051176 - Computational Techniques for Thermochemical Propulsion
Master of Science in Aeronautical Engineering

p-U Coupling in OpenFOAM

Prof. **Federico Piscaglia**
Dept. of Aerospace Science and Technology (DAER)
POLITECNICO DI MILANO, Italy
federico.piscaglia@polimi.it

Coupled Equations and their solution

Most problems in fluid dynamics and heat transfer require solution of coupled systems of equations, i.e. the dominant variable of each equation occurs in some of the other equations.

There are two types of approaches to such problems:

- **simultaneous methods/COUPLED SOLVERS**: all variables are solved for simultaneously;
- **sequential methods/SEGREGATED SOLVERS**: each equation is solved for its dominant variable, treating the other variables as known, and one iterates through the equations until the solution of the coupled system is obtained.

Sequential solution/segregated solvers

When the equations are linear and tightly coupled, the simultaneous approach is best. However, the equations may be so complex and non-linear that coupled methods are difficult and expensive to use. Solution:

- each equation is treated as if it has only a single unknown, temporarily treating the other variables as known, using the best currently available values for them;
- the equations are then solved in turn, repeating the cycle until all equations are satisfied.

When using this type of methods, you need to remind that:

- since some terms (the coefficients and source terms) that depend on the other variables change as the computation proceeds, **it is inefficient to solve the equations accurately at each iteration**. Iterations performed on each equation are called *inner iterations*.
- In order to obtain a solution which satisfies all of the equations, the coefficient matrices and source vector must be updated after each cycle and the process repeated. The cycles are called *outer iterations*.

Governing Equations

Solution of the Navier-Stokes (NS) equations is complicated by the lack of on independent equation for the pressure, whose gradient contributes to each of the three momentum equations:

Mass conservation

$$\frac{\partial \rho}{\partial t} + \nabla(\rho \boldsymbol{u}) = 0$$

Momentum conservation

$$\frac{\partial \left(\rho \boldsymbol{u}\right)}{\partial t} + \nabla \cdot (\rho \boldsymbol{u} \cdot \boldsymbol{u}) = -\nabla p + \frac{2}{3}\mu \nabla \boldsymbol{u} \boldsymbol{I} + \nabla \cdot [\mu(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^{T}] + \rho \boldsymbol{g} + \boldsymbol{S}$$

- Mass conservation is a kinematic constraint on the velocity field rather than a dynamic equation
- One way out of this difficulty is to construct the pressure field so as to guarantee satisfaction of the continuity equation.

p-U Coupling in Segregated Solvers

A very common way in CFD to solve *mass+momentum* is based on the by the so-called **segregated solvers**, where momentum and mass are solved sequentially.

In the following slides, **the basic philosophy behind some of the most popular methods of pressure-velocity coupling** **in SEGREGATED solvers** is described.

$\rightarrow$ The formulation for **COMPRESSIBLE** fluids ($\rho \neq$ const!) will be presented here.

p-U Coupling in Segregated Solvers

1) **Momentum equations determine the respective velocity components**:

$$\frac{\partial(\rho u)}{\partial t} = -\nabla \cdot (\rho \vec{u}\vec{u}) + \nabla \cdot \vec{\tau} - \nabla p = \vec{H} - \vec{\nabla} p$$

where:

a) $\vec{H} = -\nabla \cdot (\rho \vec{u}\vec{u}) + \nabla \cdot \vec{\tau}$ is the shorthand notation for the advective and viscous terms;

b) $\vec{\nabla} p$ is the discretized spatial derivative (gradient) of the pressure term.

If the velocity field is calculated from the momentum equation, pressure must be calculated by the continuity equation.

<u>But continuity does not contain pressure! How can this be done?</u>

$\rightarrow$ The most common method is based on **combining the two equations (continuity + momentum)**.

p-U Coupling in Segregated Solvers

2) If we take the **divergence of the momentum equation**:

$$\nabla \cdot \left[\frac{\partial (\rho \vec{u})}{\partial t} + \nabla \cdot (\rho \vec{u} \vec{u}) \right] = \nabla \cdot \left[-\nabla p + \nabla \cdot \vec{\tau} + \vec{S} \right]$$

There is a direct link between pressure and density though the equation of state (EOS)! The term $\nabla \cdot \frac{\partial(\rho \vec{u})}{\partial t}$ can be rewritten as a function of the pressure p, using the mass conservation $\left[\nabla \cdot \rho \boldsymbol{u} = -\frac{\partial \rho}{\partial t}\right]$ and the equation of state (EOS):

$$-\nabla \cdot \left(\frac{\partial \rho \boldsymbol{u}}{\partial t} \right) = -\frac{\partial}{\partial t} (\nabla \cdot \rho \boldsymbol{u}) =$$

$$= -\frac{\partial}{\partial t} \left(-\frac{\partial \rho}{\partial t} \right) = \frac{\partial^2 \rho}{\partial t^2} = \frac{\partial^2 (\psi \, p)}{\partial t^2}$$

so:

$$\frac{\partial^2 \rho}{\partial t^2} = \frac{\partial^2 (\psi \, p)}{\partial t^2}$$

NOTE: for perfect gases, $\psi = \frac{1}{RT}$.

p-U Coupling in Segregated Solvers

If we directly calculate the divergence of the momentum equation and we combine it with mass conservation, we obtain equations for pressure where the updated values of velocity at time n+1 appear both on the LHS and on the RHS of the NS equations.

$\rightarrow$ for this reason, the approach would be computationally not efficient.

An alternative way of dealing with the non-linearity is constructed by:

- replacing in the divergence of the momentum equation

$$\nabla \cdot (\rho \vec{u}\vec{u}) - \nabla \cdot \vec{\tau} \rightarrow \frac{\partial (\rho \vec{u})}{\partial t}$$

- linearizing the equations about the results at the preceding time-step:

$$u_i^{n+1} = u_i^n + \Delta u_i$$

In the following slides, we will discuss about how mass and momentum are solved by the Projection Method in segregated solvers.

The Projection Method in Segregated Solvers

The **Chorin's projection method** is a fractional-step method based on the Helmholtz decomposition of any vector field into a *solenoidal part* and an *irrotational* part:

"Any vector field $\boldsymbol{v}$ defined in a volume Ω admits the unique orthogonal decomposition:

$$\boldsymbol{v} = \boldsymbol{v}^* + \nabla\phi$$

being $\boldsymbol{v}^*$ a solenoidal field in the volume Ω with zero normal component on the boundary $\partial\Omega$ of Ω."

Typically, the algorithm consists of two stages:

1) in a first stage, an intermediate velocity that does not satisfy the incompressibility constraint is computed at each time step;

2) in the second, the pressure is used to project the intermediate velocity onto a space of divergence-free velocity field to get the next update of velocity and pressure.

Chorin's Projection Method: `step 1`

1) In Chorin's original version of the projection method, one first computes an intermediate velocity, $\mathbf{u}^*$, explicitly using the momentum equation by ignoring the pressure gradient term:

$$\left.\frac{\partial(\rho u_i)}{\partial t}\right|_n^* = \left(-\frac{\partial(\rho u_i u_j)^{n+1}}{\partial x_i} + \frac{\partial \tau_{ij}^{n+1}}{\partial x_j}\right)$$

Remember that the algorithm is really just an operator splitting approach in which one considers the viscous forces (in the first half step) and the pressure forces (in the second half step) **separately**.

