Thematic workshop on Thermodynamic Modeling

Ian H. Bell

National Institute of Standards and Technology, Boulder, CO, USA

Outline

- About Me
- Introduction/Scope
- Thermodynamic Properties and Equations of State
- Building an EOS
- Flash Routines and Phase Equilibria: Pure Fluids
- Transport Properties
- Mixtures

About Me

- Doctoral Research at Purdue University
    - Flooded compression in scroll compressors
        - Oil absorbs the heat of compression of the refrigerant
        - Increase cycle efficiency with IHX and oil flooding
    - Experimental campaigns
- Dissertation
    - *Theoretical and experimental analysis of liquid flooded compression in scroll compressors*

- Postdoc, 2012-2014, *University of Liège, Liège, Belgium*
    - Simulation of scroll compressors
    - Development of open-source thermophysical property library CoolProp
- Postdoc, 2015-present, *National Institute of Standards and Technology, Boulder, Colorado*
    - Development of mixture models, equations of state, etc.
    - Critical point calculation routines
    - Psychrometric properties from mixture models

Introduction

- *Today*: compressible/real pure fluids and mixtures
- *Not*: humid air, incompressible fluids (brines and secondary working fluids)
- **Today's goal**: Crack open the black boxes of thermophysical property libraries (REFPROP, CoolProp, TREND)

What properties do we care about?

For preliminary cycle design:

- Temperature: T
- Pressure: p
- Density: ρ
- Specific enthalpy: h
- Specific entropy: s
- Specific heat capacity: c_p, c_v

For component design:

- Thermal conductivity: λ
- Viscosity: η

Thermodynamic Properties and Equations of State

p-ρ-T properties

- We can measure pressure, temperature, and density (and other things like speed-of-sound)
- Q: How do we describe the relationship between these properties?
- A: An equation of state

Equation of state

- Expresses relationship between thermodynamic properties
- Wikipedia has nice treatment
- Active field of research in last century+, will not cover whole field here
- But highest accuracy formulations (discussed here), are much more complex

- Starting at the beginning, the ideal gas law:

$$p = \rho R T$$

- All units are molar-specific base-SI:
    - $[p]$ = Pa
    - $[\rho]$ = mol/m^3
    - $[R]$ = 8.314462618... J/(mol K) (exact)
    - $[T]$ = K

Ideal-Gas Law Limitations

- Low pressure gases only
- Doesn't work well near saturation
- Not so great for polar fluids either
- Doesn't give you entropy/enthalpy directly
- But in some cases, it's good enough, or it serves as a good guess

Cubic Equations of State

- van der Waals (1873):
$$p = \frac{RT}{v-b} - \frac{a}{v^2}$$
- SRK (1972):
$$p = \frac{RT}{v-b} - \frac{a}{v(v+b)}$$
- Peng Robinson (1976):
$$p = \frac{RT}{v-b} - \frac{a}{v^2 + 2bv - b^2}$$

- Can be expressed in a common form (Michelsen):

$$p = \frac{RT}{v-b} - \frac{a}{(v+\Delta_1 b)(v+\Delta_2 b)}$$

- Can be converted to Helmholtz energy $\alpha^{\rm r}$ according to the method of Bell and Jäger (J. Res. NIST, 2016)

Cubic Equations of State:

- Valid over entire surface (liquid, vapor, supercritical)
- Accuracy for VLE is adequate
- Simple to understand and implement
- Even now quite popular in industry (with some modifications)
- If T, p are known, explicit solution for v (actually $Z = \frac{pv}{RT}$) possible:
$$AZ^3 + BZ^2 + CZ + D = 0$$
- Number of roots can be as many as 3 (for instance in VLE)

Limitations

- Accuracy for liquid density is quite poor (can be off by 30% - volume translation can help)

Multiparameter EOS

- Nowadays, we have developed more accurate formulations for pure fluids
- They are explicit in Helmholtz energy (yet another mysterious thermodynamic property) as a function of volume and temperature
- Split into two parts:
  - $a = a^{\mathrm{r}} + a^0$
  - $\alpha = \frac{a}{RT} = \alpha^{\mathrm{r}} + \alpha^0$

Features

- Highest accuracy for VLE and single-phase state
- Valid over entire surface (liquid, vapor, supercritical)

Limitations

- Complex to implement $\rightarrow$ slow(er)
- No explicit solution for density given T, p as input variables
- Flexibility of form yields some "crazy" behavior if you are not careful

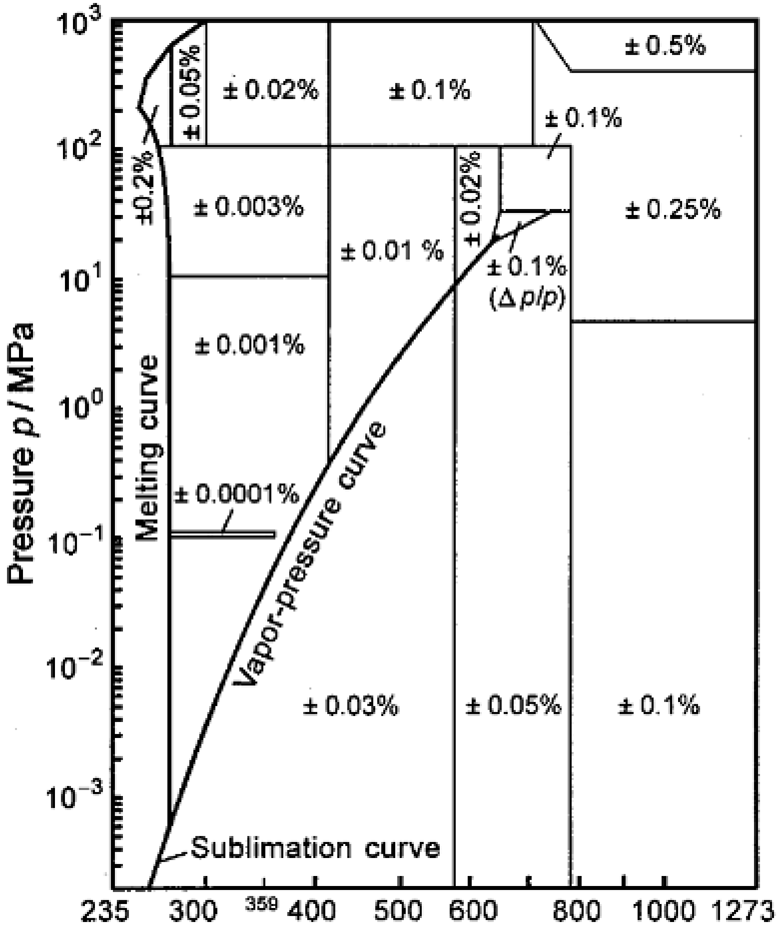
Thermodynamic potentials (why A?)

