

ROTEIRO PARA SOLUÇÃO DOS PROBLEMAS DE TERMODINÂMICA I e II

Prof. Emanuel Rocha Woiski
versão 0.3 02/11/2020

1. Dispenda um tempo lendo o problema atentamente e o interprete completamente.
2. Identifique os sistemas, subsistemas, as substâncias puras e as misturas envolvidas (homogêneas, heterogêneas, não reativas, reativas) e desenhe um esquema.
3. Analise as fronteiras dos sistemas e subsistemas, bem como as substâncias puras e misturas envolvidas e estabeleça as hipóteses necessárias.
4. Aplique a conservação de massa, a primeira lei e a segunda lei aos sistemas e subsistemas, identificando as invariantes universais.
5. Escreva o sistema de equações na forma funcional e explice as incógnitas. Se o número de incógnitas for igual ao número de equações linearmente independentes, o problema está resolvido.
6. Se o número de incógnitas é maior do que o número de equações, uma das possibilidades é obterem-se soluções para um relacionamento adequado entre as incógnitas. Outra possibilidade é tratarem-se como problemas de otimização de funções objetivo, que permitem também restrições de desigualdade.
6. Por fim, se necessário, resolva o sistema de equações numericamente, utilizando as ferramentas à sua disposição: tabelas, programas de computador, etc.

Variação da entalpia, entropia, função de Gibbs em função de T,P - expansão completa - equacionamento - 2020

December 27, 2020

Queremos obter $h = h(T, P)$, mas conhecemos apenas a equação de estado $v = v(T, P)$.

Pela regra da Cadeia:

$$dh = \left(\frac{\partial h}{\partial T} \right)_P (T, P) dT + \left(\frac{\partial h}{\partial P} \right)_T (T, P) dP$$

Pela definição do calor específico a pressão constante:

$$C_P(T, P) = \left(\frac{\partial h}{\partial T} \right)_P (T, P)$$

Pela Relação Fundamental na forma de Entalpia:

$$dh = T ds + v dP$$

Aplicando-se as derivadas na Relação Fundamental:

$$\left(\frac{\partial h}{\partial T} \right)_P = T \left(\frac{\partial s}{\partial T} \right)_P = C_P(T, P)$$

$$\left(\frac{\partial h}{\partial P} \right)_T = T \left(\frac{\partial s}{\partial P} \right)_T + v(T, P)$$

Pelas Relações de Maxwell:

$$\left(\frac{\partial s}{\partial P} \right)_T (T, P) = - \left(\frac{\partial v}{\partial T} \right)_P (T, P)$$

Juntando-se tudo:

$$dh = C_P(T, P) dT + \left\{ v(T, P) - T \left(\frac{\partial v}{\partial T} \right)_P (T, P) \right\} dP$$

Integrando-se entre o estado 1 a $(T^{(1)}, P^{(1)})$ e o estado 2 a $(T^{(2)}, P^{(2)})$:

$$\int_{(1)}^{(2)} dh = h(T^{(2)}, P^{(2)}) - h(T^{(1)}, P^{(1)}) = \int_{T^{(1)}}^{T^{(2)}} C_P(T, P) dT + \int_{P^{(1)}}^{P^{(2)}} \left\{ v(T, P) - T \frac{\partial v}{\partial T} \right\}_P (T, P) dP$$

O lado esquerdo não depende de processo, nem sequer de caminho de integração. Já as integrais do lado direito dependem cada qual do caminho de integração, pois são integrais de linha.

Vamos escolher um caminho isotérmico-isobárico-isotérmico nessa ordem:

$$h(T^{(2)}, P^{(2)}) - h(T^{(1)}, P^{(1)}) = \\ [h(T^{(2)}, P^{(2)}) - h(T^{(2)}, P^{(3)})] + [h(T^{(2)}, P^{(3)}) - h(T^{(1)}, P^{(3)})] + [h(T^{(1)}, P^{(3)}) - h(T^{(1)}, P^{(1)})]$$

Cada termo corresponde a um tipo de integral:

Isotérmica em $T^{(2)}$:

$$[h(T^{(2)}, P^{(2)}) - h(T^{(2)}, P^{(3)})] = \int_{P^{(3)}}^{P^{(2)}} \left\{ v(T^{(2)}, P) - T^{(2)} \frac{\partial v}{\partial T} \right\}_P (T^{(2)}, P) dP_{T^{(2)}}$$

Isobárica em $P^{(3)}$:

$$[h(T^{(2)}, P^{(3)}) - h(T^{(1)}, P^{(3)})] = \int_{T^{(1)}}^{T^{(2)}} C_P(T, P^{(3)}) dT_{P^{(3)}}$$

Isotérmica em $T^{(1)}$:

$$[h(T^{(1)}, P^{(3)}) - h(T^{(1)}, P^{(1)})] = \int_{P^{(1)}}^{P^{(3)}} \left\{ v(T^{(1)}, P) - T^{(1)} \frac{\partial v}{\partial T} \right\}_P (T^{(1)}, P) dP_{T^{(1)}}$$

Se $P^{(3)} = P^*$:

Isotérmica em $T^{(2)}$:

$$[h(T^{(2)}, P^{(2)}) - h(T^{(2)}, P^*)] = \int_{P^*}^{P^{(2)}} \left\{ v(T^{(2)}, P) - T^{(2)} \frac{\partial v}{\partial T} \right\}_P (T^{(2)}, P) dP_{T^{(2)}}$$

Isobárica em P^* :

$$[h(T^{(2)}, P^*) - h(T^{(1)}, P^*)] = \int_{T^{(1)}}^{T^{(2)}} C_P(T, P^*) dT_{P^*}$$

Isotérmica em $T^{(1)}$:

$$[h(T^{(1)}, P^*) - h(T^{(1)}, P^{(1)})] = \int_{P^{(1)}}^{P^*} \left\{ v(T^{(1)}, P) - T^{(1)} \frac{\partial v}{\partial T} \right\}_P (T^{(1)}, P) dP_{T^{(1)}}$$

Ou então, se os estados em P^* forem na região do gás perfeito:

Isotérmica em $T^{(2)}$:

$$[h(T^{(2)}, P^{(2)}) - h^*(T^{(2)})] = \int_0^{P^{(2)}} \left\{ v(T^{(2)}, P) - T^{(2)} \frac{\partial v}{\partial T} \right\}_P (T^{(2)}, P) dP_{T^{(2)}}$$

Isobárica em P^* :

$$[h^*(T^{(2)}) - h^*(T^{(1)})] = \int_{T^{(1)}}^{T^{(2)}} C_{Po}(T) dT$$

Isotérmica em $T^{(1)}$:

$$[h^*(T^{(1)}) - h(T^{(1)}, P^{(1)})] = \int_{P^{(1)}}^0 \left\{ v(T^{(1)}, P) - T^{(1)} \frac{\partial v}{\partial T} \right\}_P (T^{(1)}, P) dP_{T^{(1)}}$$

Vamos definir o *desvio de entalpia* ou *entalpia residual*, que representa a diferença entre a entalpia se fosse gás perfeito em (T, P) , que é dada por $h^*(T)$ e a entalpia do estado real a (T, P) , cujo valor é $h(T, P)$:

$$(h^* - h)(T, P) = \int_0^P \left\{ v(T, P) - T \frac{\partial v}{\partial T} \right\}_P (T, P) dP_T$$

Com essas definições:

$$h(T^{(2)}, P^{(2)}) - h(T^{(1)}, P^{(1)}) = (h^* - h)(T^{(1)}, P^{(1)}) - (h^* - h)(T^{(2)}, P^{(2)}) + [h^*(T^{(2)}) - h^*(T^{(1)})]$$

Para a entropia, queremos conhecer $s = s(T, P)$.