Chorin's Projection Method: `step 2`

2) In the second half of the algorithm, the "projection step", the intermediate velocity is corrected to obtain the final solution of the time step $\mathbf{u}^{n+1}$:

$$\left.\frac{\partial\left(\rho u_i\right)}{\partial t}\right|_{*}^{n+1} = -\,\nabla p^{n+1}$$

Computing the right-hand side of the second half step requires knowledge of the pressure $p^{(n+1)}$. This is obtained in two steps:

a) the divergence of the projection step is calculated:

$$-\nabla\cdot\left[\frac{\left(\rho u_i\right)^{n+1}-\left(\rho u_i\right)^{*}}{\Delta t}\right] = \nabla\cdot\left[\nabla p^{n+1}\right]$$

b) the continuity equation $\left[\left.\frac{\partial\rho}{\partial t}+\nabla\cdot\rho\boldsymbol{u}\right|_{n+1} = 0\right]$ is used to derive the so-called **Helmholtz equation** for p^{n+1}:

$$\nabla^2 p^{n+1} = \frac{1}{\Delta t}\left[\nabla\cdot\rho^{*}\mathbf{u}^{*}+\frac{\partial\rho}{\partial t}\right]$$

Chorin's Method: HELMHOLTZ equation

The final form of the mass conservation for compressible flows is called Helmholtz equation (or **pressure equation**):

$$\nabla^2 \, p^{n+1} = \frac{1}{\Delta t} \left[\frac{\partial \left(\psi \cdot p \right)}{\partial t} + \nabla \cdot \left(\rho^* \mathbf{u}^* \right) \right]$$

with

$$\psi = \frac{1}{RT}$$

- a distinguishing feature of Chorin's projection method is that the velocity field is forced to satisfy a discrete continuity constraint at the end of each time step.
- the pressure equation is called Helmholtz equation for compressible flows or *Poisson* equation for the incompressible formulation.

Chorin's Method: POISSON equation

- For *incompressible fluids*, the Helmholtz equation is known as **Poisson's equation**. Being $\rho =$ const, the Helmholtz equation becomes:

$$\nabla^2 \, p^{n+1} = \frac{1}{\Delta t} \nabla \cdot (\rho \, \boldsymbol{u}^*)$$

- the original formulation of the Chorin's scheme is first-order accurate, since both time and spatial discretization is discretized by a first order scheme a first-order time splitting error affects this scheme.

In OpenFOAM, the old time step pressure is used in the first step and second order discretization for temporal derivatives can be chosen: as a consequence, a pressure correction scheme with second-order time-splitting error is obtained.

A note on incompressible flows

IMPORTANT NOTE: For incompressible flows the static pressure can also be prescribed on either the in- or outflow boundary. As the mass flux is a function of the difference in pressure between the inflow and outflow, the velocity at the inflow boundary cannot be prescribed if the pressure is prescribed at both in- and outflow boundaries. **REMEMBER**, for incompressible flows:

$$\nabla^2 \, \frac{p^{n+1}}{\rho} = \frac{1}{\Delta t} \nabla \cdot (\boldsymbol{u}^*)$$

Example: the PISO algorithm

PISO algorithm for unsteady incompressible flows

$$\begin{tikzcd}
\text{Start simulation} \arrow[d] \\
\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot (\mathbf{U}\mathbf{U}) + \nabla \cdot \mathbf{R} = -\nabla p \arrow[d] \\
\nabla^2 p = f(\mathbf{U}, \nabla p) \arrow[d] \arrow[loop right, "\text{New } p" ', "\texttt{nNonOrthogonalCorrectors}"] \\
\mathbf{U} = f(\nabla p) \arrow[d] \arrow[u, bend right=60, swap, "\texttt{nCorrectors=2}"] \arrow[uu, bend right=80] \\
\text{End simulation}
\end{tikzcd}$$

Chorin’s Projection Method in OpenFOAM

Almost ANY solver in OpenFOAM makes use of the projection method to solve the pressure-velocity coupling, which is included in the files `UEqn.H` and `pEqn.H` in the solver’s folder.

- **PREDICTOR STEP** – UEqn.H:
  a) get velocity field components u^* and v^* $\rightarrow$ momentum equation
  b) EVENTUALLY adjust the predicted velocity $\rightarrow$ OpenFOAM, momentum predictor

- **CORRECTOR STEP** – pEqn.H:
  c) enforce continuity $\rightarrow$ Poisson equation
  d) ... and correct the predicted velocity

OpenFOAM: `predictor step 1/2`

PREDICTOR STEP

- Guess the pressure field p^* and get velocity field components u^* and v^* using discretized momentum equation.

```
tmp<fvVectorMatrix> tUEqn
(
    fvm::ddt(rho, U) + fvm::div(phi, U)
  + turbulence->divDevRhoReff(U)
 ==
    fvOptions(rho, U)
);

fvVectorMatrix& UEqn = tUEqn.ref();

UEqn.relax();
```

OpenFOAM: `predictor step 2/2`

PREDICTOR STEP

a) momentum **may be** solved a second time by inserting correct pressure field p^{**}, to get the corresponding correct velocity components u^{**} and v^{**}.

```
if (pimple.momentumPredictor())
{
     solve(UEqn == -fvc::grad(p));
     ...
}
```

NOTE: this is a deviation from the original theory of the Chorin Projection Method.

OpenFOAM: corrector step

CORRECTOR STEP (pEqn.H)

b) enforce continuity:

```
fvScalarMatrix pDDtEqn
(
    fvc::ddt(rho) + psi*correction(fvm::ddt(p))
    + fvc::div(phiHbyA)
    ==
    fvOptions(psi, p, rho.name())
);

while (pimple.correctNonOrthogonal())
{
      fvScalarMatrix pEqn(pDDtEqn - fvm::laplacian(rhorAUf, p));
      ....
}
```

c) ... and correct the predicted velocity u^*:

```
U = HbyA - rAU*fvc::grad(p);
```

Note: at the last iteration, the corrected velocity returns u^{n+1} (= it is the solution!)

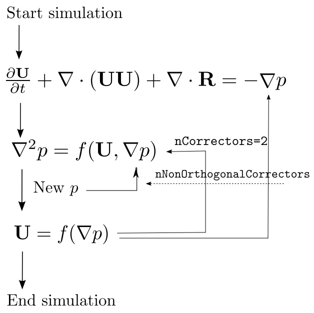
Density update

A missing note: **how ρ^* is calculated in the pressure equation**?

In compressible solvers (with varying density!), the equation of energy must be calculated between momentum and pressure AND the density is calculated/updated by the equation of state, defined in the thermophysical model.

Linearization of Non-linear Terms

If you look at the implementation of ANY solver, you will find that the segregated solution of the governing equations requires an **ITERATIVE SOLUTION**.


Start simulation
$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot (\mathbf{U}\mathbf{U}) + \nabla \cdot \mathbf{R} = -\nabla p$
$\nabla^2 p = f(\mathbf{U}, \nabla p)$
nCorrectors=2
nNonOrthogonalCorrectors
New p
$\mathbf{U} = f(\nabla p)$
End simulation

Solution algorithm in pisoFOAM

Why an iterative solution is needed? Nonlinear terms (convective flux, source term) in the coupled non-linear equations are solved by a sequential decoupled method and convective and source sink terms are usually linearized. This kind of linearization and the discretization methods commonly employed in CFD codes require to iterate and under-relax variables to achieve the solution of the equations.