- There are four primary fundamental thermodynamic potentials (though others exist)
  - $s, v \rightarrow u$
  - $s, p \rightarrow h$
  - $v, T \rightarrow a$
  - $p, T \rightarrow g$

- Thermodynamic potential allows ANY other thermodynamic property to be obtained by derivatives of the potential with respect to independent variables
- Cannot measure entropy, though, so a or g are the choices for potential
- Gibbs energy derivative discontinuous in T, p at phase transitions, also not good
- Helmholtz energy it is!

Differing levels of precision

- Reference
    - Exceptionally high accuracy EOS based on very accurate experiments
    - E.g. Argon, Nitrogen, CO_2, Water, Methane, Ethylene
- Industrial
    - Many fluids, using functional forms proven to work well
    - Some generalized formulations, with only the coefficients being fitted



Uncertainty of water density $\Delta\rho$ (Wagner & Pruss, JPCRD, 2001)

Ideal-gas part

- Total Helmholtz is energy given by: $a = u - Ts$ on a specific basis
- Or non-dimensionalized: $\frac{a}{RT} = \alpha = \frac{u}{RT} - \frac{s}{R}$
- But we know that $u = h - pv$, and for an ideal gas $u = h - RT$ because $pv = RT$, therefore

$$\alpha^0 = \frac{h^0}{RT} - 1 - \frac{s^0}{R}$$

Residual part

- Entirely empirical, not governed by theory
- For instance, for propane:

$$\alpha^{\mathrm{r}} = \sum_{k=1}^{5} N_k \delta^{d_k} \tau^{t_k} + \sum_{k=6}^{11} N_k \delta^{d_k} \tau^{t_k} \exp(-\delta^{l_k}) + \sum_{k=12}^{18} N_k \delta^{d_k} \tau^{t_k} \exp(-\eta_k (\delta - \varepsilon_k)^2 - \beta_k (\tau - \gamma_k)^2)$$

- Here we use reduced variables $\delta = \rho/\rho_c$ and $\tau = T_c/T$
- Water and CO_2 have complicated non-analytic terms that have fallen out of favor

Useful relationships:

$$p = \rho RT\left[1 + \delta\left(\frac{\partial\alpha^r}{\partial\delta}\right)_\tau\right]$$

$$\frac{h}{RT} = \tau\left[\left(\frac{\partial\alpha^0}{\partial\tau}\right)_\delta + \left(\frac{\partial\alpha^r}{\partial\tau}\right)_\delta\right] + \delta\left(\frac{\partial\alpha^r}{\partial\delta}\right)_\tau + 1$$

$$\frac{s}{R} = \tau\left[\left(\frac{\partial\alpha^0}{\partial\tau}\right)_\delta + \left(\frac{\partial\alpha^r}{\partial\tau}\right)_\delta\right] - \alpha^0 - \alpha^r$$

- And so on...
- Valid for pure fluids or multi-fluid mixture model (see later)
- "Flash" call can be computationally very expensive; we will revisit this point

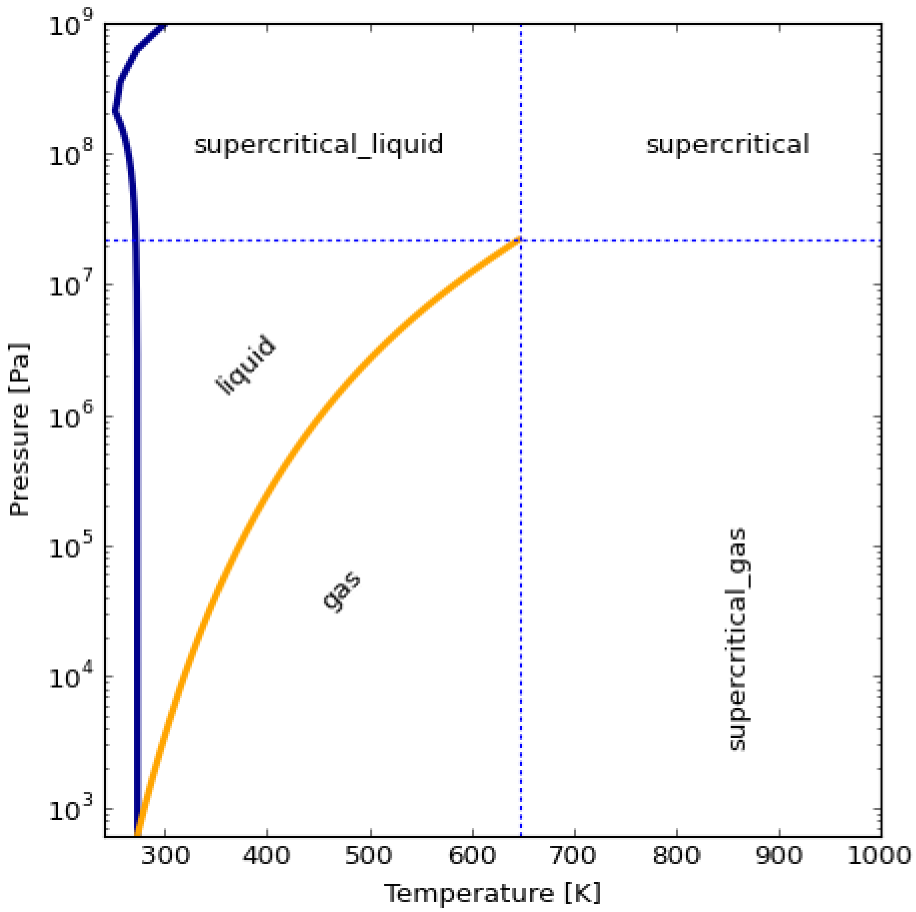
Flash Routines and Phase Equilibria: Pure Fluids

What is a flash calculation?

- The EOS has as independent variables T (as τ) and ρ (as δ)
- But often you know other thermodynamic variables:
    - p and h
    - p and T
    - ...
- Must iterate to find T and ρ (flash)
- Phase equilibria also possible inputs
    - T and vapor quality
    - p and vapor quality
- Sometimes, inputs can yield multiple solutions (T, u or T, h)

PT flash

- $\rho = f(T, p)$
- Simplest flash calculation, but not simple !