Pela Regra da Cadeia:

$$ds = \left. \frac{\partial s}{\partial T} \right)_P (T, P) dT + \left. \frac{\partial s}{\partial P} \right)_T (T, P) dP$$

Substituindo-se:

$$ds = \frac{C_P(T, P)}{T} dT - \left. \frac{\partial v}{\partial T} \right)_P (T, P) dP$$

Integrando-se entre os estados 1 a $(T^{(1)}, P^{(1)})$ e 2 a $(T^{(2)}, P^{(2)})$:

$$\int_{(1)}^{(2)} ds = s(T^{(2)}, P^{(2)}) - s(T^{(1)}, P^{(1)}) = \int_{T^{(1)}}^{T^{(2)}} \frac{C_P(T, P)}{T} dT - \int_{P^{(1)}}^{P^{(2)}} \left. \frac{\partial v}{\partial T} \right)_P (T, P) dP$$

Usamos o mesmo caminho isotérmico-isobárico-isotérmico:

$$s(T^{(2)}, P^{(2)}) - s(T^{(1)}, P^{(1)}) = \\ [s(T^{(2)}, P^{(2)}) - s(T^{(2)}, P^*)] + [s(T^{(2)}, P^*) - s(T^{(1)}, P^*)] + [s(T^{(1)}, P^*) - s(T^{(1)}, P^{(1)})]$$

A isotérmica em $T^{(2)}$:

$$[s(T^{(2)}, P^{(2)}) - s(T^{(2)}, P^*)] = \\ - \int_{P^*}^{P^{(2)}} \left. \frac{\partial v}{\partial T} \right)_P (T^{(2)}, P) dP_{T^{(2)}}$$

A isobárica em P^* :

$$[s(T^{(2)}, P^*) - s(T^{(1)}, P^*)] = \int_{T^{(1)}}^{T^{(2)}} \frac{C_P(T, P^*)}{T} dT$$

A isotérmica em $T^{(1)}$:

$$[s(T^{(1)}, P^*) - s(T^{(1)}, P^{(1)})] = \\ - \int_{P^{(1)}}^{P^*} \left. \frac{\partial v}{\partial T} \right)_P (T^{(1)}, P) dP_{T^{(1)}}$$

Entretanto, o limite do integrando quando $P^* \rightarrow 0$ torna-se ilimitado:

$$\lim_{P^* \rightarrow 0} \left. \frac{\partial v}{\partial T} \right)_P (T, P) dP = \frac{RT}{P^*} \rightarrow \infty$$

Para resolver o problema e se livrar de P^* , vamos escrever as seguintes expressões válidas convenientes para as isotérmicas em $T^{(2)}$ e em $T^{(1)}$ respectivamente:

A isotérmica em $T^{(2)}$:

$$[s(T^{(2)}, P^{(2)}) - s(T^{(2)}, P^*)] = \\ [s(T^{(2)}, P^{(2)}) - s^*(T^{(2)}, P^{(2)})] + [s^*(T^{(2)}, P^{(2)}) - s(T^{(2)}, P^*)] = \\ [s(T^{(2)}, P^{(2)}) - s^*(T^{(2)}, P^{(2)})] - \int_{P^*}^{P^{(2)}} \frac{R}{P} dP$$

A isotérmica em $T^{(1)}$:

$$[s(T^{(1)}, P^*) - s(T^{(1)}, P^{(1)})] =$$

$$\begin{aligned} & \left[s(T^{(1)}, P^*) - s^*(T^{(1)}, P^{(1)}) \right] + \left[s^*(T^{(1)}, P^{(1)}) - s(T^{(1)}, P^{(1)}) \right] = \\ & - \int_{P^{(1)}}^{P^*} \frac{R}{P} dP + \left[s^*(T^{(1)}, P^{(1)}) - s(T^{(1)}, P^{(1)}) \right] \end{aligned}$$

Então, substituindo-se na expressão da variação de entropia entre 1 e 2:

$$\begin{aligned} & s(T^{(2)}, P^{(2)}) - s(T^{(1)}, P^{(1)}) = \\ & \left[s(T^{(2)}, P^{(2)}) - s^*(T^{(2)}, P^{(2)}) \right] - \int_{P^*}^{P^{(2)}} \frac{R}{P} dP - \int_{P^{(1)}}^{P^*} \frac{R}{P} dP + \\ & \left[s^*(T^{(1)}, P^{(1)}) - s(T^{(1)}, P^{(1)}) \right] + \left[s(T^{(2)}, P^*) - s(T^{(1)}, P^*) \right] \end{aligned}$$

