The SIMPLE (Semi-IMplicit algorithm for Pressure-Linked Equations) algorithm

Incompressible N-S equations are difficult to solve numerically with traditional FVMs because of coupling between pressure and velocity in the momentum equations

Reminder: incompressible N-S is a set of transient 4 equations (1 scalar continuity eq. and 3 scalar momentum equations, one per each spatial component of the vector momentum equation) in 4 variables (u_x, u_y, u_z, p)

Reinterpretation: rather than thinking to the continuity equ. as an additional DOF of the system one thinks of it as a constraint instead i.e. I solve momentum eq. independently and then I want my solution to satisfy the continuity eq.

Solution: derive from momentum eq. an eq. for pressure so that the system is decoupled; afterward derive a corrector s.t. it ensures that $\bar{u}$ does satisfy the constraint imposed by the continuity eq.

$$\begin{cases} \partial_t \bar{u} + \bar{u}(\bar{u}\cdot\bar{\nabla}) - \nu\Delta(\bar{u}) = \overbrace{-\bar{\nabla}(p)}^{\text{coupling } p-\bar{u}} & \text{(momentum eq.)} \quad (1) \\ \bar{\nabla}\cdot\bar{u} = 0 & \text{(continuity eq.)} \quad (2) \end{cases}$$

$$\downarrow$$

Apply div operator to (1) (here shown for 2D case only, $\bar{x} = (x, y)$)

$$\begin{aligned} \bar{\nabla}\cdot((1)) \Rightarrow\ & \overbrace{\partial_x(\partial_t u_x) + \partial_y(\partial_t u_y)}^{\bar{\nabla}\cdot\partial_t\bar{u}} + \underbrace{\partial_x(u_x\partial_x u_x) + \partial_x(u_y\partial_y u_x) + \partial_y(u_x\partial_x u_y) + \partial_y(u_y\partial_y u_y)}_{\bar{\nabla}\cdot((\bar{u}\cdot\bar{\nabla})\bar{u})} \\ & \underbrace{-\nu\Big(\partial_x(\partial_x^2 u_x) + \partial_x(\partial_y^2 u_x) + \partial_y(\partial_x^2 u_y) + \partial_y(\partial_y^2 u_y)\Big)}_{\bar{\nabla}\cdot(\nu\Delta\bar{u})} = \underbrace{-(\partial_x^2 p + \partial_y^2 p)}_{\Delta(p)} \overset{*}{\Rightarrow} \end{aligned}$$

$\ast$

$\Rightarrow$ rearrange (exchang ∂t with $\partial x, \partial y$ and apply the der. product rule)

$$
\begin{aligned}
&\partial t\underbrace{(\partial_x U_x + \partial_y U_y)}_{=\bar{\nabla}\cdot\bar{u}=0} + \underbrace{\partial_x U_x \partial_x U_x}_{(\partial_x U_x)^2} + U_x \partial_x^2 U_x + \partial_x U_y \partial_x U_x + U_y \partial_x^2 U_x + \\
&+ \underbrace{\partial_y U_y \partial_y U_y}_{(\partial_y U_y)^2} + U_y \partial_y^2 U_y + \partial_y U_x \partial_y U_y + U_x \partial_y^2 U_y - \\
&- \nu\left(\partial_x^3 U_x + \partial_x \partial_y^2 U_x + \partial_y^3 U_y + \partial_y \partial_x^2 U_y\right) = \\
&= -\left(\partial_x^2 p + \partial_y^2 p\right)
\end{aligned}
$$

(This is to impose that the eq. satisfies the continuity constraint)

$$\Downarrow$$

use the quadratic formula to get

$$(\partial_x U_x)^2 + (\partial_y U_y)^2 = \underbrace{(\partial_x U_x + \partial_y U_y)^2}_{=(\bar{\nabla}\cdot\bar{u})^2=0} - 2\,\partial_x U_x \partial_y U_y$$

and replace it back into the eq.

$$
\begin{aligned}
&-2\,\partial_x U_x \partial_y U_y + \left(U_x \partial_x^2 U_x + U_y \partial_y^2 U_y\right) + \left(U_y \partial_x^2 U_x + U_x \partial_y^2 U_y\right) + \\
&+ \partial_x U_y \partial_x U_x + \partial_y U_x \partial_y U_y - \nu\left(\partial_x^3 U_x + \partial_x \partial_y^2 U_x + \partial_y \partial_x^2 U_y + \partial_y^3 U_y\right) = \\
&= -\left(\partial_x^2 p + \partial_y^2 p\right)
\end{aligned}
$$