PT for water

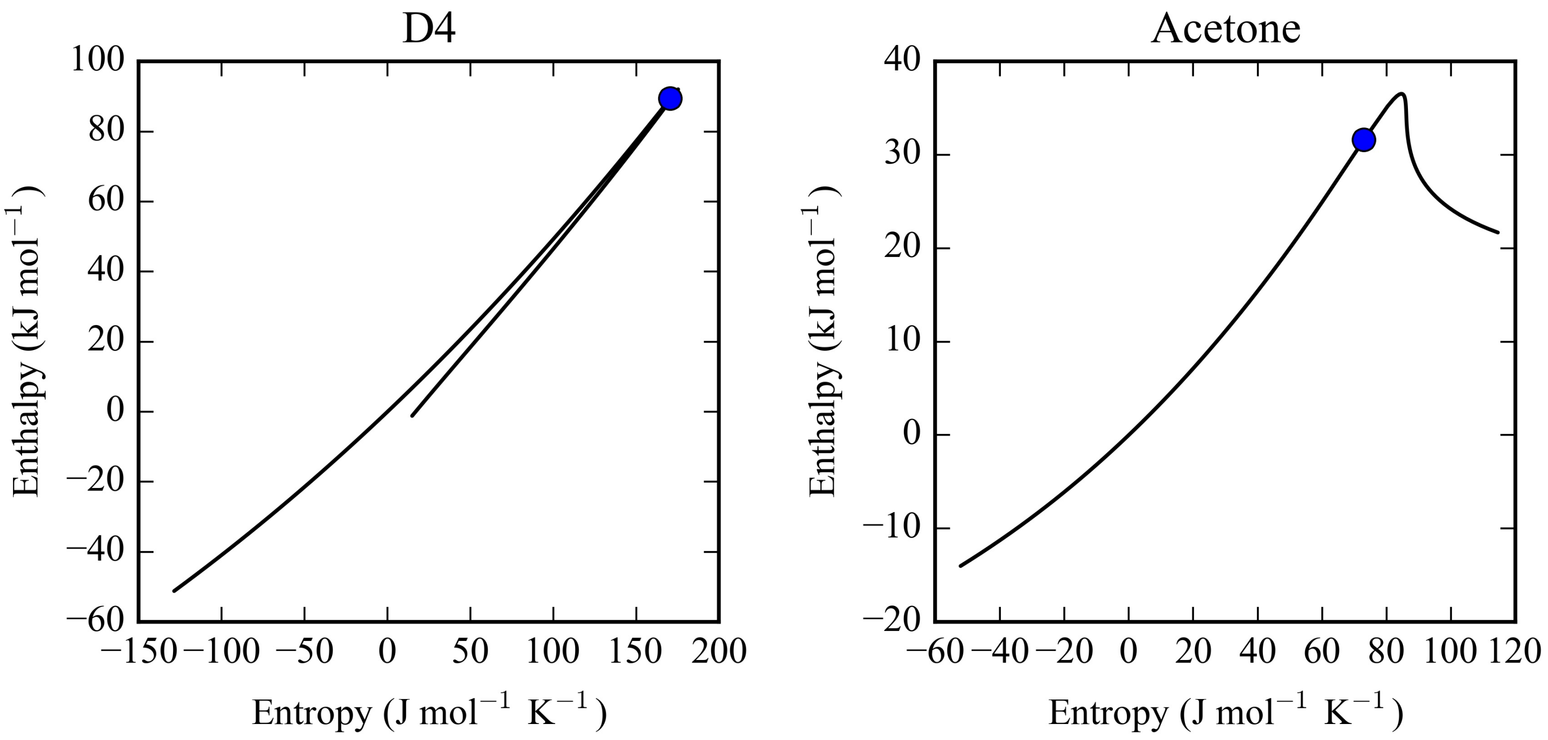
- Q1: Where am I?
- Q2: Given my location, what information do I know?
    - gas/supercritical: ideal-gas/SRK ok as first guess for ρ (explicit solution)
    - liquid: density is greater than saturated liquid density (for $p < p_c$)

- Algorithm
  - Given the guess density, drive the residual function $F(\rho)$ to zero
    $$F(\rho) = p(T, \rho) - p_{\text{given}}$$
  - Here it is a one-dimensional function of ρ, and we know the analytic derivatives
    $$\left(\frac{\partial p}{\partial \rho}\right)_T \text{and} \left(\frac{\partial^2 p}{\partial \rho^2}\right)_T$$
  - Use Secant or Halley's method for $F(\rho) \to 0$
  - Also possible to use Brent's method with quadratic updates if you have bounds, correct solution with Secant/Halley not guaranteed

Phase Equilibria (Pure Fluid)

- At vapor-liquid equilibrium:
    - Vapor and liquid phases at same T, p (thermal equilibrium)
    - Gibbs energy the same for both phases (chemical equilibrium)
        - Rate(!) of material transfer is balanced
- Metastability not considered

HS flash



D4
Acetone
Enthalpy (kJ mol^{-1})
Entropy (J mol^{-1} K^{-1})
100
80
60
40
20
0
−20
−40
−60
−150
−100
−50
0
50
100
150
200
Enthalpy (kJ mol^{-1})
Entropy (J mol^{-1} K^{-1})
40
30
20
10
0
−10
−20
−60
−40
−20
0
20
40
60
80
100
120

Mixtures

- **Why mixtures?**
  - Environmental concerns (ODP, GWP, flammability, etc.)
  - Much more complex to model
  - New "interesting" things to worry about (composition, phase stability, critical points)
  - Many blends form "boring" mixtures that behave like pure fluids

Mixture modeling

- GERG formulation for α^{r}

$$\alpha^{\mathrm{r}} = \alpha^{\mathrm{r}}_{LM} + \alpha^{\mathrm{r}}_{A}$$

$$\alpha^{\mathrm{r}}_{\mathrm{LM}}(\delta, \tau, \mathbf{x}) = \sum_{i=1}^{N} x_i \alpha^{\mathrm{r}}_{oi}(\delta, \tau)$$

$$\alpha^{\mathrm{r}}_{A}(\delta, \tau, \mathbf{x}) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} x_i x_j F_{ij} \alpha^{\mathrm{r}}_{ij}(\delta, \tau)$$