Vamos definir o *desvio de entropia* ou *entropia residual*, que representa a diferença entre a entropia se fosse gás perfeito em (T, P) , que é dada por $s^*(T, P)$ e a entropia do estado real a (T, P) , cujo valor é $s(T, P)$. Observe que o integrando se anula quando $P^* \rightarrow 0$:

$$(s^* - s)(T, P) = \int_0^P \left\{ \left(\frac{\partial v}{\partial T} \right)_P (T, P) - \frac{R}{P} \right\} dP$$

Então:

$$\begin{aligned} & s(T^{(2)}, P^{(2)}) - s(T^{(1)}, P^{(1)}) = \\ & -(s^* - s)(T^{(2)}, P^{(2)}) + (s^* - s)(T^{(1)}, P^{(1)}) + \int_{T^{(1)}}^{T^{(2)}} \frac{C_{Po}(T)}{T} dT - R \ln \left(\frac{P^{(2)}}{P^{(1)}} \right) \end{aligned}$$

Ou:

$$\begin{aligned} & s(T^{(2)}, P^{(2)}) - s(T^{(1)}, P^{(1)}) = \\ & -(s^* - s)(T^{(2)}, P^{(2)}) + (s^* - s)(T^{(1)}, P^{(1)}) + s^*(T^{(2)}, P^{(2)}) - s^*(T^{(1)}, P^{(1)}) \end{aligned}$$

Embora o arrazoado tenha sido muito mais complexo devido a dependência da entropia de T e P, a expansão para a entropia resultou em uma expressão muito parecida com a expansão completa da variação da entalpia obtida anteriormente.

O procedimento básico, portanto, agora deve estar claro: Assume-se a variação da entalpia (ou da entropia) entre os estados dados como se fosse de gases perfeitos naqueles estados e corrigir-se para cada estado, usando-se os respectivos desvios de entalpia (ou de entropia).

Como se obtém os desvios de entalpia ou entropia? Para isso, tínhamos assumido lá no início o conhecimento da equação de estado explícita em v, tal que $v = v(T, P) = Z(T, P)RT/P$. Dessa forma,

$$\left(\frac{\partial v}{\partial T} \right)_P (T, P) = \frac{R}{P} \left[Z(T, P) + T \left(\frac{\partial Z}{\partial T} \right)_P (T, P) \right]$$

O desvio de entalpia se torna:

$$(h^* - h)(T, P) = - \int_P^0 \left\{ v(T, P) - T \frac{\partial v}{\partial T} \right\}_P (T, P) dP_T = RT^2 \int_0^P \frac{1}{P} \frac{\partial Z}{\partial T} \Big|_P (T, P) dP_T$$

E o desvio de entropia:

$$(s^* - s)(T, P) = \int_0^P \left\{ \frac{\partial v}{\partial T} \Big|_P (T, P) - \frac{R}{P} \right\} dP = R \int_0^P \frac{1}{P} \left[Z(T, P) + T \frac{\partial Z}{\partial T} \Big|_P (T, P) - 1 \right] dP_T$$

Seja a assinatura da substância pura (M, T_c, P_c, ω, Y) .

Então: $T_r = T/T_c$, $\$P_r = P/P_c$ e $Z = Z(T_r, P_r, \omega, Y)$ e podemos adimensionalizar a entalpia residual e a entropia residual, que serão funções de (T_r, P_r, ω, Y) :

$$\begin{aligned} \left[\frac{(h^* - h)}{RT_c} \right] (T_r, P_r, \omega, Y) &= T_r^2 \int_0^{P_r} \frac{1}{P_r} \frac{\partial Z}{\partial T_r} \Big|_{P_r} (T_r, P_r, \omega, Y) dP_{rT_r} \\ \left[\frac{(s^* - s)}{R} \right] (T_r, P_r, \omega, Y) &= \\ \int_0^{P_r} \frac{1}{P_r} \left[Z(T_r, P_r, \omega, Y) + T_r \frac{\partial Z}{\partial T_r} \Big|_{P_r} (T_r, P_r, \omega, Y) - 1 \right] dP_{rT_r} \end{aligned}$$