Picard Approach for Convective Terms

The convective term in the momentum equation is non-linear.

$$\frac{\partial(\rho u_i u_j)}{\partial x_j} \qquad \text{and} \qquad \int_S \rho u_i \boldsymbol{v} \cdot \boldsymbol{n} \cdot \boldsymbol{n} dS$$

Nonlinear terms (convective fluxes, source terms) in the coupled non-linear equations are solved by a sequential decoupled method and **are usually linearized using Picard iteration approach**. A typical example is given by the convective terms in the momentum equation that appears in OpenFOAM as:

```
fvm::div(phi,U)
```

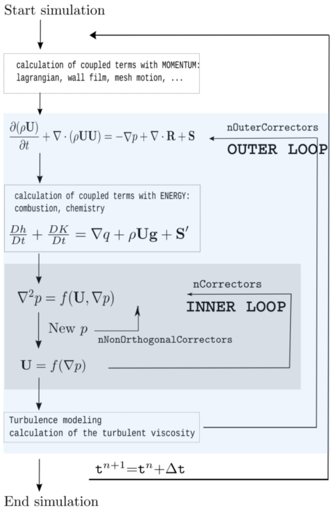
begin `phi` = $(\rho u_j)^*$ and `U` = u_i. The mass flux is treated as known, so the non-linear convective term in the equation for the u_i momentum component is approximated by:

$$\rho u_j u_i \simeq (\rho u_j)^* \, u_i$$

where index $*$ denotes that the values are taken from the result of the previous outer iteration.

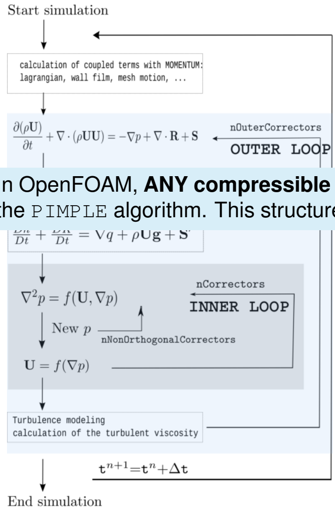
This kind of linearization, together with the discretization methods commonly employed in CFD codes, requires to iterate and under-relax variables to achieve the solution of the equations.

Sequential Decoupled Method (`rhoPimpleFoam`-based solvers)


Start simulation
calculation of coupled terms with MOMENTUM: lagrangian, wall film, mesh motion, ...
$\frac{\partial(\rho\mathbf{U})}{\partial t}+\nabla\cdot(\rho\mathbf{U}\mathbf{U})=-\nabla p+\nabla\cdot\mathbf{R}+\mathbf{S}$
nOuterCorrectors
OUTER LOOP
calculation of coupled terms with ENERGY: combustion, chemistry
$\frac{Dh}{Dt}+\frac{DK}{Dt}=\nabla q+\rho\mathbf{U}\mathbf{g}+\mathbf{S}'$
$\nabla^2 p=f(\mathbf{U},\nabla p)$
nCorrectors
INNER LOOP
New p
nNonOrthogonalCorrectors
$\mathbf{U}=f(\nabla p)$
Turbulence modeling calculation of the turbulent viscosity
$t^{n+1}=t^n+\Delta t$
End simulation

- SIMPLE (steady) :
  - `nOuterCorrectors`>1
  - `nCorrectors`=1
  - `nNonOrthogonalCorrectors` ∈ [0;∞)
- PISO (unsteady):
  - `nOuterCorrectors`=1
  - `nCorrectors`=2
  - `nNonOrthogonalCorrectors` ∈ [0;∞)
- PIMPLE (unsteady, transient SIMPLE):
  - `nOuterCorrectors`>1
  - `nCorrectors`>1
  - `nNonOrthogonalCorrectors` ∈ [0;∞)

Sequential Decoupled Method (`rhoPimpleFoam`-based solvers)

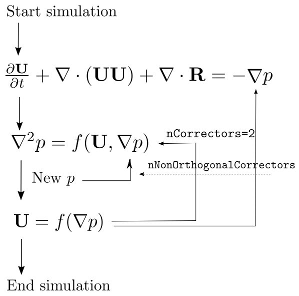

Start simulation
calculation of coupled terms with MOMENTUM: lagrangian, wall film, mesh motion, ...
$\frac{\partial(\rho\mathbf{U})}{\partial t}+\nabla\cdot(\rho\mathbf{U}\mathbf{U})=-\nabla p+\nabla\cdot\mathbf{R}+\mathbf{S}$
nOuterCorrectors
OUTER LOOP
$\frac{Dh}{Dt}+\frac{DK}{Dt}=\nabla q+\rho\mathbf{U}\mathbf{g}+\mathbf{S}$
nCorrectors
INNER LOOP
$\nabla^2 p=f(\mathbf{U},\nabla p)$
New p
nNonOrthogonalCorrectors
$\mathbf{U}=f(\nabla p)$
Turbulence modeling calculation of the turbulent viscosity
$t^{n+1}=t^{n}+\Delta t$
End simulation

- SIMPLE (steady) :
  - `nOuterCorrectors`>1
  - `nCorrectors`=1
  - `nNonOrthogonalCorrectors` ∈ [0;∞)

In OpenFOAM, **ANY compressible flow solver** is usually (with very few exceptions!) based on the `PIMPLE` algorithm. This structure is replicated in any PISO/SIMPLE/PIMPLE-Foam solver!

  - `nOuterCorrectors`=1
  - `nCorrectors`=2
  - `nNonOrthogonalCorrectors` ∈ [0;∞)
- PIMPLE (unsteady, transient SIMPLE):
  - `nOuterCorrectors`>1
  - `nCorrectors`>1
  - `nNonOrthogonalCorrectors` ∈ [0;∞)

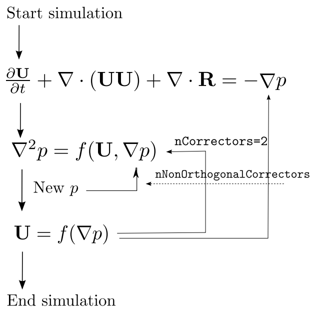
Remember...


Start simulation
$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot (\mathbf{U}\mathbf{U}) + \nabla \cdot \mathbf{R} = -\nabla p$
$\nabla^2 p = f(\mathbf{U}, \nabla p)$
nCorrectors=2
nNonOrthogonalCorrectors
New p
$\mathbf{U} = f(\nabla p)$
End simulation

`Solution algorithm in pisoFOAM`

In this moment, **we are looking at the needed iterations to achieve the sequential solution of the equation of a governing equation (inner and outer iteration of the solver)**. For each of these iterations, the linearized equation (i.e. linear system) is solved with iterative (preconditioned) methods!

Remember...