Reducing functions

- $\tau = T_\mathrm{r}(\mathbf{x})/T$ and $\delta = \rho/\rho_\mathrm{r}(\mathbf{x})$

$$T_\mathrm{r}(\mathbf{x}) = \sum_{i=1}^{N} x_i^2 T_{c,i} + \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} 2\beta_{T,ij}\gamma_{T,ij} \frac{x_i x_j (x_i + x_j)}{\beta_{T,ij}^2 x_i + x_j} (T_{c,i} T_{c,j})^{0.5}$$

$$\frac{1}{\rho_\mathrm{r}(\mathbf{x})} = \sum_{i=1}^{N} x_i^2 \frac{1}{\rho_{c,i}} + \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \beta_{v,ij}\gamma_{v,ij} \frac{x_i x_j (x_i + x_j)}{\beta_{v,ij}^2 x_i + x_j} \frac{1}{4} \left(\frac{1}{\rho_{c,i}^{1/3}} + \frac{1}{\rho_{c,j}^{1/3}} \right)^3$$

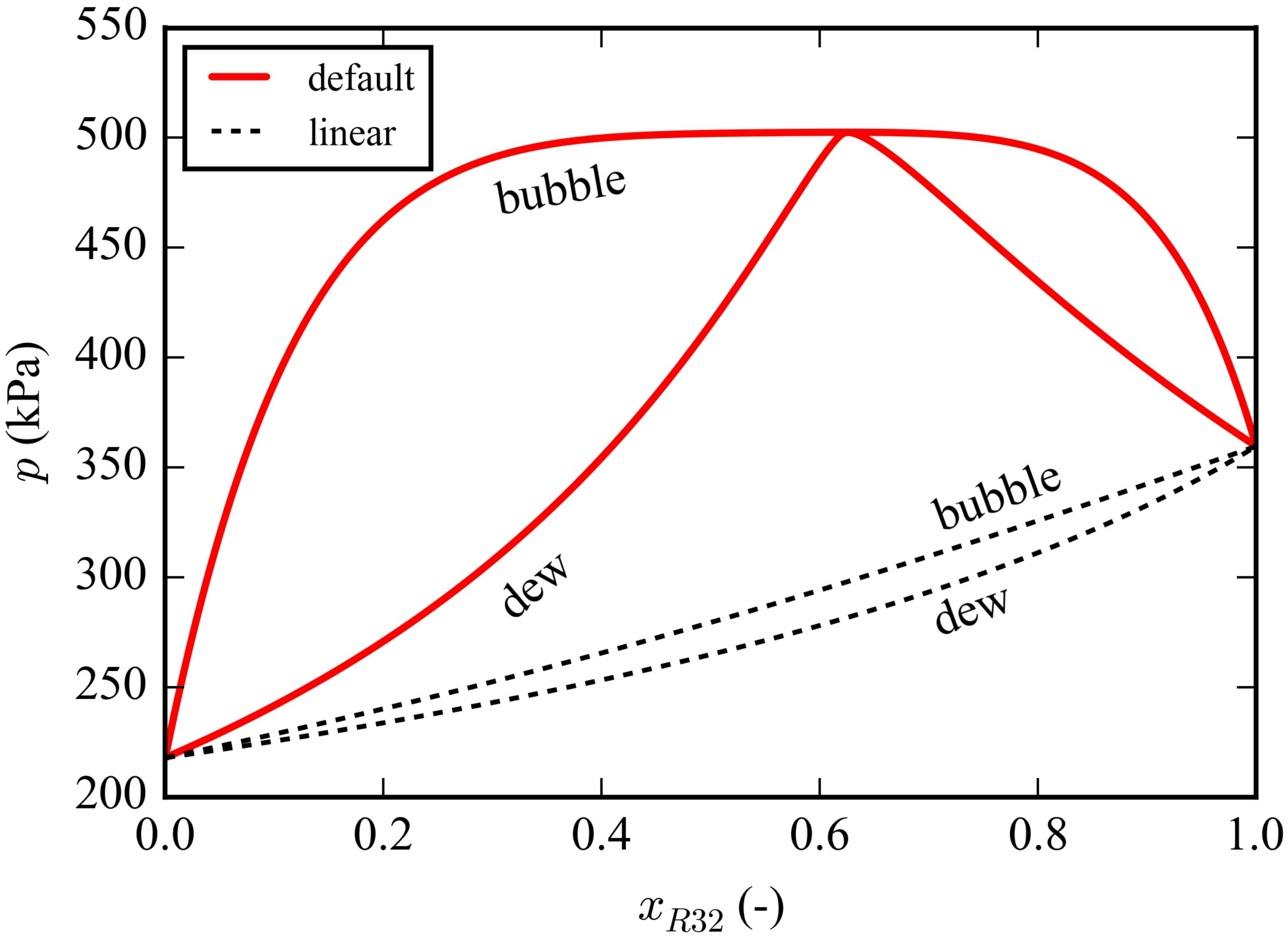
- Adjustable parameters: $\beta_{T,ij}, \gamma_{T,ij}, \beta_{v,ij}, \gamma_{v,ij}$ for ij pair
- Parameters are entirely empirical

Reducing functions

- Binary interaction parameter selection
- What β, γ should I use?

    1. Fitted parameters from literature
    2. Estimation schemes (**WARNING!!**)
    3. Simple mixing rules (linear, Lorentz-Berthelot) (**WARNING!!**)

Simple mixing rules



R32-Propane p-x plot at 250 K

Fit your own parameters!

- Work developed at NIST
- Interaction parameters fit for more than 1000 mixtures
- Fully-automatic fitting of parameters $\beta_{T,ij}$ and $\gamma_{T,ij}$
- Open-source formulation using python and DEAP, powered by REFPROP
- For source code, email ian.bell@nist.gov

Vapor-liquid equilibria

- Complex and quite challenging !!
- Equate fugacities and moles of each component
- Obtaining initial guess for composition particularly challenging

- Common types of calculations:

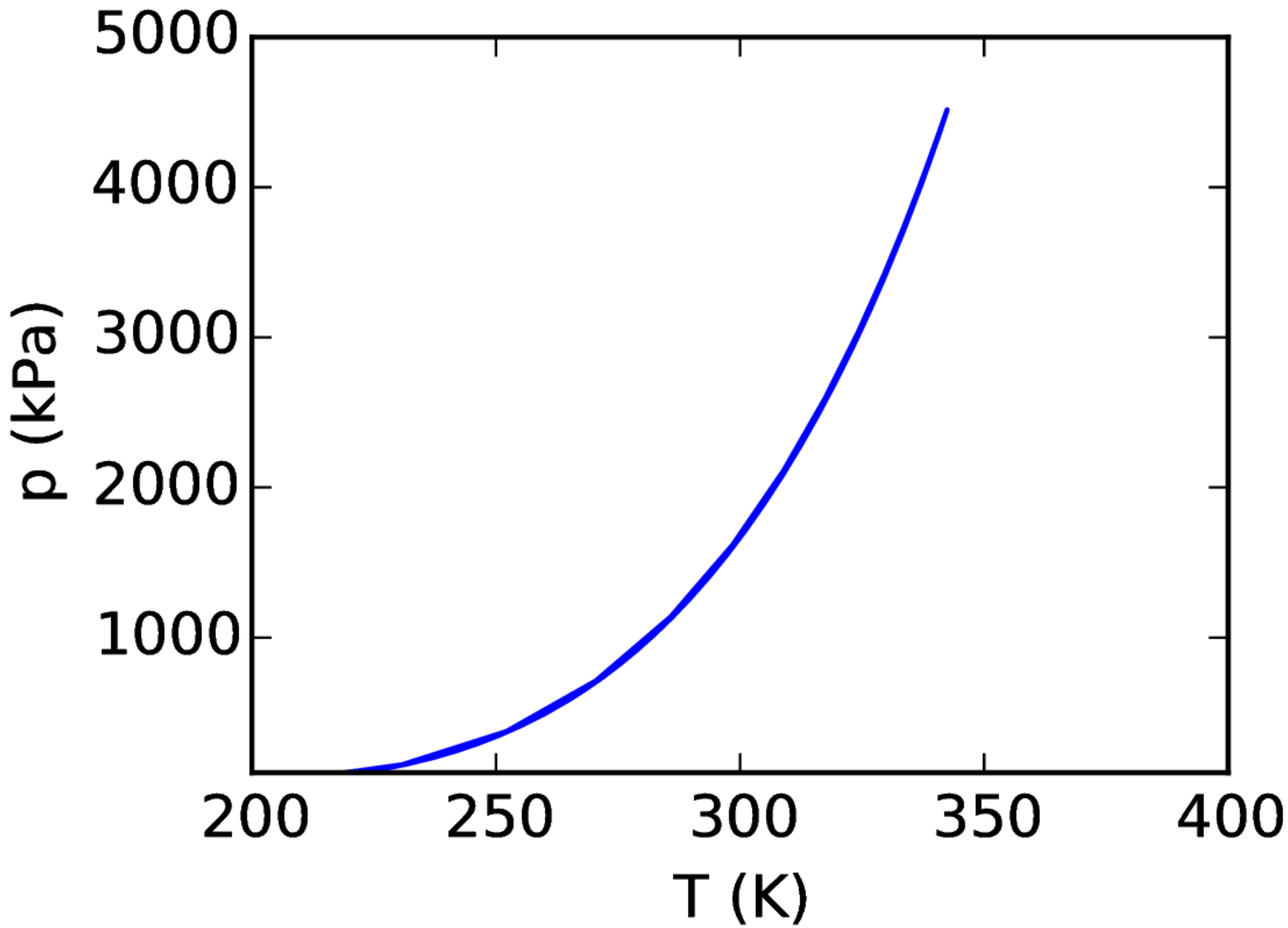
  - PQ: $p, \mathbf{x}$ at saturation $\rightarrow T, \mathbf{y}$
  - PQ: $p, \mathbf{y}$ at saturation $\rightarrow T, \mathbf{x}$
  - TQ: $T, \mathbf{x}$ at saturation $\rightarrow p, \mathbf{y}$
  - TQ: $T, \mathbf{y}$ at saturation $\rightarrow p, \mathbf{x}$
  - PT: $T, p \rightarrow \mathbf{x}, \mathbf{y}$

- Often, we "solve" these problems by constructing phase envelopes and interpolating

Phase envelope

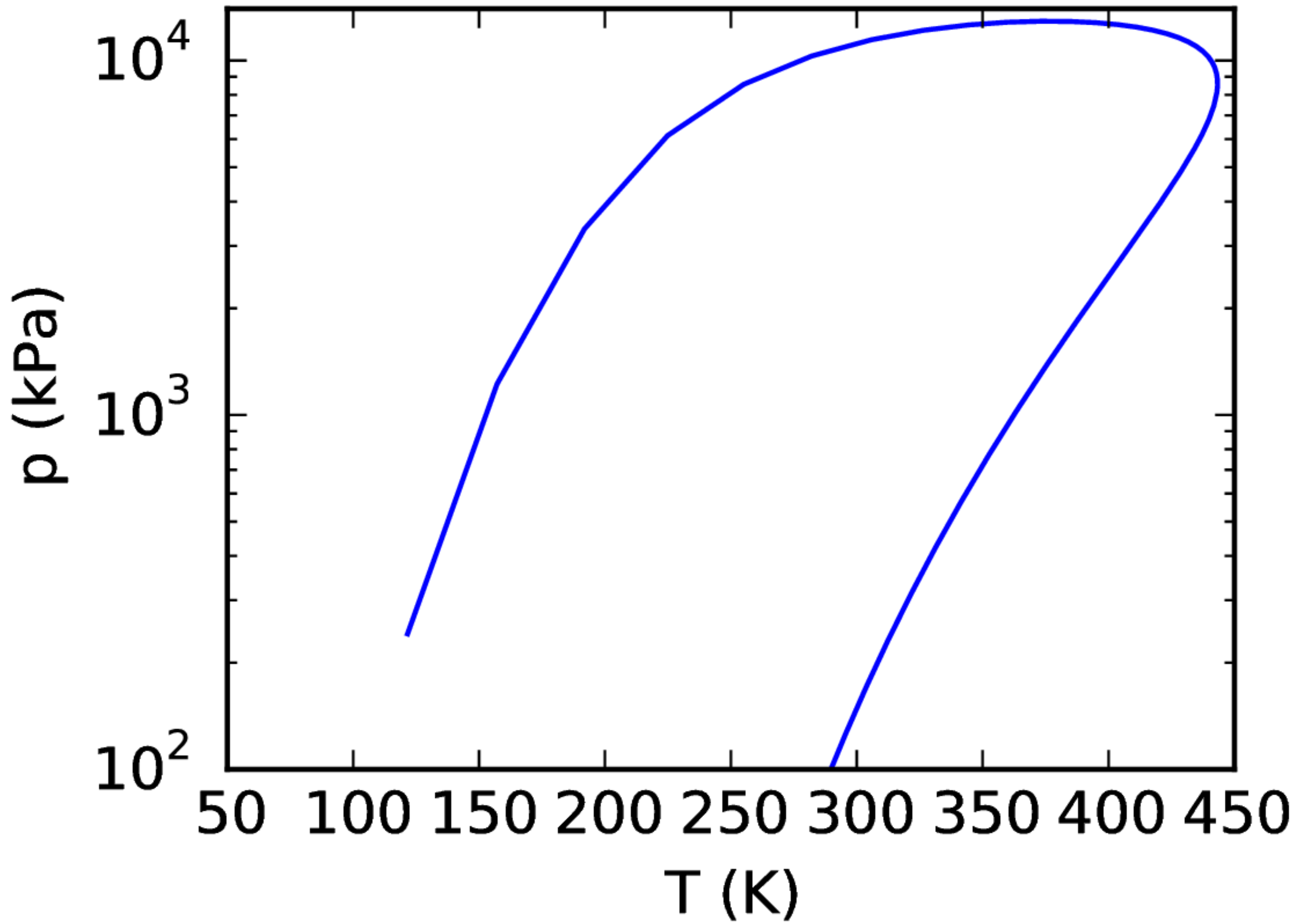
- What is a phase envelope?
    - The "vapor pressure" curve for a mixture
- For a "mixture" that is actually a pure fluid, it is a single line

