RMG Study Group

Liquid Reactor

Yunsie Chung

9/29/2016

Massachusetts Institute of Technology

Agenda

- Liquid Reactor Theoretical Backgrounds
- Input File Format
- Code Changes & Fixes in Progress
- Solvation Database
- CoolProp Module
- Issues / Future Work

Backgrounds: Solvent Effects in RMG

1. Diffusion Limits	✓☐ →	-Stokes-Einstein equation -Effective rate constant of a bimolecular reaction
2. Thermo Corrections	✓☐ →	-Linear solvation energy relationships (LSERs) to estimate thermo at 298 K -Japas & Harvey relationships to estimate thermo at high T (*will be merged soon*)
3. Kinetic Corrections	x →	-Belinda in Prof. West's group is currently working on this

**** All the corrections are made based on the dilute solution assumption***

Diffusion Limits

- Diffusivity and effective rate constant are calculated from:

$$D_{AB} = \frac{k_B T}{6\pi\mu R_A} \qquad k_{eff} = \frac{4\pi R D k_{int}}{4\pi R D + k_{int}}$$

$\mu = solvent\ viscosity,\quad R_A = solute\ radius,\quad k_{int} = intrinsic\ rate\ from\ gas\ phase\ reactions$

$R = sum\ of\ solutes\ radii,\quad D = sum\ of reactants\ diffusivities$

- Solvent viscosity calculated from the solvent parameters in the RMG-database
- Solute radius calculated from the McGowan Volume (V)
- If the solute descriptor V is not found in RMG-database, it is estimated using atom sizes

Current RMG Thermo Correction

The Abraham LSER:

$$\log_{10} K(298K) = c + \overset{\text{Solvent descriptor}}{e}\underset{\text{Solute descriptor}}{E} + sS + aA + bB + lL$$

The Mintz correlation:

$$\Delta H^{\circ}_{solv}(298K) = c' + e'E + s'S + a'A + b'B + l'L$$

- Experimentally fitted 25 solvent & 125 solute descriptors are available
- Other solute descriptors can be estimated by the Platts group additivity

Current RMG Thermo Correction

Linear temperature dependence is assumed (fails at approximately T > 400 K):

$$\Delta G^{\circ}_{solv}(T) = \Delta H^{\circ}_{solv}(298K) - T\Delta S^{\circ}_{solv}(298K)$$

Thermo correction for both solvent & solute:

$$\Delta G^{l}_{\mathrm{f}}(T) = \underbrace{\Delta G^{g}_{\mathrm{f}}}_{\text{gas phase free energy}}(T) + \Delta G^{\circ}_{solv}(T)$$

Standard States ° :

ideal gas and dilute ideal solution with equal concentrations of solutes

Massachusetts Institute of Technology

RMG Thermo Correction in Progress: Solutes

- For 298K ≤ T ≤ 420K:

$$\boxed{T ln\left(K_{2,1}^{\infty}\right) = A + B\left(1 - \frac{T}{T_c}\right)^{0.355} + CT^{0.59} exp\left(1 - \frac{T}{T_c}\right)}$$

Eqn. (1)

- For 420K ≤ T < T_c:

$$\boxed{T ln\left(K_{2,1}^{\infty}\right) = D(\rho_1^l - \rho_{c,1})}$$

Eqn. (2)

$$K_{2,1}^{\infty}(T) = \lim_{x_2 \to 0}\left(\frac{y_2}{x_2}\right) \qquad \Delta G_{solv}^{\circ}(T) = RT \ln\left(\frac{K_{2,1}^{\infty}(T) P_1^{vap}(T)}{RT C_1^l(T)}\right)$$

$A, B, C, D = empirical\ parameters$
$1 = solvent$
$2 = solute$
$c_1^l = solvent\ concentration$

**** Solvent properties are obtained from CoolProp module, evaluated along the saturation curve***
**** If solvents cannot be found in CoolProp, only simple correction using the Abraham and Mintz LSERs is applied***

