
- In order to achieve convergence, **under-relaxation** is used:

$$p^{**} = p^{*} + \alpha_P (p - p^{*})$$

and

$$\mathbf{u}^{**} = \mathbf{u}^{*} + \alpha_U (\mathbf{u} - \mathbf{u}^{*})$$

where p and $\mathbf{u}$ are the solution of the pressure and momentum equations and $\mathbf{u}^*$ and p^* represent a series of pressure and velocity approximations. Note that in practice momentum under-relaxation is implicit and pressure (elliptic equation) is under-relaxed explicitly

$$\frac{a_P^{\mathbf{u}}}{\alpha_U}\mathbf{u}_P = \mathbf{H}(\mathbf{u}) - \nabla p^{*} + \frac{1-\alpha_U}{\alpha_U} a_P^{\mathbf{u}} \mathbf{u}_P^{*}$$

Under-Relaxation

Stability of the SIMPLE Iteration Sequence

- α_P and α_U are the pressure and velocity under-relaxation factors. Some guidelines for choosing under-relaxation are

$$0 < \alpha_P \leq 1$$

$$0 < \alpha_U \leq 1$$

$$\alpha_P + \alpha_U \approx 1$$

  or the standard set (guidance only!!!)

$$\alpha_P = 0.2$$

$$\alpha_U = 0.8$$

- Under-relaxation dampens the oscillation in the pressure-velocity coupling and is very efficient in stabilising the algorithm

PISO Algorithm

Pressure Correction Equation

- SIMPLE algorithm prescribes that the momentum predictor will be solved using the available pressure field. The role of pressure in the momentum equation is to ensure that the velocity field is divergence free
- After the first momentum solution, the velocity field is not divergence-free: we used a guessed pressure field
- Therefore, the pressure field after the first pressure corrector will contain two parts
  - Physical pressure, consistent with the global flow field
  - A “pressure correction” component, which enforces the continuity and counter-balances the error in the initial pressure guess

  Only the first component should be built into the physical pressure field
- In SIMPLE, this is handled by severely under-relaxing the pressure

PISO Algorithm

Under-Relaxation and PISO

- Having 2 under-relaxation coefficients which balance each other is very inconvenient: difficult tuning
- The idea of PISO is as follows:
  - Pressure-velocity system contains 2 complex coupling terms
    - Non-linear convection term, containing $\mathbf{u} - \mathbf{u}$ coupling
    - Linear pressure-velocity coupling
  - On low Co number (small time-step), the pressure velocity coupling is much stronger than the non-linear coupling
  - It is therefore possible to repeat a number of pressure correctors without updating the discretisation of the momentum equation (using the new fluxes)
  - In such a setup, the first pressure corrector will create a conservative velocity field, while the second and following will establish the pressure distribution
- Since multiple pressure correctors are used with a single momentum equation, it is no longer necessary to under-relax the pressure. In steady-state simulations, the system is stabilised by momentum under-relaxation
- On the negative side, derivation of PISO is based on the assumption that momentum discretisation may be safely frozen through a series of pressure correctors, which is true only at small time-steps

PISO Algorithm

Sequence of Operations:

- Use the available pressure field p^* from previous corrector or time-step. Conservative fluxes corresponding to p^* are also available
- Discretise the momentum equation with the available flux field
- Solve the momentum equation using the guessed pressure. This step is called **momentum predictor**

$$a_P^{\mathbf{u}} \mathbf{u}_P = \mathbf{H}(\mathbf{u}) - \nabla p^*$$

- Calculate the new pressure based on the velocity field. This is called a **pressure correction** step

$$\nabla \bullet \left[(a_P^{\mathbf{u}})^{-1} \nabla p \right] = \nabla \bullet ((a_P^{\mathbf{u}})^{-1} \mathbf{H}(\mathbf{u}))$$

- Based on the pressure solution, assemble conservative face flux F

$$F = \mathbf{s}_f \bullet \mathbf{H}(\mathbf{u}) - a_N^p (p_N - p_P)$$

PISO Algorithm

Sequence of Operations, cont'd:

- Explicitly update cell-centred velocity field with the assembled momentum coefficients

$$\mathbf{u}_P = (a_P^{\mathbf{u}})^{-1}(\mathbf{H}(\mathbf{u}) - \nabla p)$$

- Return to pressure correction step if convergence is not reached
- Proceed from step beginning for for a new time-step

PISO Algorithm

- PISO is useful in kinds of simulations where the time-step is controlled by external issues and temporal accuracy is important. In such cases, assumption of slow variation over non-linearity holds and the cost of momentum assembly and solution can be safely avoided. Example: Large Eddy simulation
- Functional equivalent of the PISO algorithm is alo used as a preconditioner in Krylov space saddle-point solvers

Accelerated Steady-State Solutions

Accelerated Steady-State Solutions

- In transient flows there is a natural time-step limit, related to mesh size
- In many cases local mesh resolution (eg. near walls) limits global time-step size to unreasonable level: need to violate the limit locally “without effect on global time accuracy”
- Options
  - **Transient SIMPLE-based algorithms**: consider each time-step as a quasi-steady solution, by assuming the time derivative term is a “local inertial source”
  - **Inertial under-relaxation**: in small cells the temporal accuracy will be ruined anyway: assume is already the case and add “inertial under-relaxation” in problematic cells
  - **Sub-cycling**: in cases where a large time-step or inertial under-relaxation is not an option, (eg. wave propagation), a transport equation can be solved in sub-steps to satisfy the Co number limit
- Note: non-uniform time step size will potentially lead to unboundedness problems. Local time-stepping algorithms can be reformulated to work around this problem.

Block-Coupled Algorithms

Block-Coupled Pressure-Velocity Algorithms

- It is possible to formulate a system of equations which will discretise both the pressure and momentum equation in a single matrix and solve them together: **pressure-based block-coupled solver**
- Efficiency of such algorithms is good for steady-state computations
- . . . but at a cost of considerable increase in storage and choice of linear solver technology
- Advantage of the pressure-based formulation is that it does not break down at the incompressibility limit, unlike the density-based block-coupled algorithms