```
// Initialise fluid field pointer lists

PtrList<multiphaseSystem> thermoFluid(fluidRegions.size());
PtrList<volScalarField> rhoFluid(fluidRegions.size());
PtrList<volScalarField> TFluid(fluidRegions.size());
PtrList<volVectorField> UFluid(fluidRegions.size());
PtrList<uniformDimensionedScalarField> hRefFluid(fluidRegions.size());
PtrList<volScalarField> ghFluid(fluidRegions.size());
PtrList<surfaceScalarField> ghfFluid(fluidRegions.size());
PtrList<volScalarField> kappaLK(fluidRegions.size());
PtrList<CompressibleTurbulenceModel<multiphaseSystem>> turbulenceFluid(fluidRegions.size());
PtrList<volScalarField> p_rghFluid(fluidRegions.size());
PtrList<volScalarField> KFluid(fluidRegions.size());
PtrList<volScalarField> dpdtFluid(fluidRegions.size());
PtrList<multivariateSurfaceInterpolationScheme<scalar>::fieldTable>
    fieldsFluid(fluidRegions.size());
List<scalar> initialMassFluid(fluidRegions.size());
List<bool> frozenFlowFluid(fluidRegions.size(), false);
List<bool> correctPhiFluid(fluidRegions.size(), true);
List<bool> ddtCorrFluid(fluidRegions.size(), true);
PtrList<fv::options> fluidFvOptions(fluidRegions.size());
List<label> pRefCellFluid(fluidRegions.size());
List<scalar> pRefValueFluid(fluidRegions.size());
PtrList<dimensionedScalar> rhoMinFluid(fluidRegions.size());
PtrList<dimensionedScalar> rhoMaxFluid(fluidRegions.size());
PtrList<dimensionedScalar> rhoRFluid(fluidRegions.size());
PtrList<volScalarField> rhokFluid(fluidRegions.size());
PtrList<volScalarField> CpFluid(fluidRegions.size());
PtrList<volScalarField> rhoCpFluid(fluidRegions.size());
PtrList<volScalarField> pFluid(fluidRegions.size());
PtrList<surfaceScalarField> rhoPhiFluid(fluidRegions.size());
PtrList<pimpleControl> pimpleFluid(fluidRegions.size());

PtrList<pressureControl> pressureControls(fluidRegions.size());

const uniformDimensionedVectorField& g = meshObjects::gravity::New(runTime);

// Populate fluid field pointer lists
forAll(fluidRegions, i)
{
    Info<< "*** Reading fluid mesh thermophysical properties for region "
        << fluidRegions[i].name() << nl << endl;
    pimpleFluid.set
    (
        i,
        new pimpleControl(fluidRegions[i])
    );

    p_rghFluid.set
    (
        i,
        new volScalarField
        (
            IOobject
            (
```

```
            "p_rgh",
            runTime.timeName(),
            fluidRegions[i],
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        fluidRegions[i]
    )
);
Info<< "    Adding to UFluid\n" << endl;
UFluid.set
(
    i,
    new volVectorField
    (
        IOobject
        (
            "U",
            runTime.timeName(),
            fluidRegions[i],
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        fluidRegions[i]
    )
);
Info<< "    Adding to TFluid\n" << endl;
TFluid.set
(
    i,
    new volScalarField
    (
        IOobject
        (
            "T",
            runTime.timeName(),
            fluidRegions[i],
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        fluidRegions[i]
    )
);


Info<< "    Adding to hRefFluid\n" << endl;
hRefFluid.set
(
    i,
    new uniformDimensionedScalarField
    (
        IOobject
        (
            "hRef",
            runTime.constant(),
```

```
            fluidRegions[i],
            IOobject::READ_IF_PRESENT,
            IOobject::NO_WRITE
        ),
        dimensionedScalar("hRef", dimLength, Zero)
    )
);

Info<< "Calculating field g.h\n" << endl;
#include "readGravitationalAcceleration.H"

dimensionedScalar ghRef
(
    mag(g.value()) > SMALL
  ? g & (cmptMag(g.value())/mag(g.value()))*hRefFluid[i]
  : dimensionedScalar("ghRef", g.dimensions()*dimLength, 0)
);

Info<< "    Adding to ghFluid\n" << endl;
ghFluid.set
(
    i,
    new volScalarField
    (
        "gh",
        (g & fluidRegions[i].C()) - ghRef
    )
);

Info<< "    Adding to ghfFluid\n" << endl;
ghfFluid.set
(
    i,
    new surfaceScalarField
    (
        "ghf",
        (g & fluidRegions[i].Cf()) - ghRef
    )
);

pFluid.set
(
    i,
    new volScalarField
    (
        IOobject
        (
            "p",
            runTime.timeName(),
            fluidRegions[i],
            IOobject::NO_READ,
            IOobject::AUTO_WRITE
        ),
        p_rghFluid[i]
    )
```

```
);
Info<< "    Adding to thermoFluid\n" << endl;