Massachusetts Institute of Technology

Eqn. (1): Allan H. Harvey, *AIChE Journal*, 42 (1491-1494), 1996
Eqn. (2): M. L. Japas, J. M. H. Levelt Sengers, *AIChE Journal*, 35 (705-713), 1989

RMG Thermo Correction in Progress: Solutes

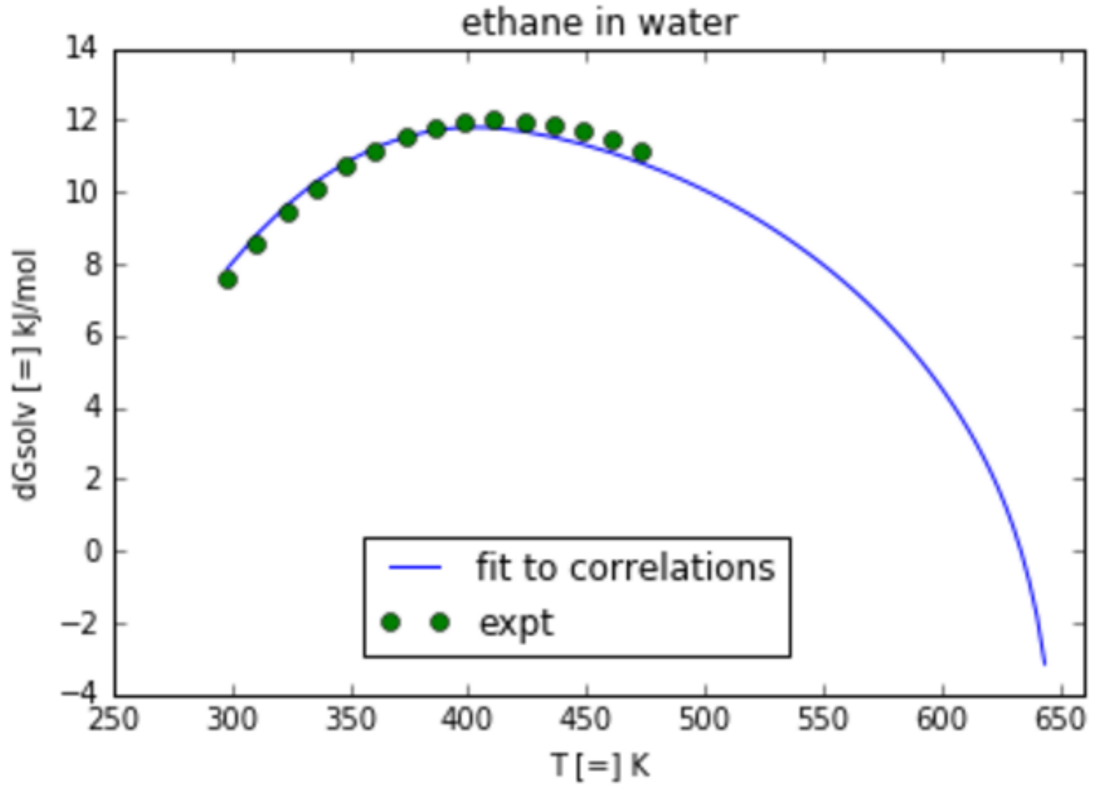
Determining the parameters (A, B, C, D):

1. ΔG°_{solv}(298K) from the Abraham LSER

2. $\left.\frac{d\Delta G^{\circ}_{solv}}{dT}\right|_{298K, eqn\ (1)}$ from the Mintz LSER

3. $\left.\frac{d\Delta G^{\circ}_{solv}}{dT}\right|_{420K, eqn\ (1)} = \left.\frac{d\Delta G^{\circ}_{solv}}{dT}\right|_{420K, eqn\ (2)}$