Closed phase envelope of near-azeotropic mixture



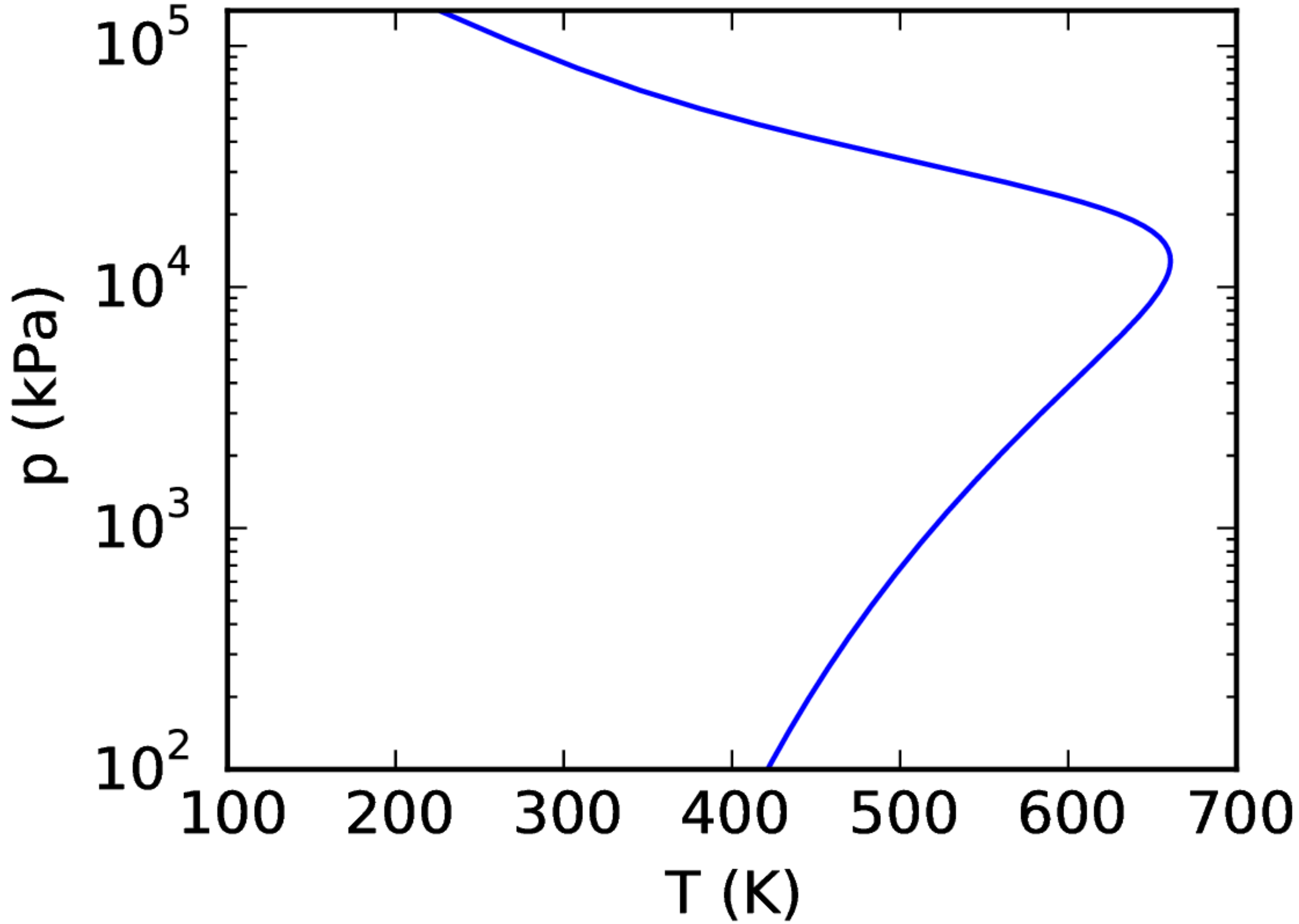
50/50 molar R32-R125 phase envelope

Closed phase envelope of large-temperature-glide mixture



50/50 molar pentane-methane phase envelope

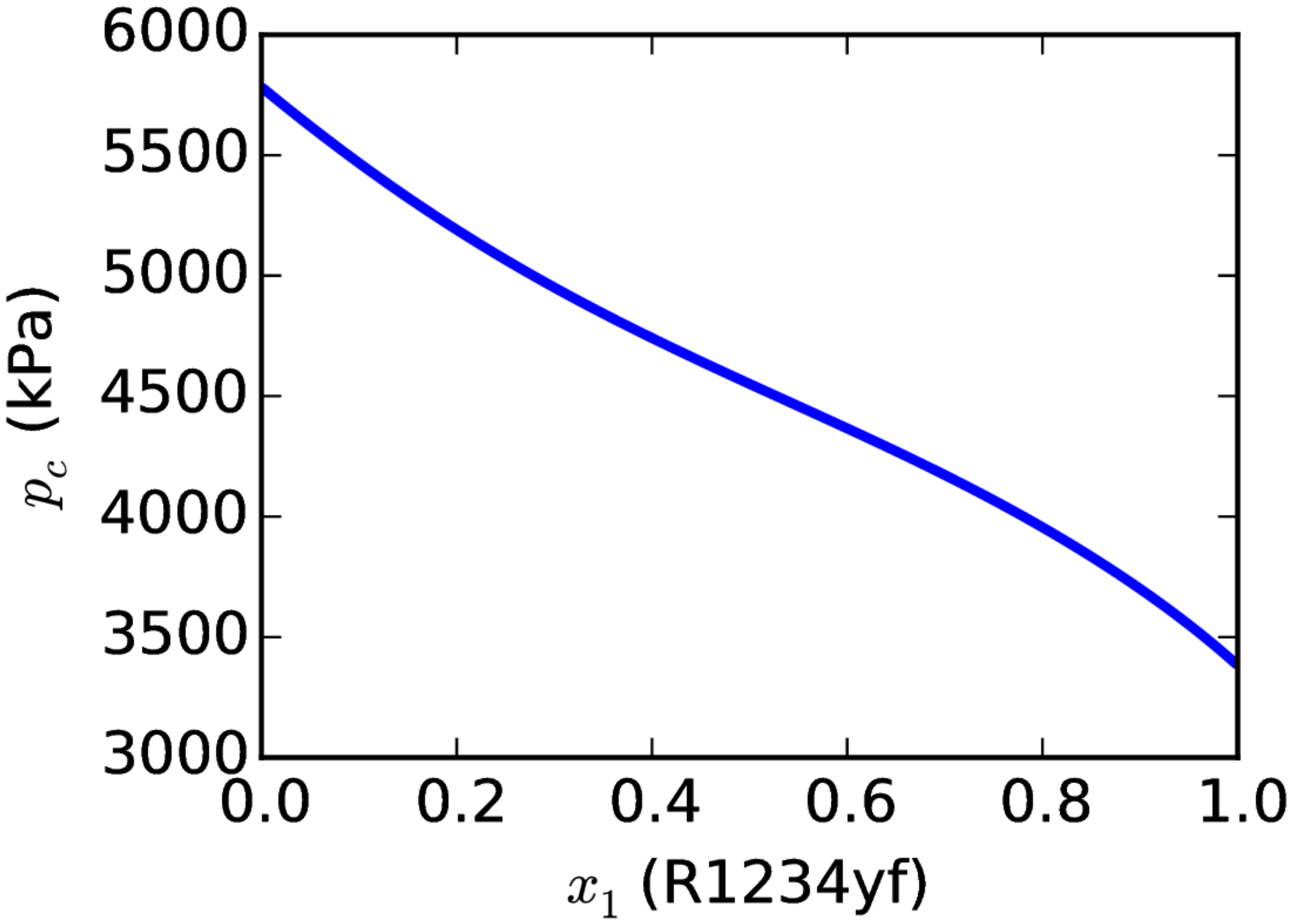
Open phase envelope



50/50 molar nitrogen-decane phase envelope

Critical lines

- Pure fluid: one critical point
- Mixture: critical lines



Critical pressure for R-1234yf + R-32 mixture

Thermophysical Property Libraries

Quite a lot of options, depending on your needs:

- REFPROP (NIST)
- CoolProp
- TREND (Bochum, Dresden)
- ASPEN
- PRODE
- ...

State-of-the-art libraries

REFPROP

- *The* reference thermophysical property library
- Development currently proceeds at a slow pace $\longrightarrow$ STABLE
- Industry standard!

State-of-the-art libraries

CoolProp

- +: Open-source, free for all uses
- +: Most user-friendly interface (similar interface available in REFPROP)
- +: Tabular interpolation, incompressible fluids, brines, psychrometric properties
- -: Mixture flash routines not competitive with REFPROP

State-of-the-art libraries

TREND

- +: Good support on complex phase equilibria (hydrate and solid phases)
- -: Not very fast

Interfacing with REFPROP

Calling REFPROP

- Call REFPROP directly
    - Fastest option (especially in FORTRAN)
    - Interfacing with DLL
        - Things to worry about because of FORTRAN (name mangling, strings, etc.)
        - Verbose for simple problems
        - Inconsistent set of units

Calling REFPROP

- Call through CoolProp
