Cantera in OpenFOAM

Installation and settings

- Installation in MSI
- Refer to the folder “OpenFOAM_cantera” in the shared COMMON folder

Cantera chemistry reader in OpenFOAM

- constant/thermophysicalProperties settings

```
thermoType
{
    type            heRhoThermo;
    mixture         reactingMixture;
    transport       sutherland;
    thermo          janaf;
    energy          sensibleEnthalpy;
    equationOfState perfectGas;
    specie          specie;
}

//chemistryReader foamChemistryReader;
//foamChemistryFile "$FOAM_CASE/constant/reactions";
//foamChemistryThermoFile "$FOAM_CASE/constant/thermo";
chemistryReader canteraChemistryReader;
canteraChemistryFile "gri30.cti";
canteraFileID "gri30";
canteraTransportFile "$FOAM_CASE/chemkin/transportProperties";
```

Specify chemistryReader and the Cantera mechanism files

Cantera chemistry models in OpenFOAM

- In constant/chemistryProperties file
- Default: standard chemistry model

```
chemistryType
{
    solver              ode;
}
```

- Specify chemistry model: cantera

```
chemistryType
{
    solver              ode;
    method              cantera;
}
```

- Specify chemistry model: TDAC

```
chemistryType
{
    solver              ode;
    method              TDAC;
}
```

Note:
canteraChemistryModel and standardChemistryModel are inherited classes from basicChemistryModel. TDAC is inherited from standardChemistryModel. Future work is to replace all the functions in TDAC inherited from standardChemistryModel by the functions in canteraChemistryModel and develop a TDAC_cantera class.

Transport models

- constant/thermophysicalProperties

```
thermoType
{
    type            canteraPsiThermo;
    mixture         reactingMixture;
    transport       sutherland;
    thermo          janaf;
    energy          sensibleEnthalpy;
    equationOfState perfectGas;
    specie          specie;
}
```

Still Sutherland here,
but the viscosity and thermal diffusion is calculated
in canteraPsiThermo.C

```
//chemistryReader foamChemistryReader;
//foamChemistryFile "$FOAM_CASE/constant/reactionsGRI";
//foamChemistryThermoFile "$FOAM_CASE/constant/thermo.compressibleGasGRI";
chemistryReader canteraChemistryReader;
canteraChemistryFile "gri30.cti";
canteraFileID "gri30";
canteraTransportFile "$FOAM_CASE/constant/transportProperties";
transportModel "mixtureAveraged";
// transportModel "multiComponent"
```

Specify which transport model you want to use.

Transport models

- Modification needed in solver (use reactingFoam as an example)
- In YEqn.H

```
PtrList<volVectorField> Dm =
dynamic_cast<const reactingMixture<gasHThermoPhysics>&>
(composition).Dm(p, T, Y, rho);

PtrList<volScalarField> hsi(Y.size());
PtrList<surfaceScalarField> J(Y.size());
PtrList<volScalarField> D(Y.size());
```

Dm: diffusion velocity, its is calculated in reactingMixtureI.H

Definition of sensible enthalpy, diffusion flux
and mass diffusion coefficient

Transport models

```
forAll(Y, i)
{
    hsi.set
    (
        i,
        new volScalarField
        (
            IOobject
            (
                "hsi",
                mesh.time().timeName(),
                mesh
            ),
            mesh,
            dimensionedScalar
            (
                "hsi",
                dimEnergy/dimMass,
                Zero
            )
        )
    );

    D.set
    (
        i,
        mag(Dm[i])/(dimensionedScalar("SMALL", dimensionSet(0,-1,0,0,0),
        Foam::SMALL) + mag(fvc::grad(Y[i]).ref()))
    );

    J.set
    (
        i,
        linearInterpolate(Dm[i]*rho) & mesh.Sf()
    );
}
```

Initialize hsi to zero

Calculate mass diffusion coefficient

Calculate mass flux

Transport models

```
forAll (Y, i)
{
    volScalarField& tHsi = hsi[i];
    forAll(tHsi, celli)
    {
        tHsi[celli] = composition.Hs(i, p[celli], T[celli]);
    }
    volScalarField::Boundary& Bf = tHsi.boundaryFieldRef();
    forAll(Bf, patchi)
    {
        forAll(Bf[patchi], facei)
        {
            Bf[patchi][facei] =
            composition.Hs
            (
                i,
                p.boundaryField()[patchi][facei],
                T.boundaryField()[patchi][facei]
            );
        }
    }
}
```

Calculate hsi in the field

Transport models

```
fvScalarMatrix YiEqn
(
    fvm::ddt(rho, Yi)
  + mvConvection->fvmDiv(phi, Yi)
  - fvm::laplacian(rho*D[i], Yi)
 // - fvm::laplacian(turbulence->muEff(), Yi)
 ==
    reaction->R(Yi)
  + fvOptions(rho, Yi)
);
```

Updated diffusion term, D is either calculated by mixture averaged model or multi-component model

OpenFOAM original diffusion term. With this term, Schmidt number is assumed to be unity

Transport model

- In EEqn.H, add the heat flux terms caused by mass flux and mass gradient

```
forAll(Y, k)
{
    EEqn -= fvc::laplacian(turbulence->alphaEff()*hsi[k], Y[k]);
    EEqn -= fvc::div(J[k], hsi[k], "div(Ji,hsi)");
}
```

- Remember modify the species equation (YEqn) and energy equation (EEqn) accordingly when using mixture averaged or multicomponent transport model

CVODES

- Seulex is the traditional stiff ODE solver in OpenFOAM and it is found to fail to solve the chemistry source terms for fuels with NTC. Additionally, it is also found to be unstable reported in lieteratures.
- CVODES from the sundials library is widely used in the combustion community (e.g Cantera, NGA).
- Hence, CVODES was also coupled into OpenFOAM.
- In chemistryProperties file, specify the ODE solver.

```
odeCoeffs
{
    solver          CVODES;
    absTol          1e-10;
    relTol          1e-3;
}
```