Start simulation
$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot (\mathbf{U}\mathbf{U}) + \nabla \cdot \mathbf{R} = -\nabla p$
$\nabla^2 p = f(\mathbf{U}, \nabla p)$
nCorrectors=2
nNonOrthogonalCorrectors
New p
$\mathbf{U} = f(\nabla p)$
End simulation

```
Time = 0.01

Courant Number mean:  0.0976825 max:  0.585607
smoothSolver:  Solving for Ux, Initial residual = 0.160686, Final residual = 6.83031e-06, No Iterations 19
smoothSolver:  Solving for Uy, Initial residual = 0.260828, Final residual = 9.65939e-06, No Iterations 18
DICPCG: Solving for p, Initial residual = 0.428925, Final residual = 0.0103739, No Iterations 22
time step continuity errors :  sum local = 0.000110788, global = 3.77194e-19, cumulative = -6.72498e-20
DICPCG: Solving for p, Initial residual = 0.30209, Final residual = 5.26569e-07, No Iterations 33
time step continuity errors :  sum local = 6.61987e-09, global = -2.74872e-19, cumulative = -3.42122e-19
ExecutionTime = 0.01 s ClockTime = 0 s
```

Solution algorithm in pisoFOAM

UNDER-RELAXATION OF VARIABLES IN THE SEQUENTIAL SOLUTION

Under-Relaxation in the Sequential Solution

The optimization of the segregated solution method requires careful choice of the number of inner iterations per outer iteration.

$$\begin{tikzcd}
\text{Start simulation} \arrow[d] \\
\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot (\mathbf{U}\mathbf{U}) + \nabla \cdot \mathbf{R} = -\nabla p \arrow[d] \\
\nabla^2 p = f(\mathbf{U}, \nabla p) \arrow[d, "\text{New } p"] \arrow[loop right, dashed, "\texttt{nNonOrthogonalCorrectors}"] \\
\mathbf{U} = f(\nabla p) \arrow[d] \arrow[u, bend right, "\texttt{nCorrectors=2}"'] \arrow[uu, bend right=60] \\
\text{End simulation}
\end{tikzcd}$$

It is necessary to **limit the change in each variable from one outer iteration to the next** (*under-relaxation*), because a change in one variable changes the coefficients in the other equations, which may slow or prevent convergence!

Unfortunately, it is hard to analyze the convergence of these methods so the selection of under-relaxation factors is largely empirical.

Under-Relaxation in the Sequential Solution

The **conservation equation for a general scalar variable** ϕ can be expressed as:

$$\underbrace{\underbrace{\frac{\partial(\rho\phi)}{\partial t}}_{\texttt{temporal derivative}} + \underbrace{\nabla\cdot\rho(\boldsymbol{u}-\boldsymbol{u_b})\phi}_{\texttt{convection term}}}_{\frac{D\phi}{Dt}} = \underbrace{\nabla\cdot\left(\Gamma^{\phi}\nabla\phi\right)}_{\texttt{diffusion term}} + \underbrace{Q^{\phi}}_{\texttt{source term}}$$

We shall present one under-relaxation technique that is widely used to solve it.

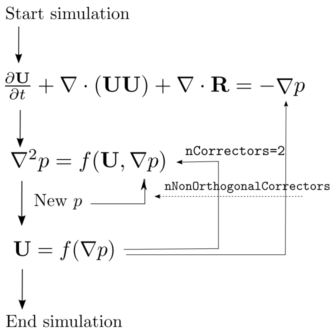
Under-Relaxation Technique

On the n^{th} outer iteration, the algebraic equation for $\phi^{(n)}$ at a typical point P may be written in the discretized form as:

$$A_P \phi_P^{(n)} + \sum_l A_I \phi_I^{(n)} = Q_P \qquad (1)$$

where:

- Q contains all the terms that do not depend explicitly on $\phi^{(n)}$;
- the coefficients A_l and the source Q may involve $\phi^{(n-1)}$


Start simulation
$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot (\mathbf{UU}) + \nabla \cdot \mathbf{R} = -\nabla p$
$\nabla^2 p = f(\mathbf{U}, \nabla p)$
nCorrectors=2
nNonOrthogonalCorrectors
New p
$\mathbf{U} = f(\nabla p)$
End simulation

SOME NOTES:

- The discretization scheme is unimportant here;
- Eq. (1) is linear and the system of equations for the whole solution domain is solved usually iteratively (inner iterations).

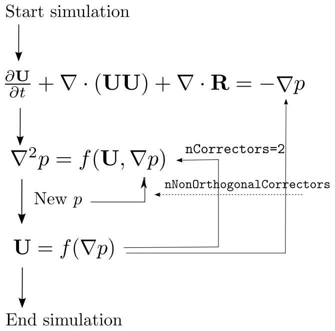
Under-Relaxation Technique

On the n^{th} outer iteration, the algebraic equation for $\phi^{(n)}$ at a typical point P may be written in the discretized form as:

$$A_P \phi_P^{(n)} + \sum_l A_I \phi_I^{(n)} = Q_P \qquad (1)$$

where:

- Q contains all the terms that do not depend explicitly on $\phi^{(n)}$;
- the coefficients A_l and the source Q may involve $\phi^{(n-1)}$



Start simulation
$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot (\mathbf{UU}) + \nabla \cdot \mathbf{R} = -\nabla p$
$\nabla^2 p = f(\mathbf{U}, \nabla p)$
nCorrectors=2
New p
nNonOrthogonalCorrectors
$\mathbf{U} = f(\nabla p)$
End simulation

In the early outer iterations, allowing ϕ to change by as much as Eq. (1) requires could cause instability, so we allow ϕ^n to change only a fraction α_ϕ of the would-be difference:

$$\phi_P^n = \phi_P^{n-1} + \alpha_\phi \left(\phi^{\text{new}} - \phi^{n-1}\right) \qquad (2)$$

where ϕ^{new} is the result of Eq. (1) and the under-relaxation factor satisfies $0 < \alpha_\phi < 1$.

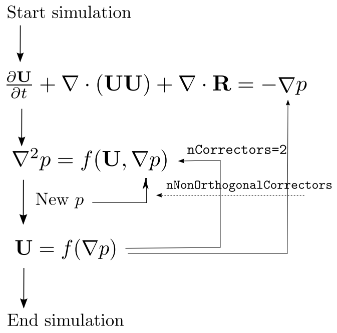
Under-Relaxation Technique

Since the old iterate is usually no longer required after the coefficient matrix and source vector are updated, the new solution can be written over it. Replacing ϕ^{new} in Eq. (2) by:

$$\phi^{\text{new}} = \frac{Q_P - \sum_l A_l \phi_l^n}{A_P} \tag{3}$$

from which follows:

$$\underbrace{\frac{A_P}{\alpha_\phi}}_{A_P^*} \phi_P^n + \sum_l A_l \phi_l^n = \underbrace{Q_P + \frac{1-\alpha_\phi}{\alpha_\phi} A_P \phi_P^{n-1}}_{Q_P^*} \tag{4}$$



Start simulation
$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot (\mathbf{U}\mathbf{U}) + \nabla \cdot \mathbf{R} = -\nabla p$
$\nabla^2 p = f(\mathbf{U}, \nabla p)$
nCorrectors=2
nNonOrthogonalCorrectors
New p
$\mathbf{U} = f(\nabla p)$
End simulation

where A_P^* and Q_P^* are modified main diagonal matrix elements and source vector components. This modified equation is solved within inner iterations.

When the outer iterations converge, the terms involving α_ϕ cancel out and we obtain the solution of the original problem.

Under-Relaxation Technique

How do we get Eq. (4)?

