Flow CCS functionality

Alf B. Rustad with *huge* help from Tor Harald Sandve
CO2STORE in Flow

-- .............................................................................................................................
--
-- RUNSPEC SECTION
--
-- .............................................................................................................................
RUNSPEC
--
-- FLUID TYPES AND TRACER OPTIONS
--
--
-- ACTIVATE CO2 STORAGE IN THE MODEL (OPM FLOW CO2 STORAGE KEYWORD)
--
CO2STORE
--
-- ACTIVATE GAS-WATER THE MODEL (OPM FLOW KEYWORD)
--
GASWAT
--
-- DISSOLVED GAS IN WATER IS PRESENT IN THE RUN (OPM FLOW KEYWORD)
--
DISGASW
--
-- VAPORIZED WATER IN DRY/WET GAS IS PRESENT IN THE RUN (OPM FLOW KEYWORD)
--
VAPWAT
Salinity in Flow

The first example activates the standard Brine model and has no terminating “/”.

```
--
-- ACTIVATE STANDARD BRINE MODEL IN THE RUN
--
BRINE
```

The second example illustrates how to activate OPM Flow’s Salt Precipitation model.

```
--
-- ACTIVATE STANDARD BRINE MODEL IN THE RUN
--
BRINE
--
-- ACTIVATE THE OPM FLOW SALT PRECIPITATION MODEL (OPM FLOW KEYWORD)
--
PRECSALT
--
-- VAPORIZED WATER IN DRY/WET GAS IS PRESENT IN THE RUN (OPM FLOW KEYWORD)
--
VAPWAT
```
Case 1; synthetic models

- The models are homogeneous and isothermal
- Injection rate is $1 \times 10^6 \text{Sm}^3$ (0.68 mtpa) from a single injector in 25 years
- With and without salinity
The grids

E300

OPM Flow

Pflotran
Gas injection

Gas Injection Rate

- F_CO2STORE_03_F
- E_CO2STORE_03_F

Gas Injection Total

- F_CO2STORE_03_F
- E_CO2STORE_03_F
Dissolved CO2 mass
• Model with salinity
Average pressure
Injection rates

Gas Injection Rate

- F_CO2STORE_04_F
- E_CO2STORE_04_F

Gas Injection Total

- F_CO2STORE_04_F
- E_CO2STORE_04_F
Reservoir rate
Dissolution
Observations

• For models without salinity
  • Generally good agreement between simulators

• For models with salinity deviations are observed
  • For FWCD the mixing model matters
  • For reservoir rates we need to understand difference
• Thermal simulations
  • With salinity
The grids

Pflotran

E300
Average pressure
Dissolution
CO2 densities

\[ P = \left( \frac{RT_K}{V - b_{mix}} \right) \left( \frac{a_{mix}}{T_K^{1/2}V(V + b_{mix})} \right) \]

Figure 3. Comparison of density between Span-Wagner used by OPM Flow and Redlich-Kwong.
Solubility model and ACTCO2S

The mutual solubilities of water and CO$_2$ are then expressed as follows:

$$y_{H_2O} = A(1 - x_{CO_2})$$

$$x_{CO_2} = B(1 - y_{H_2O})$$

with parameters $A$ and $B$ defined as (using values of $K$ given by Eqs. 5-7)

$$A = \frac{K_{H_2O} y_{H_2O}}{\Phi_{H_2O} P_{tot}}$$

$$B = \frac{\Phi_{CO_2} P_{tot}}{55.508 \gamma_{CO_2} K_{CO_2}}$$

$$\ln(y_{H_2O}) = (A_M - 2A_M x_{H_2O}) x_{CO_2}^2$$  \hspace{1cm} (12)

$$\ln(y_{CO_2}) = 2A_M x_{CO_2} x_{H_2O}^2$$  \hspace{1cm} (13)

In these equations, $A_M$ is a Margules parameter that, after several tests, we chose to express as a function of temperature, as follows:

$$A_M = 0 \text{ (thus } y_{CO_2} \text{ and } y_{H_2O} = 1 \text{) at } T \leq 100^\circ \text{C}$$  \hspace{1cm} (14)

$$A_M = a(T_K - 373.15) + b(T_K - 373.15)^2 \text{ at } T > 100^\circ \text{C}$$  \hspace{1cm} (15)

Duan and Sun 2003 (Spycher and Pruess 2005)