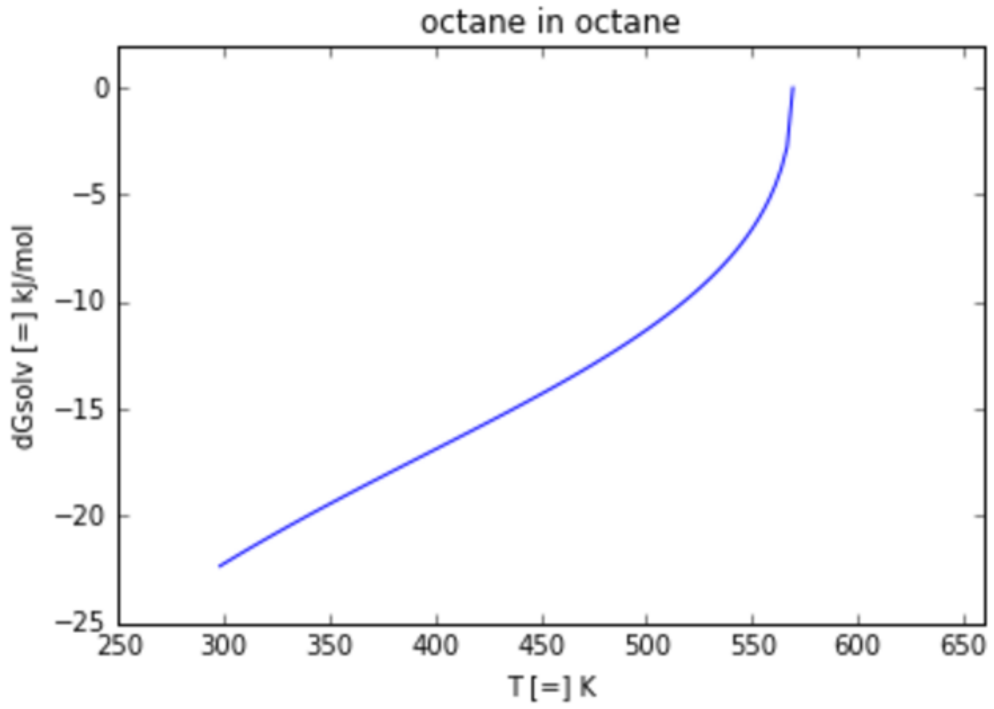
4. ΔG°_{solv}(420K, *eqn(1)*) = ΔG°_{solv}(420K, *eqn(2)*)



RMG Thermo Correction in Progress: Solvents

- For solvent species in CoolProp, ΔG°_{solv} can be directly computed at any temperatures

$$\Delta G^{\circ}_{solv} = -RT \ln \left(\frac{\rho^{g}}{\rho^{l}} \right)_{eq}$$



Massachusetts Institute of Technology

RMG Thermo Correction in Progress

Completed Part:

- ΔG°_{solv} is calculated using the new correlations
- After the thermo correction is applied, NASA and Wilhoit models are fitted to the corrected gibbs free energy

In Progrees:

- The thermo output file should have the temperature range up to T_c
- Make unittests and documentations

Input File Format

```
# Reaction systems
liquidReactor(
    temperature=(450,'K'),
    initialConcentrations={
        "octane": (6.154e-3,'mol/cm^3'),
        "oxygen": (4.953e-6,'mol/cm^3'),
    },
#    terminationTime=(300,'s'),
    terminationConversion={'octane': 0.9},
    constantSpecies=['oxygen']
)

solvation(
    solvent='octane'
)
```

