side is explicitly calculated using known field values from the current or previous time step; the corresponding time derivative and gradient can be programmed using `fvc::ddt` and `fvc::grad`, respectively.

**Listing 1** Implementation of temperature equation 7 with OpenFOAM (in `EEqn.H`).

```cpp
fvScalarMatrix TEqn(
    (porosity*rho*Cp+(1.0-porosity)*rho_rock*cp_rock)*fvm::ddt(T)
    +fvm::div(phi*fvc::interpolate(Cp),T)
    ==
    fvm::laplacian(kr,T) + fvm::Sp(fvc::div(phi*fvc::interpolate(Cp)),T)
          + mu/permeability*magSqr(U) + fvm::Sp(alphaP*(porosity*fvc::ddt(p)+(U &
           fvc::grad(p)))),T)
);
TEqn.solve();
```

3. The pressure field $p^{n+1}$ is implicitly computed by solving the pressure equation (Equation 5), the code snippet is shown in Listing 2. The temperature temporal term and divergence of $\phi_g$ on the right-hand side are evaluated explicitly by using `fvc::ddt(T)` and `fvc::div(phig)` (see line 7 in Listing 2). Although pressure boundary conditions are customized