    - Easy-to-use interface for REFPROP consistent with interface of CoolProp
    - Consistent base-SI units
    - Very easy-to-use wrappers for comprehensive range of target environments (C++, python, MATLAB, Excel, Java, ...) that are both simple to understand and introduce little computational overhead.
    - Input arguments described: http://www.coolprop.org/coolprop/HighLevelAPI.html#table-of-string-inputs-to-propssi-function (http://www.coolprop.org/coolprop/HighLevelAPI.html#table-of-string-inputs-to-propssi-function)

Calling REFPROP

- Call through ctypes-based interface for Python
  - https://github.com/usnistgov/REFPROP-wrappers (https://github.com/usnistgov/REFPROP-wrappers)
  - Developed by me
  - Also allows calling from MATLAB, Julia, etc.

- Order of operations
    - `SETUPdll` or `SETMIXdll`
    - `TPFLSHdll` (or others) to get T,ρ
- Or use the new `REFPROPdll` function

The first calculation - NBP of water

In [17]:
```
import CoolProp.CoolProp as CP
from CoolProp.CoolProp import PropsSI
PropsSI('T','P',101325,'Q',0,'REFPROP::Water')
```

Out[17]:
```
373.1242958476953
```

Remember: all units are base-SI

Excercise

- Calculate the vapor pressure curve of R125 from the triple-point to critical point

In [4]:
```
### To be filled in live ###
import CoolProp.CoolProp as CP
import numpy as np

Tt = CP.PropsSI('T_triple','REFPROP::R125')
Tc = CP.PropsSI('Tcrit','REFPROP::R125')
Ts = np.linspace(Tt, Tc-0.001, 4)
ps = CP.PropsSI('P','T',Ts,'Q',0,'REFPROP::R125')
print(ps)
```

```
[2.91404173e+03 1.17183940e+05 9.21353723e+05 3.61784354e+06]
```