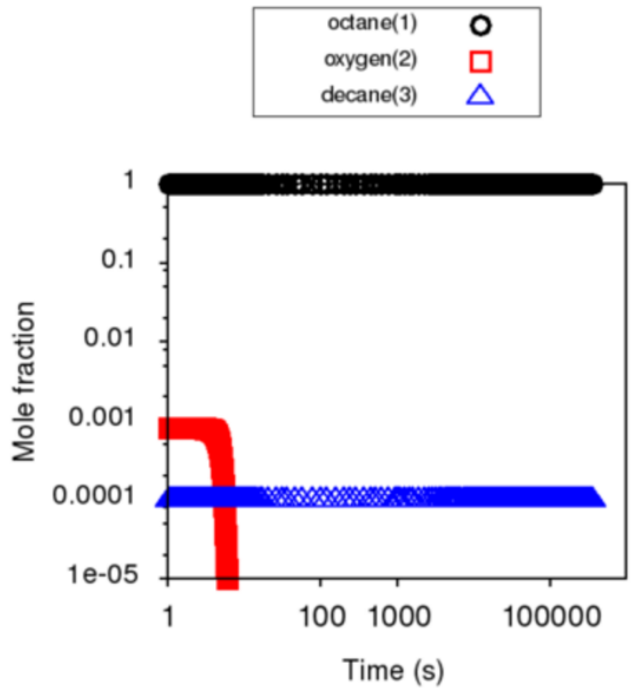
$\rightarrow$ Independent of pressure

$\rightarrow$
-Only concentration acceptable
-The solver output is in mole fraction
-Solvent must be listed as one of the initial species

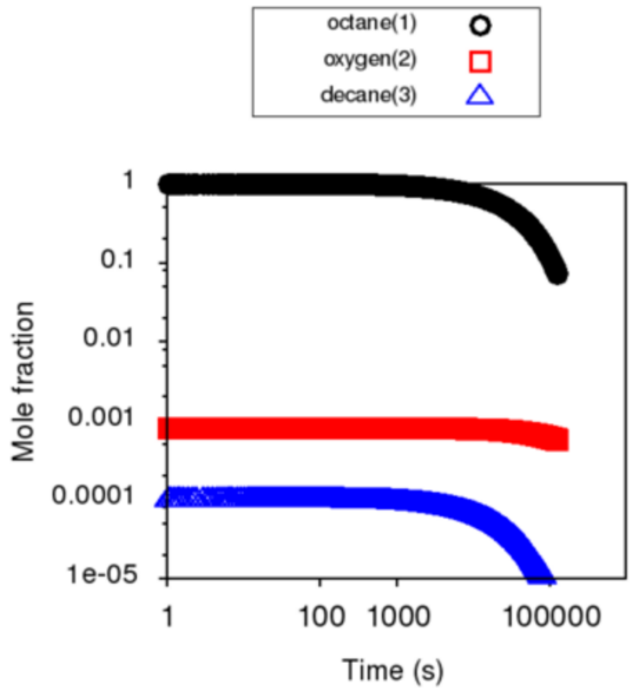
$\rightarrow$
-Optional
-Multiple constant species possible

$\rightarrow$
-Can specify only one solvent
-The solvent solvation data must be in RMG-database