If we substitute Eq. (3) in Eq. (2), it follows:

$$\phi_P^n = \phi_P^{n-1} + \alpha_\phi \left(\frac{Q_P - \sum_l A_l \phi_l^n}{A_P} - \phi^{n-1} \right)$$

By multiplying by $\frac{A_P}{\alpha_\phi}$, the above equation is re-arranged to:

$$\begin{aligned} \frac{A_P}{\alpha_\phi} \phi_P^n + \sum_l A_l \phi_l^n &= \frac{A_P}{\alpha_\phi} \phi_P^{n-1} + Q_P - \alpha_\phi \phi_P^{n-1} \\ &= Q_P + A_P \left(\frac{1}{\alpha_\phi} - 1 \right) \phi_P^{n-1} \\ &= Q_P + \frac{A_P}{\alpha_\phi} \left(1 - \alpha_\phi \right) \phi_P^{n-1} \end{aligned}$$

and finally:

$$\underbrace{\frac{A_P}{\alpha_\phi}}_{A_P^*} \phi_P^n + \sum_l A_l \phi_l^n = \underbrace{Q_P + \frac{1-\alpha_\phi}{\alpha_\phi} A_P \phi_P^{n-1}}_{Q_P^*}$$

Under-Relaxation Technique

$$\phi^{(n)} = \phi^{(n-1)} + \alpha_\phi(\phi^{\text{new}} - \phi^{(n-1)})$$

This kind of under-relaxation was proposed by **Patankar (1980).**

- It has a positive effect on many iterative solution methods since the diagonal dominance of the matrix A is increased (the element A^*_P is larger than A_p, while A_l remains the same). It is more efficient than explicit application of the expression (3).
- Optimum under-relaxation factors are problem dependent. A good strategy is to use a small under-relaxation factor in the early iterations and increase it towards unity as convergence is approached.
- Under-Relaxation may be applied not only to the dependent variables but also to individual terms in the equations. It is often necessary to do so when the fluid properties (viscosity, density, Prandtl number etc.) depend on the solution and need be updated.

Iterative solution methods can often be regarded as solving an unsteady problem until a steady state is reached. Control of the time step is then important in controlling the evolution of the solution. In this sense, time step may be interpreted as an under-relaxation factor! The under-relaxation scheme described above may be interpreted as using different time steps at different nodes.

Under-Relaxation in OpenFOAM

Where is under-relaxation set in OpenFOAM?

```
relaxationFactors
{
    fields
    {
        p                0.3;
    }
    equations
    {
        U                0.7;
        ``(k|omega|epsilon).*'' 0.7;
    }
}

// ************************************************************************* //
```

In the `fvSolution` dictionary in the `system` directory of the simulation folder:

- **relaxation factors for under-relaxation of fields** are specified within a `field` sub-dictionary;
- **relaxation factors for equation under-relaxation** are within an `equations` sub-dictionary. The factors are specified for pressure p, pressure U, and turbulent fields grouped using a regular expression.

As apparent in the `fvSolution` file, under-relaxation can be applied both to `fields` (i.e. variables) and `equations` (i.e. matrices).

Under-Relaxation in OpenFOAM

Where can I find this in OpenFOAM? You will find this at the `solver`-level, i.e. in the solvers available in the subfolders of `$FOAM_SOLVERS`, usually in `UEqn.H`:

```
tmp<fvVectorMatrix> tUEqn
(
    fvm::ddt(rho, U) + fvm::div(phi, U)
    + MRF.DDt(rho, U)
    + turbulence->divDevRhoReff(U)
    ==
    fvOptions(rho, U)
);
fvVectorMatrix& UEqn = tUEqn.ref();

UEqn.relax();
```

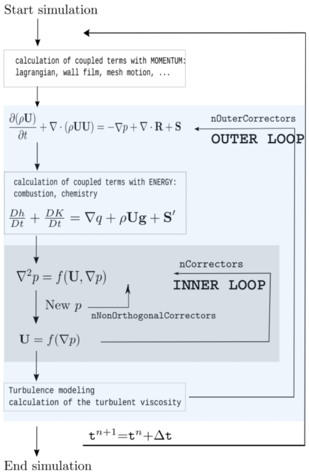
In the function `relax()`, defined at line 523 of file `fvMatrix.C` (see OpenFOAM-7), the **relaxed coefficient matrix** is calculated, together with the **the relaxation contribution to the source**:

$$A_P^* = \frac{A_P}{\alpha_\phi}$$

$$Q_P^* = Q_P + \frac{1-\alpha_\phi}{\alpha_\phi} A_P \, Q_P^{n-1}$$

Under-Relaxation in OpenFOAM

As just said, iterative solution methods can often be regarded as solving an unsteady problem until a steady state is reached. **Now please have a look again at the OpenFOAM solver and think..**



Start simulation
calculation of coupled terms with MOMENTUM: lagrangian, wall film, mesh motion, ...
$\frac{\partial(\rho\mathbf{U})}{\partial t}+\nabla\cdot(\rho\mathbf{U}\mathbf{U})=-\nabla p+\nabla\cdot\mathbf{R}+\mathbf{S}$
nOuterCorrectors
OUTER LOOP
calculation of coupled terms with ENERGY: combustion, chemistry
$\frac{Dh}{Dt}+\frac{DK}{Dt}=\nabla q+\rho\mathbf{U}\mathbf{g}+\mathbf{S}'$
nCorrectors
INNER LOOP
$\nabla^2 p=f(\mathbf{U},\nabla p)$
New p
nNonOrthogonalCorrectors
$\mathbf{U}=f(\nabla p)$
Turbulence modeling calculation of the turbulent viscosity
$t^{n+1}=t^n+\Delta t$
End simulation

You should now understand why in `simpleFoam` the entry `timeStep` in `system/fvSolution` represents the **global** number of iterations of the SIMPLE solver!

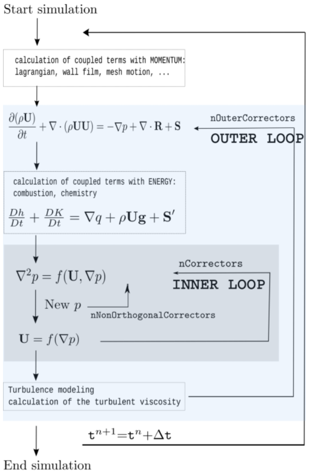
Under-Relaxation in OpenFOAM

Under-Relaxation may be applied not only to the dependent variables but also to individual terms in the equations. It is often necessary to do so when the fluid properties (viscosity, density, Prandtl number etc.) depend on the solution and need be updated.
In `$FOAM_CASE/system/fvSolution` you will find:

```
relaxationFactors
{
    fields
    {
        rho                1;
        p_rgh              0.7;
    }
    equations
    {
        U                  0.3;
        "(h|e)"            0.3;
        k                  0.3;
        epsilon            0.3;
    }
}
```

CONVERGENCE IN THE STEADY-STATE SOLUTION (SIMPLE)

Methods to Improve Convergence (SIMPLE)


Start simulation
calculation of coupled terms with MOMENTUM: lagrangian, wall film, mesh motion, ...
$\frac{\partial(\rho \mathbf{U})}{\partial t} + \nabla \cdot (\rho \mathbf{U}\mathbf{U}) = -\nabla p + \nabla \cdot \mathbf{R} + \mathbf{S}$
nOuterCorrectors
OUTER LOOP
calculation of coupled terms with ENERGY: combustion, chemistry
$\frac{Dh}{Dt} + \frac{DK}{Dt} = \nabla q + \rho \mathbf{U}\mathbf{g} + \mathbf{S}'$
$\nabla^2 p = f(\mathbf{U}, \nabla p)$
nCorrectors
INNER LOOP
New p
nNonOrthogonalCorrectors
$\mathbf{U} = f(\nabla p)$
Turbulence modeling calculation of the turbulent viscosity
$t^{n+1} = t^n + \Delta t$
End simulation

Boundedness of the convective derivative:

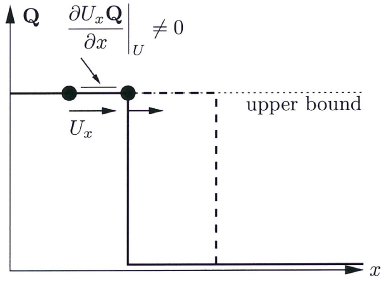
- Transport equations involve a material time derivative D/Dt
- This can be expressed in terms of a spatial time derivative and convection.
  For incompressible flows:

$$\frac{D\boldsymbol{Q}}{Dt} = \frac{\partial \boldsymbol{Q}}{\partial t} + \boldsymbol{U} \cdot \nabla \boldsymbol{Q} = \frac{\partial \boldsymbol{Q}}{\partial t} + \boldsymbol{U} \cdot \nabla \boldsymbol{Q} - (\nabla \cdot \boldsymbol{U})\boldsymbol{Q}$$

- the term $-(\nabla \cdot \boldsymbol{U})\boldsymbol{Q}$ could be theorically omitted, since $(\nabla \cdot \boldsymbol{U}) = 0$ (for incompressible flows)
- ... but since we solve numerically $(\nabla \cdot \boldsymbol{U}) \neq 0$

Tt is better to consider $(\nabla \cdot \boldsymbol{U}) \neq 0$ while we are approaching the convergent solution; this is valid especially in steady-state solvers where the aim is to iterate towards $(\nabla \cdot \boldsymbol{U}) = 0$

Boundedness of $\nabla \cdot (UQ)$ - Steady Solvers

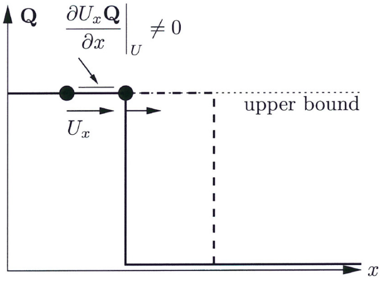

Q
$\left.\frac{\partial U_x \mathbf{Q}}{\partial x}\right|_U \neq 0$
upper bound
U_x
x

If we consider advection when U_x is inside the gradient of $\boldsymbol{Q}$:

$$\frac{\partial \boldsymbol{Q}}{\partial t} = -\frac{\partial (U_x \boldsymbol{Q})}{\partial x} \quad \text{equivalent to} \quad \frac{\partial \boldsymbol{Q}}{\partial t} + \nabla \cdot (\boldsymbol{U}\boldsymbol{Q}) = 0$$

- Upwind (U) is unbounded if $\partial U_x / \partial x < 0$, i.e. if $\nabla \cdot \boldsymbol{U} \neq 0$
- only $\boldsymbol{U} \cdot \nabla \boldsymbol{Q}$ is formally bounded with upwind
- including $-(\nabla \cdot \boldsymbol{U})\boldsymbol{Q}$ term improves boundedness when $\nabla \cdot \boldsymbol{U} \neq 0$

Boundedness of $\nabla \cdot (UQ)$ - Steady Solvers


$\mathbf{Q}$
$\left.\frac{\partial U_x \mathbf{Q}}{\partial x}\right|_U \neq 0$
upper bound
U_x
x

Example:

```
divSchemes
{
     div(div(phi,U)) bounded Gauss linearUpwind grad(U);
     div(phi,k)      bounded Gauss upwind;
     div(phi,omega)  bounded Gauss upwind;
}
```

SOME NOTES ON COMPRESSIBLE FLOWS

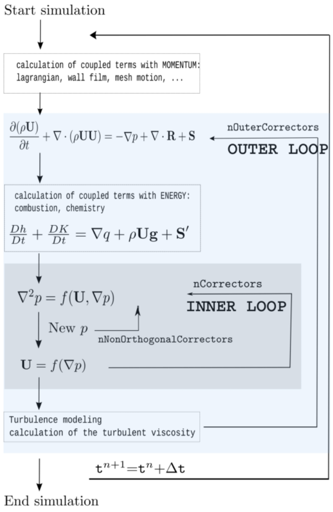
Compressible Flows Computation

While incompressible flow solutions do not generally require solving the energy equation, **compressibility effects couple hydrodynamics and thermodynamics**, necessitating the simultaneous solution of the continuity, momentum, and energy equations.

As a result:

1) the **dependence of density on pressure and temperature**, a relation expressed via an equation of state, further **complicates the velocity-pressure coupling** present in incompressible flows;
2) the derivation of the **pressure correction equation now involves a density correction** that introduces to the equation a convection-like term, in addition to the diffusion-like term introduced by the velocity correction;
3) complexity of **boundary conditions** arises in compressible flow problems.

rhoPimpleFoam



Start simulation
calculation of coupled terms with MOMENTUM: lagrangian, wall film, mesh motion, ...
$\frac{\partial(\rho\mathbf{U})}{\partial t} + \nabla\cdot(\rho\mathbf{U}\mathbf{U}) = -\nabla p + \nabla\cdot\mathbf{R} + \mathbf{S}$
nOuterCorrectors
OUTER LOOP
calculation of coupled terms with ENERGY: combustion, chemistry
$\frac{Dh}{Dt} + \frac{DK}{Dt} = \nabla q + \rho\mathbf{U}\mathbf{g} + \mathbf{S}'$
nCorrectors
INNER LOOP
$\nabla^2 p = f(\mathbf{U}, \nabla p)$
New p
nNonOrthogonalCorrectors
$\mathbf{U} = f(\nabla p)$
Turbulence modeling
calculation of the turbulent viscosity
$t^{n+1} = t^n + \Delta t$
End simulation

The discretization of the compressible momentum equations is essentially identical to that employed for the incompressible equations:

- to obtain the solution at the new time level, several outer iterations are performed;

- if time step is small, only a few outer iterations per time step are necessary. For steady problems, the time step may be infinite and the under-relaxation parameter acts like a pseudo-time step.

Compressible Flows Computation

The role of ENERGY

To compute compressible flows, it is necessary to solve not only the continuity and momentum equations but also a conservation equation for the thermal energy (or one for the total energy) and an equation of state. The latter is a thermodynamic relation connecting the density, temperature, and pressure:

$$\rho = \psi \, p$$

being ψ the flow compressibility (for perfect gases, $\psi = \frac{1}{RT}$).

- For incompressible flows, the energy equation reduces to a scalar transport equation for the temperature and only the convection and heat conduction are important.
- In compressible flows, viscous dissipation may be a significant heat source and conversion of internal energy to kinetic energy (and vice versa) by means of flow dilatation is also important.

Compressible Flows Computation

SIMPLE-family of methods for Compressible Flows

Computational Fluid Dynamics methods have been traditionally classified into two families denoted by density-based solvers (simulation of transonic and supersonic flows, aeronautics industry) and pressure-based solvers (e.g. SIMPLE algorithm).

- Patankar provided a clear resolution to this problem, and allowed for `SIMPLE`-based methods to genuinely develop into methods capable of resolving fluid flow at all speeds.
- **The critical development was the reformulation of the pressure equation to include density and velocity correction** such that the type of the equation changed from purely elliptic for incompressible flows to hyperbolic in transonic and supersonic compressible flows.

This allowed the SIMPLE-family of methods to seamlessly solve flow problems across the entire Mach number spectrum, with pressure playing the dual role of affecting density in the limit of high Mach compressible flow and velocity in the limit of incompressible flow, in order to enforce mass conservation.