thermoFluid.set(i, multiphaseSystem::New(fluidRegions[i]).ptr());
Info<< "    Adding to rhoFluid\n" << endl;
rhoFluid.set
(
    i,
    new volScalarField
    (
        IOobject
        (
            "rho",
            runTime.timeName(),
            fluidRegions[i],
            IOobject::NO_READ,
            IOobject::AUTO_WRITE
        ),
        thermoFluid[i].rho()
    )
);
rhoFluid[i].oldTime();

const dictionary& thermophysicalPropertiesFluid =
    fluidRegions[i].lookupObject<IOdictionary>("thermophysicalProperties.liquid");


rhoRFluid.set
(
    i,
    new dimensionedScalar
    (
    "rhoRef",
    dimDensity,
    thermophysicalPropertiesFluid.subDict("mixture").subDict("equationOfState")
    )
);

Info<< "    Calculating rhok\n" << endl;
rhokFluid.set
(
    i,
    new volScalarField
    (
        IOobject
        (
            "rhok",
            runTime.timeName(),
            fluidRegions[i],
            IOobject::NO_READ,
            IOobject::AUTO_WRITE
        ),
        thermoFluid[i].rho() - rhoRFluid[i]
    )
);
rhokFluid[i].oldTime();
```

```
Info<< "    Adding to rhoPhiFluid\n" << endl;
rhoPhiFluid.set
(
    i,
    new surfaceScalarField
    (
        IOobject
        (
            "rhoPhi",
            runTime.timeName(),
            fluidRegions[i],
            IOobject::NO_READ,
            IOobject::NO_WRITE
        ),
        thermoFluid[i].rhoPhi()
    )
);

Info<< "    Adding to CpFluid\n" << endl;
CpFluid.set
(
    i,
    new volScalarField
    (
        IOobject
        (
            "Cp",
            runTime.timeName(),
            fluidRegions[i],
            IOobject::NO_READ,
            IOobject::AUTO_WRITE
        ),
        thermoFluid[i].Cp()
    )
);

Info<< "    Adding to Kappa Lookup\n" << endl;
kappaLK.set
(
    i,
    new volScalarField
    (
        IOobject
        (
            "kappa",
            runTime.timeName(),
            fluidRegions[i],
            IOobject::NO_READ,
            IOobject::AUTO_WRITE
        ),
        thermoFluid[i].kappa()
    )
);
```

```
Info<< "    Adding to rhoCpFluid\n" << endl;
rhoCpFluid.set
(
    i,
    new volScalarField
    (
        IOobject
        (
            "rhoCp",
            runTime.timeName(),
            fluidRegions[i],
            IOobject::NO_READ,
            IOobject::NO_WRITE
        ),
        thermoFluid[i].rho()*thermoFluid[i].Cp()
    )
);
rhoCpFluid[i].oldTime();

Info<< "    Adding to turbulenceFluid\n" << endl;
turbulenceFluid.set
(
    i,
    CompressibleTurbulenceModel<multiphaseSystem>::New
    (
        rhoFluid[i],
        UFluid[i],
        rhoPhiFluid[i],
        thermoFluid[i]
    )
);

pFluid[i] = p_rghFluid[i] + rhokFluid[i]*ghFluid[i];

Info<< "    Adding to KFluid\n" << endl;
KFluid.set
(
    i,
    new volScalarField
    (
        "K",
        0.5*magSqr(UFluid[i])
    )
);

Info<< "    Adding to dpdtFluid\n" << endl;
dpdtFluid.set
(
    i,
    new volScalarField
    (
        IOobject
        (
            "dpdt",
            runTime.timeName(),
```

```
            fluidRegions[i]
        ),
        fluidRegions[i],
        dimensionedScalar(thermoFluid[i].p().dimensions()/dimTime, Zero)
    )
);

pimpleFluid[i].dict().readIfPresent("correctPhi", correctPhiFluid[i]);
pimpleFluid[i].dict().readIfPresent("ddtCorr", ddtCorrFluid[i]);
const dictionary& pimpleDict =
    fluidRegions[i].solutionDict().subDict("PIMPLE");

pimpleDict.readIfPresent("frozenFlow", frozenFlowFluid[i]);

rhoMaxFluid.set
(
    i,
    new dimensionedScalar("rhoMax", dimDensity, GREAT, pimpleDict)
);

rhoMinFluid.set
(
    i,
    new dimensionedScalar("rhoMin", dimDensity, Zero, pimpleDict)
);

pressureControls.set
(
    i,
    new pressureControl(thermoFluid[i].p(), rhoFluid[i], pimpleDict, false)
);

Info<< "    Adding fvOptions\n" << endl;
fluidFvOptions.set
(
    i,
    new fv::options(fluidRegions[i])
);

pRefCellFluid[i] = 0;
pRefValueFluid[i] = 0.0;

setRefCell
(
    pFluid[i],
    p_rghFluid[i],
    pimpleDict,
    pRefCellFluid[i],
    pRefValueFluid[i]
);

if (p_rghFluid[i].needReference())
{
    pFluid[i] += dimensionedScalar
        (
```

```
            "p",
            pFluid[i].dimensions(),
            pRefValueFluid[i] - getRefCellValue(pFluid[i], pRefCellFluid[i])
        );
        p_rghFluid[i] = pFluid[i] - rhokFluid[i]*ghFluid[i];
    }

}
```