Single-phase Derivatives

$$\left(\frac{\partial A}{\partial B}\right)_C = \frac{\left(\frac{\partial A}{\partial \tau}\right)_\delta\left(\frac{\partial C}{\partial \delta}\right)_\tau - \left(\frac{\partial A}{\partial \delta}\right)_\tau\left(\frac{\partial C}{\partial \tau}\right)_\delta}{\left(\frac{\partial B}{\partial \tau}\right)_\delta\left(\frac{\partial C}{\partial \delta}\right)_\tau - \left(\frac{\partial B}{\partial \delta}\right)_\tau\left(\frac{\partial C}{\partial \tau}\right)_\delta} = \frac{N}{D}$$

In [11]:

```
### To be filled in live c_p = ??; key for c_p is C, d(Hmass)/d(T)|P ###
print(CP.PropsSI('C','T',298,'P',101235,'Water'))
print(CP.PropsSI('d(Hmass)/d(T)|P','T',298,'P',101235,'Water'))
```

```
4181.377468575649
4181.377468575648
```

Mixtures: predefined, pseudo-pure

NBP of liquid R410A

- Mixtures can be modeled:
    - As a pure fluid (for R410A, R404A, SES36, etc.) - limited number, only single-phase
    - As a mixture with predefined composition - using mixture model but composition imposed
    - As a mixture with user-specified composition

In [12]:

```python
from CoolProp.CoolProp import PropsSI

# Treating the mixture as a pure fluid
print(PropsSI('T','P',101325,'Q',0,'REFPROP::R410A'))

# Using the mixture model with the "fixed" composition of R410A
print(PropsSI('T','P',101325,'Q',0,'REFPROP::R410A.mix'))

# Using the mixture model with the molar composition specified
print(PropsSI('T','P',101325,'Q',0,'REFPROP::R32[0.69761]&R125[0.30239]'))
```

```
221.7081279166191
221.70710547048648
221.70711662650234
```

REFPROP through CoolProp (low-level)

Why low-level interface ?

- Closer to the compiled code
- Less overhead (integer-values only)
- More caching of values, reduce duplication of flash routines

In [13]:

```python
# Preparation and imports
from __future__ import print_function
import CoolProp, CoolProp.CoolProp as CP, numpy as np
import matplotlib.pyplot as plt

# Print some information about the versions in use
print(CoolProp.__version__, CoolProp.__gitrevision__)
print(CP.get_global_param_string("REFPROP_version"))
```

```
6.4.0 fcc01ad6b9739346273713b59af306d5cd9b7d24
10.0
```

In [14]:

```
AS = CP.AbstractState('REFPROP','Water') # Single SETUPdll call!
AS.update(CP.DmassT_INPUTS, 1, 1000)
print(AS.p(),'Pa')
```

```
460975.82625354873 Pa
```

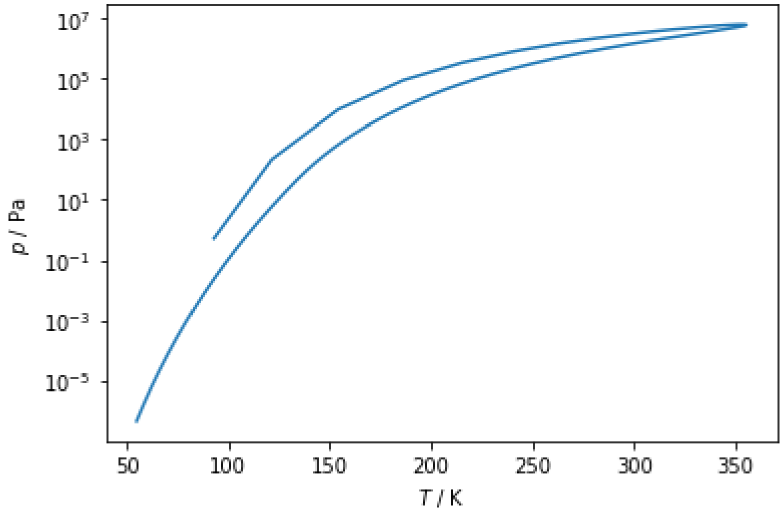
In [15]:

```
%timeit AS.update(CP.DmassT_INPUTS, 1, 1000)
%timeit AS.update(CP.PT_INPUTS, 101325, 300)
%timeit AS.update(CP.PT_INPUTS, 101325, 1000)
```

```
1.99 µs ± 260 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)
5.73 µs ± 238 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)
5.41 µs ± 377 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)
```

Phase envelope

In [18]:
```
AS = CP.AbstractState('REFPROP','CO2&Propane')
AS.set_mole_fractions([0.3,0.7])
AS.build_phase_envelope("")
PE = AS.get_phase_envelope_data()
plt.yscale('log')
plt.gca().set(xlabel='$T$ / K', ylabel='$p$ / Pa')
plt.plot(PE.T, PE.p);
```



Computational Speed Considerations

My code is too slow!!

Classical approaches

- Use the right independent variables
    - Reformulate problem in terms of density and temperature
    - Will in almost all cases be the fastest inputs
    - See for instance my dissertation, conversion from UM to DT in PDSim
- Simpler EOS
    - Use cubic EOS in place of Helmholtz-energy-based EOS
    - Much faster to evaluate

Classical approaches

- Skip phase evaluation
  - If inputs are known to be gas, don't check whether liquid/gas/supercritical
  - REFPROP: use PHFL1 instead of PHFLSH, etc.
  - CoolProp: call *specify_phase*
- Provide initial guess values (for $\rho, \mathbf{x}, \mathbf{y}$, etc.)
  - REFPROP
    - Some specialized flash routines like SATTP that use guesses, see also TPRHO
    - Many wrappers around REFPROP do not expose these specialized functions
  - CoolProp: call *update_with_guesses*

Classical approaches

- Ensure that you are limiting the number of setup calls
    - REFPROP: Don't call `SETUPdll` very often, put multiple components in a mixture and set mole fractions
    - CoolProp: Construct an instance of AbstractState, and then use it
- Minimize calling overhead
    - Use vectorized functions when possible

Bicubic interpolation

```
In [20]: import CoolProp
z = [0.5, 0.5]
BICU = CoolProp.AbstractState('BICUBIC&HEOS',
                              'R32&R125')
BICU.set_mole_fractions(z)
%timeit BICU.update(CoolProp.PQ_INPUTS, 101325, 0)

RP = CoolProp.AbstractState('HEOS','R32&R125')
RP.set_mole_fractions(z)
%timeit RP.update(CoolProp.PQ_INPUTS, 101325, 0)
```

```
1.09 µs ± 24.9 ns per loop (mean ± std. dev. of 7 runs, 1000000 loops each)
589 µs ± 14.5 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```

Thank You for your attention!

Questions?