Rumpf et al. 1994 (Spycher and Pruess 2005)

Spycher and Pruess 2009
The effect of salt

\[
y'_{\text{CO}_2} = \left(1 + \frac{\sum m_{i \neq \text{CO}_2}}{55.508}\right) \exp\left\{2\lambda(m_{\text{Na}} + m_{\text{K}} + 2m_{\text{Ca}} + 2m_{\text{Mg}}) + \xi m_{\text{Cl}}(m_{\text{Na}} + m_{\text{K}} + m_{\text{Ca}} + m_{\text{Mg}}) - 0.07 m_{\text{SO}_4}\right\}
\]

\[
B' = \frac{\Phi_{\text{CO}_2} P_{\text{tot}}}{55.508 y_{\text{CO}_2} y'_{\text{CO}_2} K_{\text{CO}_2}}
\]

The mutual solubilities of water and \(\text{CO}_2\) are then expressed as follows:

\[
y_{\text{H}_2\text{O}} = A(1 - x_{\text{CO}_2})
\]

\[
x_{\text{CO}_2} = B(1 - y_{\text{H}_2\text{O}})
\]

with parameters \(A\) and \(B\) defined as (using values of \(K\) given by Eqs. 5–7)

\[
A = \frac{K_{\text{H}_2\text{O}} y_{\text{H}_2\text{O}}}{\Phi_{\text{H}_2\text{O}} P_{\text{tot}}}
\]

\[
B = \frac{\Phi_{\text{CO}_2} P_{\text{tot}}}{55.508 y_{\text{CO}_2} K_{\text{CO}_2}}
\]
The models
Where we left off

• Deviation between E300/pFlotran and Flow was close to 9 degrees Celsius

• Pflotran was claimed to use Span-Wagner?

• It was commented that temperature could be critical, Sleipner as example
/* Tables for CO2 fluid properties calculated according to Span and Wagner (1996).
   * THIS AN AUTO-GENERATED FILE! DO NOT EDIT IT!
   * Temperature range: 280.000 K to 400.000 K, using 200 sampling points
   * Pressure range: 0.100 MPa to 100.000 MPa, using 500 sampling points
   * Generated using:
   */

struct TabulatedDensityTraits {
  typedef double Scalar;
  static const char *name;
  static const int numX = 200;
  static const Scalar xMin;
  static const Scalar xMax;
  static const int numY = 500;
  static const Scalar yMin;
  static const Scalar yMax;

  static const std::vector<std::vector<Scalar>> vals;
};

inline const double TabulatedDensityTraits::xMin = 2.00000000000000e+02;
inline const double TabulatedDensityTraits::xMax = 4.00000000000000e+02;
inline const double TabulatedDensityTraits::yMin = 1.00000000000000e+00;
inline const double TabulatedDensityTraits::yMax = 1.00000000000000e+08;
inline const char *TabulatedDensityTraits::name = "density";

inline const std::vector<std::vector<double>> TabulatedDensityTraits::vals =
{
  { 1.92062665274410e+00,
      2.24837179664650e+00,
      2.68799303157356e+00,
      3.15091719428634e+00,
      3.63590724036341e+00,
      4.12627141801649e+00,
      4.63509001956886e+00,
      5.20794710517262e+00,
      5.86262882920214e+00,
      6.59647786923172e+00,
      7.32082513131737e+00,
      8.06868274798979e+01,
      8.83286692966833e+01,
      9.61082513131737e+01,
      1.03859497892255e+02,
      1.12068749101649e+02,
      1.20327430827459e+02,
      1.2866721453141e+02,
      1.3692573281732e+02,
      1.44951887474298e+02,
      1.5271209196321e+02,
      1.60065567625713e+02,
      1.67037539784197e+02,
      1.73619425745379e+02,
      1.80382674798979e+01,
      1.87059497892255e+01,
      1.93632178427111e+01,
      1.99937746274870e+01,
      2.05943931265710e+01,
      2.11698415982010e+01,
      2.172030175789e+01,
      2.22483717966465e+01,
      2.27430827459159e+01,
      2.32082513131737e+01,
      2.36346081908617e+01,
      2.40404784575014e+01,
      2.44245263727865e+01,
      2.47719570906450e+01,
      2.51063573728772e+01,
      2.54182460017828e+01,
      2.57089414517063e+01,
      2.59831901001291e+01,
      2.62398020951809e+01,
      2.6468804173723e+01,
  },
CO2 Enthalpy main difference

```cpp
168 169 const Evaluation pressure,
170 171 bool extrapolate = false);
172 {
173 174  // TEST 2nd degree polynomial fitted with Coolprop data in temperature
174 175  // range (273.15, 403.15) with reference state T=288.15 K (=15 C) and P = 101325 Pa
175 176  return (temperature - 273.15 - 15)*(8.43923504e+02 + 4.54571769e-01*(temperature - 273.15 - 15) - 0.805 * (pressure - 1.01325e5));
176 177  // return tabulatedEnthalpy.eval(temperature, pressure, extrapolate);
177 178 }
178 179 /*!
179 
180 
181 
182 
183 */
```
Comparing Enthalpy representations
Comparison with Pflotran and E300