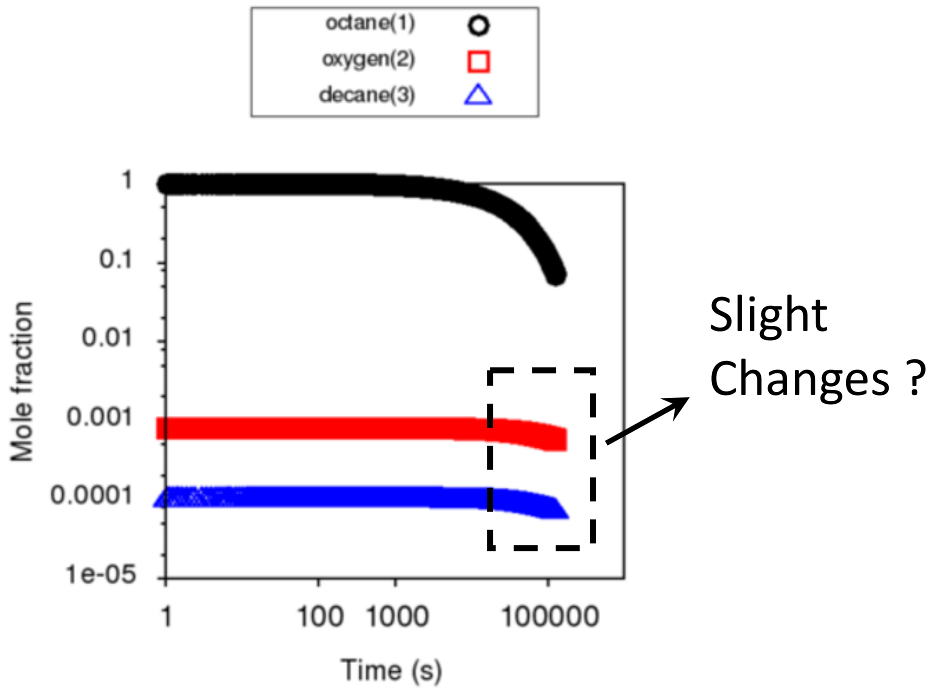
Constant Species Simulation: Octane/Decane Oxidation



No constant species

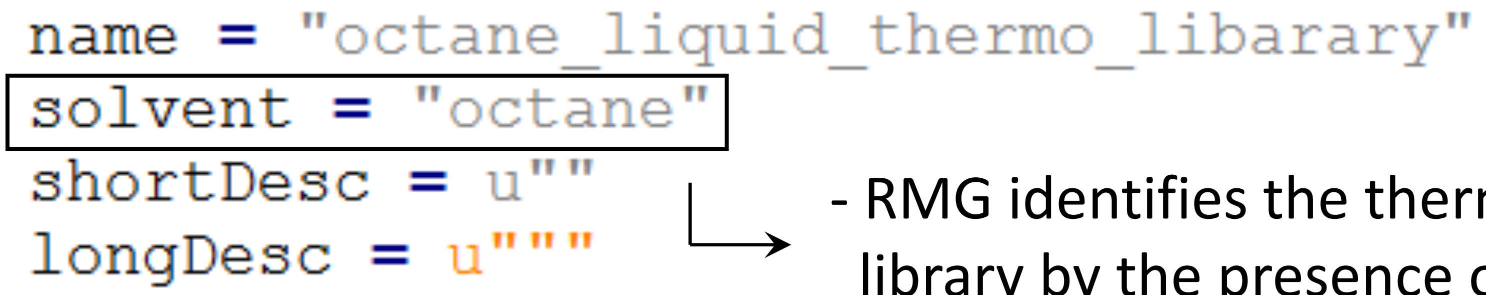


O2 constant



O2 & Decane constant

Liquid Thermo Library



```
name = "octane_liquid_thermo_libarary"
solvent = "octane"
shortDesc = u""
longDesc = u"""
```

- RMG identifies the thermo library as “liquid thermo” library by the presence of the solvent block
- Each library must be specific to one solvent
- If the solvents in the thermo library and in the input file are different, RMG will raise an error

Code Change and Fixes in Progress

Current Code	New Code
• Solvent related attributes are stored under rmgpy.rmg.model.Species() by creating a subclass of Species()	• A new wrapper class for solvent, Solvent() is created and stored under RMG() class. • Remove rmgpy.rmg.model.Species() • Move coreSizeAtCreation attribute to under rmgpy.Species()
• Checks whether the solvent in listed in the initial species in rmgpy.rmg.main	• Identifies the solvent from the initial species in rmgpy.rmg.input • The identified solvent species is stored as solventSpecies object in Solvent() class