Pressure-based Compressible Solvers

An **important advantage** of the pressure-based approach is its ability to **resolve fluid flows in the various Mach number regimes without any artificial treatment to promote convergence and stabilize computations**. This ability of the pressure based method is due to the dual role the pressure plays in compressible flows, which can best be described by considering the following two extreme cases:

- At very low Mach numbers, the pressure gradient needed to establish the flow field through momentum conservation is so small that it does not significantly influence the density, and the flow can be considered to be incompressible. Hence, density and pressure in addition to density and velocity are very weakly related indicating that variations in density are not sensitive to variations in velocity. In this case the continuity equation can no longer be considered as an equation for density, rather, it acts as a constraint on the velocity field.
- At hypersonic speeds changes in velocity become relatively small as compared to the velocity magnitude, indicating that variations in pressure do significantly affect density.

Pressure Correction

Despite the similarity in appearance to the pressure-correction equation for incompressible flows, there are important differences:

- the incompressible equation is a discretized Poisson equation, i.e. the coefficients represent an approximation to the Laplacian operator:

$$\nabla^2 \, p^{n+1} = \frac{1}{\Delta t} \nabla \cdot (\rho \, \boldsymbol{u}^*)$$

- In the compressible case, there are contributions that represent the fact that the equation for the pressure in a compressible flow contains convective and unsteady terms, i.e. it is actually a convected wave equation.

$$\nabla^2 \, p^{n+1} = \frac{1}{\Delta t} \left[\frac{\partial \left(\psi \cdot p \right)}{\partial t} + \nabla \cdot (\rho^* \mathbf{u}^*) \right]$$

Pressure-based Compressible Solvers

The dual role of PRESSURE

The above limiting cases highlight the dual role played by pressure in compressible flow situations. It clearly shows that pressure has a dual role:

- compressible flows: pressure affects density alone through the equation of state to satisfy mass conservation;
- incompressible flows: pressure acts on the velocity field via the gradient in the momentum equation to enforce mass conservation.

This dual role explains the success of the pressure-based approach to predict fluid flow at all speeds. This fact however did not deter workers in the density based track from using the artificial compressibility technique to develop methods capable of solving fluid flow at all speeds.

Pressure-Velocity-Density Coupling

Segregated solution method for compressible flows

Also for compressilble flows, we consider the segregated solution method, in which the linearized (around values from the previous outer iteration) equations for velocity components, pressure correction, temperature and other scalar variables are solved in turn.

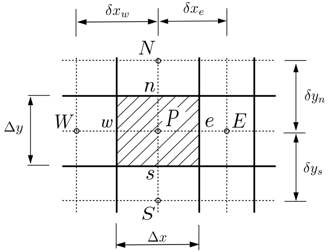
The discretized momentum equation for the velocity component u_i at the m_{th} outer iteration is essentially identical to that employed for the incompressible equations. While solving for one variable, other variables are treated as known. The momentum equation may be written as:

$$u_{i,P}^{m^*} = \frac{Q_{u_i}^{m-1} - \sum_l A_l^{u_i} u_{i,l}^{m^*}}{A_P^{u_i}} - \frac{\Delta V}{A_P^{u_i}} \left(\frac{\delta p^{m-1}}{\delta x_i} \right)_P$$

Here $Q_{u_i}^{m-1}$ represents the source term minus the contribution of the pressure term; in what follows, the discretization methods applied to this term and the pressure term are not important.

NOTE: the velocities obtained by solving linearized momentum equations and using 'old' pressure and density do not satisfy the mass conservation equations; they are denoted by an *asterisk*.

Pressure-Velocity-Density Coupling



When the mass fluxes m^* computed from these velocities and the *old* density:

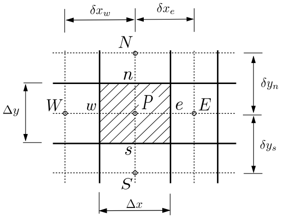
$$\dot{m}_i^* = \int_S (\rho \boldsymbol{v})^{m-1} \cdot \boldsymbol{n} dS$$

are inserted into the discretized continuity equation:

$$\frac{\rho^{m-1} - \rho^m}{\Delta t} \Delta V + \dot{m}_e^* + \dot{m}_w^* + \dot{m}_n^* + \dot{m}_s^* = Q_m^*$$

There results an imbalance Q_m^* that must be eliminated by a **correction method**.

Correction Method



$$\frac{\rho^{m-1}-\rho^{m}}{\Delta t}\Delta V+\dot{m}_e^*+\dot{m}_w^*+\dot{m}_n^*+\dot{m}_s^*=Q_m^*$$

A note about the correction method on Q_m^* to ensure mass conservation. For incompressible flows, the mass flux and the velocity are essentially equivalent and the imbalance is corrected by correcting the velocity (density is constant!):

$$\dot{m}_i^*=\int_S(\rho\boldsymbol{v})^{m-1}\cdot\boldsymbol{n}dS=\rho\int_S(\boldsymbol{v})^{m-1}\cdot\boldsymbol{n}dS=\rho\sum_i v_iS_i \qquad i=e,w,n,s$$

Since velocity correction is proportional to the gradient of the pressure correction:

$$u_{i,P}^{m}=\tilde{u}_{i,P}^{m^*}-\frac{1}{A_P^{u_i}}\left(\frac{\delta p^m}{\delta x_i}\right)_P$$

then an equation for the pressure correction can be derived and solved.

This procedure is not applicable in the compressible case.

Prescribed mass flux through a boundary

This leads to the pressure-correction equation having Neumann conditions on all boundaries and makes it singular. To make the solution unique, one usually takes the pressure at one point to be fixed, so the pressure correction calculated at that point is subtracted from all the corrected pressures. Another choice is to set the mean pressure to some value, say zero.

When you run an incompressible flo
`Solution` file, you will find:

```
relaxationFactors
{
    fields
    {
        rho                 1;
        p_rgh               0.7;
    }
    equations
    {
        U                   0.3;
        "(h|e)"             0.3;
        k                   0.3;
        epsilon             0.3;
    }
}
```

Pressure-Velocity-Density Coupling

COMPRESSIBLE FLOWS

In compressible flows, the mass flux depends on both the velocity component normal to the cell face v_n and the (variable) density ρ. To correct the mass flux imbalance, both the density and the velocity must be corrected. The corrected mass flux on the '*e*' face of a CV can be expressed as:

$$\dot{m}_e^m = (\rho^{m-1} + \rho')_e (v_n^{m^*} + v_n')_e S_e$$

where ρ' and v_n' represent the density and velocity corrections, respectively. The mass flux correction is thus:

$$\dot{m'}_e = (\rho^{m-1} S v_n')_e + (v_n^{m^*} S \rho')_e + \underline{(\rho' v_n' S)}_e$$

The underscored term is usually neglected as it is of second order in the corrections and thus becomes zero more rapidly than the other two terms. Near convergence, this approximation is certainly permissible; one hopes that it does not affect the rate of convergence of the method when the solution is far from converged. This term can be taken into account using a predictor-corrector approach, as described for the treatment of non-orthogonality in the pressure-correction equation.