- Pflotran confirmed not to have heat of dissolution

- Both Pflotran and E300 confirms the low temperatures of OPM Flow when Span-Wagner properties for CO2 enthalpy are used
CO2 dissolution secondary effect

- Dissolution effect for CO2 gives a temperature «bump»

- Removing dissolution model from Flow removes the main difference

- What about temperature effect of vaporization?
Fig. 6. The heat of solution of CO₂ in water (the model of this study vs. experimental data).
CO2 Dissolution in Flow

```cpp
// heat of dissolution for CO2 according to Fig. 6 in Duan and Sun 2003 (kJ/kg)

// In the relevant temperature ranges CO2 dissolution is exothermic /

// delta_hCO2 = (-57.4375 - T * 0.1325) * 1000/44;

// delta_hCO2 = (-57.4375 + T * 0.1325) * 1000/44;

// enthalpy contribution of CO2 (kJ/kg) /

h_g = CO2::gasEnthalpy(T, p, extrapolate)/1E3 + delta_hCO2;

h_g = CO2::gasEnthalpy(T, p, extrapolate)/1E3; // delta_hCO2;

// Enthalpy of brine with dissolved CO2 /

return (h_is1 - X_CO2_w*hw + h_g*X_CO2_w)*1E3; // J/kg/
```
Water density (diff with Ezrkohi, RS is CO2 content)

REL_DIFF

- NaCl = 0.0, RS = 0.0
- NaCl = 0.0, RS = 5.0
- NaCl = 0.0, RS = 10.0
- NaCl = 0.0, RS = 15.0
- NaCl = 0.0, RS = 20.0

- NaCl = 1.0, RS = 0.0
- NaCl = 1.0, RS = 5.0
- NaCl = 1.0, RS = 10.0
- NaCl = 1.0, RS = 15.0
- NaCl = 1.0, RS = 20.0

- NaCl = 2.0, RS = 0.0
- NaCl = 2.0, RS = 5.0
- NaCl = 2.0, RS = 10.0
- NaCl = 2.0, RS = 15.0
- NaCl = 2.0, RS = 20.0

- NaCl = 5.0, RS = 0.0
- NaCl = 5.0, RS = 5.0
- NaCl = 5.0, RS = 10.0
- NaCl = 5.0, RS = 15.0
- NaCl = 5.0, RS = 20.0

Pressure [bar]

<table>
<thead>
<tr>
<th>Temp [°C]</th>
<th>50</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>0.0004</td>
<td>0.0002</td>
</tr>
<tr>
<td>400</td>
<td>0.0004</td>
<td>0.0002</td>
</tr>
<tr>
<td>600</td>
<td>0.0004</td>
<td>0.0002</td>
</tr>
<tr>
<td>800</td>
<td>0.0004</td>
<td>0.0002</td>
</tr>
<tr>
<td>1000</td>
<td>0.0004</td>
<td>0.0002</td>
</tr>
</tbody>
</table>
Density models for saline water in E300, Intersect, Pflotran and Flow


• Pflotran supports Batzle & Wang in addition to Ezrokhi for density of saline water, while the CO2 solution impact is from Duan & Sun, ref. https://docs.opengosim.com/manual/input_deck/thermodynamic_procs/eos_water/

• Note that Batzle & Wang is more accurate than Ezrokhi. However, as implemented in Flow it only takes NaCl into account, will need to be extended to take other salts into account. While Ezrokhi allows to manually provide parameters to the model to account for salt combination.