Code Change and Fixes in Progress

Current Code	New Code
• Same diffusion limit is applied for all species	• For the reaction in which the solvent is reacting, that forward / reverse rate is not diffusion-limited
• For solvents in CoolProp, if T ≥ Tc, CoolProp will crash	• Checks whether the rxn T exceeds the critical T of the solvent. If it does, it raises error and displays the error message with the critical T
• If conda environment is not updated, RMG will crash	• makefile checks for the CoolProp module • If it is not found, it displays the error message with the command for updating the conda environment

Solvation Database

Solvent descriptors:

- Abraham and Mintz parameters
- Viscosity parameters
- Solute parameters for intrinsic rate correction in H-abstraction rxn (alpha and beta) and dielectric constant (eps). *Currently not used*

- inCoolProp: True if the solvent can be found in CoolProp. False if not
- NameinCoolProp: solvent name that can be used in CoolProp. None if inCoolProp is False

$\hookrightarrow$ *Not in RMG-database yet.*
To be merged

Solute descriptors: Abraham parameters

Solute & solvent descriptors can be found in: http://rmg.mit.edu/database/solvation/libraries/

CoolProp Module

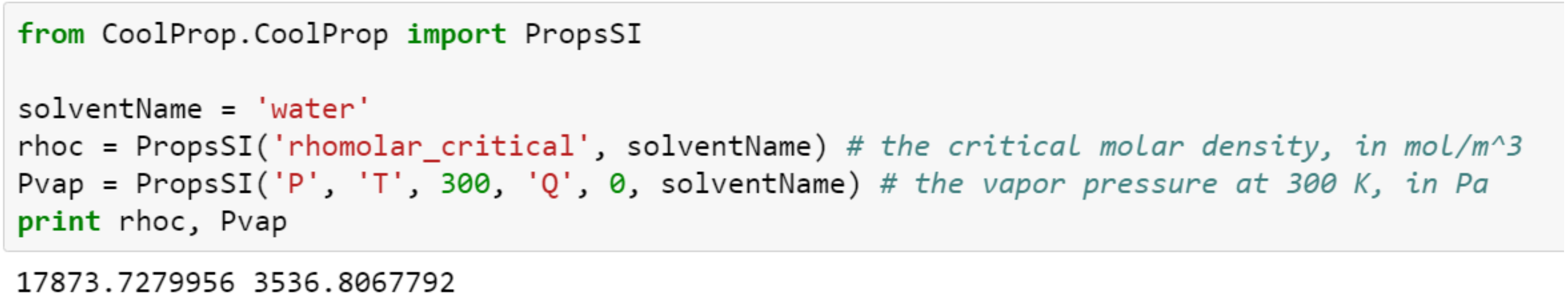
- Computes thermo and transport properties for 122 pure components and some mixture solvents
- Implements EOS explicit in Helmholtz energy, modified Benedict-Webb-Rubin EOS, and an extended corresponding states model
- Mixture: mixing rules to the Helmholtz energy

- List of fluids can be found in:
  http://www.coolprop.org/fluid_properties/PurePseudoPure.html#list-of-fluids

- List of available properties can be found in:
  http://www.coolprop.org/coolprop/HighLevelAPI.html#propssi-function

Using CoolProp Module

- Can be easily used in Python / ipython notebook (jupyter notebook)
- Connie uploaded the slides on how to use ipython notebook in dropbox
- Sample ipython notebook codes:



```python
from CoolProp.CoolProp import PropsSI

solventName = 'water'
rhoc = PropsSI('rhomolar_critical', solventName) # the critical molar density, in mol/m^3
Pvap = PropsSI('P', 'T', 300, 'Q', 0, solventName) # the vapor pressure at 300 K, in Pa
print rhoc, Pvap
```

```
17873.7279956 3536.8067792
```

Issues / Future Work

- RMG Chemkin output file does not contain the information on diffusion limit and its T dependence

  -> maybe use MultiArrhenius models?

- Chemkin does not have liquid reactors
- Solvent effect on the intrinsic rate
- Others?