The new equation, called Poisson's pressure equation

$$\Delta p = \overline{\nabla} \cdot ((\overline{u} \cdot \overline{\nabla}) \overline{u}) = \Delta u + 2 \partial_x u_y \partial_y u_x \qquad (3)$$

is an identity derived from (1) that ensures (2) therefore it's physical. This eq. is a new eq. for pressure which can be used, together with (1) to form a new set of decoupled equations

$$\begin{cases} \partial_t \overline{u} + (\overline{u} \cdot \overline{\nabla}) \overline{u} - \nu \Delta \overline{u} = -\overline{\nabla} p \\ \overline{\nabla} \cdot \overline{u} = 0 \end{cases} \Rightarrow \begin{cases} \partial_t \overline{u} + (\overline{u} \cdot \overline{\nabla}) \overline{u} - \nu \Delta \overline{u} = -\overline{\nabla} p \\ \Delta p = \overline{\nabla} \cdot ((\overline{u} \cdot \overline{\nabla}) \overline{u}) \end{cases}$$

coupled — decoupled

set s.t. it satisfies $\overline{\nabla} \cdot \overline{u} = 0$

Now we approximate the decoupled system using a FV scheme to obtain the algebraic system

$$\begin{cases} \overline{\overline{M}}\, \overline{u}_h^{(k)} = \overline{b} \quad , \overline{b} = -\overline{\nabla} p_h^{(k)} & (4) \\ \Delta p_h^{(k)} = \overline{\overline{F}}\, \overline{u}_h^{(k)} & (5) \end{cases}$$

, where $k = 0, 1, 2, 3$ is the iterative index of the algorithm that solves the system (e.g. Gauss-Seidel)

Notice that by using Gauss-Seidel or SOR we get:

$$\overline{\overline{D}}\, \overline{u}_h^{(k)} = -\overline{\overline{H}}\, \overline{u}_h^{(k-1)} - \overline{\nabla} p_h^{(k)} \Rightarrow \overline{u}_h^{(k)} = -\overline{\overline{D}}^{-1} \overline{\overline{H}}\, \overline{u}_h^{(k-1)} - \overline{\overline{D}}^{-1} \overline{\nabla} p_h^{(k)} \qquad (6)$$

where $\overline{\overline{D}}$ is the diagonal part of the decomposed $\overline{\overline{M}} = \overline{\overline{D}} + \overline{\overline{H}}$ and thus it's straight-forward to invert

$$(\overline{\overline{D}}^{-1})_{jj} = ((\overline{\overline{D}})_{jj})^{-1} \quad \forall j = 1, \dots, N$$

Eq. (6) will be our momentum predictor in which (by solving it) we'll derive a prediction for the velocity field called intermediate velocity field.

Eq (5) will be our pressure correction equation in which we compute a pressure value s.t. it satisfies the continuity equation and that we will use to update the velocity field

SIMPLE PSEUDOCODE

set BCs
guess an initial value for $p^{(0)}$
for $k=0$: miter || toll < 1e-10
    solve (6) to obtain $\overset{*}{u}_h^{(k)}$
    put $\overset{*}{u}_h^{(k)}$ into (5) and obtain updated $p^{(k)}$ that satisfies (2)
    set $p^{(k+1)} = p^{(k)}$
end

SIMPLE IN OF

```
p.storePrevIter();
tmp<fvVectorMatrix> UEqn
(
    fvm::div(phi, U) - fvm::laplacian(mu,U)
);
solve (UEqn == - fvc::grad(p));
volScalarField  D=UEqn().A( );
U=UEqn.H()/D;
fvScalarMatrix pEqn
(
    fvm::laplacian (1.0/D, p)== fvc::div(phi)
);
pEqn.solve();
U -= fvc::grad(p)/D;
U.correctBoundaryConditions();
```

The SIMPLE algorithm is usually not recommended for unsteady flows the reason being that as the discretization step Δt in the time derivative

$$\bar{u}_t \cong \frac{\bar{u}_h^{(n+1)} - \bar{u}_h^{(n)}}{\Delta t} \qquad \Delta t = t_{n+1} - t_n$$

gets smaller, the more this term tends to dominate the spatial discretizations in $\Delta \bar{u}$ and $\bar{\nabla} \cdot (\bar{u} \otimes \bar{u})$.
As such the linear system tends to be more diagonally dominant and requires under-relaxation of the momentum equation