Pressure-Velocity-Density Coupling

$$\dot{m}'_e = (\rho^{m-1} S v'_n)_e + (v_n^{m^*} S \rho')_e + \underline{\cancel{(\rho' v'_n S)}_e} \simeq (\rho^{m-1} S v'_n)_e + (v_n^{m^*} S \rho')_e$$

The first term in the mass flux correction is identical to the one obtained for incompressible flows:

$$(\rho^{m-1} S v'_n)_e = -(\rho^{m-1} S \Delta V)_e \overline{\left(\frac{1}{A_P^{v_n}}\right)_e} \left(\frac{\delta\, p'}{\delta\, n}\right)_e$$

where n is the coordinate in the direction of the outward normal to the cell face. Since the coefficient A_p is the same for any Cartesian velocity component, we can take $A_P^{v_n} = A_P^u$.

Pressure-Velocity-Density Coupling

$$\dot{m}'_e = (\rho^{m-1} S v'_n)_e + (v_n^{m^*} S \rho')_e$$

The second term in the mass flux correction is due to compressibility; it involves the correction to density at the CV face. If the SIMPLE method is to be extended to compressible flows, we must also express the density correction in terms of the pressure correction. If the temperature T is regarded as fixed for one outer iteration:

$$\rho' \simeq \left(\frac{\partial \rho}{\partial p}\right)_T p' = \psi\, p'$$

For a perfect gas:

$$\left(\frac{\partial \rho}{\partial p}\right)_T = \frac{1}{RT} = \psi$$

Pressure-Velocity-Density Coupling

For other (non-perfect) gases, the derivative may need to be computed numerically. The converged solution is independent of this coefficient because all corrections are then zero; only the intermediate results are affected. It is important that the connection between the density and pressure corrections be qualitatively correct and the coefficient can, of course, influence the convergence rate of the method. The second term in the mass flux correction can then be written:

$$\left(v_n^{m^*} S \rho'\right)_e = \left(\frac{C_\rho \dot{m}^*}{\rho^{m-1}}\right)_e p'_e$$

The mass flux correction on the 'e' face of a CV is then:

$$\dot{m}'_e = \underbrace{-\left(\rho^{m-1} S \Delta V\right)_e \overline{\left(\frac{1}{A_P^{v_n}}\right)_e} \left(\frac{\delta p'}{\delta n}\right)_e}_{\texttt{incompressible part}} + \underbrace{\left(\frac{C_\rho \dot{m}^*}{\rho^{m-1}}\right)_e p'_e}_{\texttt{compressible, transonic/highRe}}$$

The value of p' at the cell face center and the normal component of the gradient of p' at the cell face center need to be approximated.

Pressure-Velocity-Density Coupling

The continuity equation, which must be satisfied by the corrected mass fluxes and density

$$\frac{\rho'_P \Delta V}{\Delta t} + \dot{m}^*_e + \dot{m}^*_w + \dot{m}^*_n + \dot{m}^*_s + Q^*_m = 0$$

If the already described relation $\rho' \simeq \left(\frac{\partial \rho}{\partial p}\right)_T p' = C_p\, p' = \psi\, p'$ is used to express ρ'_P in terms of p'_P, and the approximation of the mass flux correction $\dot{m}^*$ is substituted into this equation, the algebraic system of equations for the pressure correction is obtained:

$$A_P p'_P + \sum_l A_l p'_l = -Q^*_m$$

Despite the similarity in appearance to the pressure-correction equation for incompressible flows, there are important differences.

- The incompressible equation is a discretized Poisson equation, i.e. the coefficients represent an approximation to the Laplacian operator. For an incompressible flow, if the mass flux is prescribed at the boundary, the pressure may be indeterminate to within an additive constant. The presence of convective terms in the compressible pressure equation makes the solution unique.
- In the compressible case, there are contributions that represent the fact that the equation for the pressure in a compressible flow contains convective and unsteady terms, i.e. it is actually a convected wave equation.

Pressure Correction

Despite the similarity in appearance to the pressure-correction equation for incompressible flows, there are important differences.

$$\nabla^2 \, p^{n+1} = \frac{1}{\Delta t} \left[\frac{\partial \left(\psi \cdot p \right)}{\partial t} + \nabla \cdot \left(\rho^* \mathbf{u}^* \right) \right]$$

1. For an incompressible flow, if the mass flux is prescribed at the boundary, the pressure may be indeterminate to within an additive constant. The presence of convective terms in the compressible pressure equation makes the solution unique.
2. The relative importance of the two terms in the mass flux correction depends on the type of flow. The diffusive term is of order $(1/Ma^2)$ relative to the convective term so the Mach number is the determining factor:
   - at low Mach numbers, the Laplacian term dominates and we recover the Poisson equation;
   - at high Mach numbers (highly compressible flow), the convective term dominates, reflecting the hyperbolic nature of the flow. Solving the pressure-correction equation is then equivalent to solving the continuity equation for density.

Thus the pressure-correction method automatically adjusts to the local nature of the flow and the same method can be applied to the entire flow region.

Pressure Equation: discretization

When the pressure correction equation for compressible flows

$$\nabla^2 \, p^{n+1} = \frac{1}{\Delta t}\left[\frac{\partial\left(\psi \cdot p\right)}{\partial t} + \nabla \cdot \left(\rho^* \mathbf{u}^*\right)\right]$$

is solved in its discretized form, coefficients depend on the approximations used for the gradients and cell face values of the pressure correction

$$A_P p'_P + \sum_l A_l p'_l = -Q^*_m$$

- For the approximation of the Laplacian, central difference approximations are always applied.
- For the approximation of convective terms a variety of approximations may be used, just as is the case for the convective terms in the momentum equations. If higher-order approximations are used, the 'deferred correction' method may be used:
  - on the left-hand side of the equation, the matrix is constructed on the basis of the first-order upwind approximation while the right-hand side contains the difference between the higher-order approximation and the first-order upwind approximation, assuring that the method converges to the solution belonging to the higher-order approximation
  - if the grid is severely non-orthogonal, deferred correction can be used to simplify the pressure-correction equation.

Pressure Equation: discretization

When the pressure correction equation for compressible flows

$$\nabla^2 p^{n+1} = \frac{1}{\Delta t}\left[\frac{\partial(\psi \cdot p)}{\partial t} + \nabla \cdot (\rho^* \mathbf{u}^*)\right]$$

is solved in its discretized form, coefficients depend on the approximations used for the gradients and cell face values of the pressure correction

$$A_P p'_P + \sum_l A_l p'_l = -Q^*_m$$

- For the approximation of the Laplacian, central difference approximations are always applied.
- For the approximation of convective terms a variety of approximations may be used, just as

These differences are reflected in the pressure-correction equation: because the equation is no longer a pure Poisson equation, the central coefficient A_p is not the negative of the sum of the neighbor coefficients.

  - on the left-hand side of the equation, the matrix is constructed on the basis of the first-order upwind approximation while the right-hand side contains the difference between the higher-order approximation and the first-order upwind approximation, assuring that the method converges to the solution belonging to the higher-order approximation
  - if the grid is severely non-orthogonal, deferred correction can be used to simplify the pressure-correction equation.

Pressure correction in OpenFOAM

Where is pressure correction at the boundaries in OpenFOAM? Please have a look at the code, in `$FOAM_SOLVERS`, at the source files. You will find lot of `correctBoundaryConditions()` calls! An example in `pEqn.H` (see line 12):

```
1  ...
2
3  pEqn.solve();
4
5  if (simple.finalNonOrthogonalIter())
6  {
7      phi = phiHbyA - pEqn.flux();
8  }
9
10 p.relax();
11 U = HbyA - rAU*fvc::grad(p);
12 U.correctBoundaryConditions();
```

Thank you for your attention!

contact: federico.piscaglia@polimi.it