A fugacidade de uma substância pura é definida a partir da Reação Fundamental na forma de Função de Gibbs:

$$dg = -sdT + vdP$$

Pela definição de fugacidade f :

$$dg_T = v(T, P)dP_T = Z(T, P)RT \frac{dP_T}{P} = RT d\ln f_T$$

Uma vez que $g = h - Ts$ e $g^* = h^* - Ts^*$, o desvio de função de Gibbs ou função de Gibbs residual ficará:

$$(g^* - g)(T, P) = -RT \ln \left(\frac{f}{P} \right) (T, P) = (h^* - h)(T, P) - T(s^* - s)(T, P)$$

Na forma adimensional, como função de (T_r, P_r, ω, Y) :

$$\left[\frac{(g^* - g)}{RT_c} \right] (T_r, P_r, \omega, Y) =$$

$$-T_r \ln \left(\frac{f}{P} \right) (T_r, P_r, \omega, Y) = \left[\frac{(h^* - h)}{RT_c} \right] (T_r, P_r, \omega, Y) - T_r \left[\frac{(s^* - s)}{R} \right] (T_r, P_r, \omega, Y)$$

Se $\omega = Y = 0$, ou seja, se for *substância simples* (e somente nesse caso), você tem a sua disposição as Tabelas A15.1-5 e as Figuras A.7-10 do VW 4a. Caso contrário, terá que lançar mão de programas de computador, como o LK_proptermo.

[]:

Estados adimensionais - substâncias simples - saturação tab A15-1

VW 4a - 2020

December 26, 2020

```
[1]: %matplotlib inline
from LK_proptermo import *
```

```
[2]: #Parte da assinatura da substância pura: (M, Tc, Pc, w, Y)
w = 0. # fator acêntrico de Pitzer
Y = 0. # fator de polaridade de Wu-Stiel
```

```
[3]: e = lk.Nondim_State(Tr=0.48,x=0.5,w=w,Y=Y).prop
```

```
[4]: e # estado adimensional (Tr,Pr,w,Y)
```

```
[4]: {'Tr': 0.48,
      'Pr': 0.0028821303534695662,
      'hl': 5.525525436132204,
      'hv': 0.01190966666643719,
      'h': 2.7687175513993205,
      'u': 2.526858685379786,
      'ul': 5.0458187436692254,
      'uv': 0.007898627090347447,
      'sl': 11.503185751796925,
      'sv': 0.016486232076478226,
      's': 5.7598359919367015,
      'x': 0.5,
      'Zl': 0.0006110573687934896,
      'Zv': 0.991643667549813,
      'Z': 0.4961273624593032,
      'fase': 'saturated mixture'}
```

```
[5]: np.log(e['Pr'])
```

```
[5]: -5.849225552216154
```

```
[6]: def Zsat(Tr,w,Y):
    e = lk.Nondim_State(Tr=Tr,x=0,w=w,Y=Y).prop
    cf = -1/Tr*e['hv'] + e['sv']
    return ( np.log(e['Pr']),e['Pr'],e['Zl'],
```

```

e['Zv'],e['hl'],e['hv'],
e['sl'],e['sv'],cf)

[7]: def Z(Tr,Pr,w,Y):
    e = lk.Nondim_State(Tr=Tr,Pr=Pr,w=w,Y=Y).prop
    return e['Z']

[8]: Zsat(0.4,w,Y)

[8]: (-8.211927080504267,
       0.0002713972127523896,
       6.476995101700225e-05,
       0.9986499093507051,
       5.763669788908394,
       0.0016325705837251959,
       14.40782519822066,
       0.002732150223933796,
       -0.0013492762353791935)

[9]: Trs = np.r_[0.30:0.38:5j,.40:1:0.02,0.98,0.99,0.995,1.]#
      Trs

[9]: array([0.3 , 0.32 , 0.34 , 0.36 , 0.38 , 0.4 , 0.42 , 0.44 , 0.46 ,
          0.48 , 0.5 , 0.52 , 0.54 , 0.56 , 0.58 , 0.6 , 0.62 , 0.64 ,
          0.66 , 0.68 , 0.7 , 0.72 , 0.74 , 0.76 , 0.78 , 0.8 , 0.82 ,
          0.84 , 0.86 , 0.88 , 0.9 , 0.92 , 0.94 , 0.96 , 0.98 , 0.98 ,
          0.99 , 0.995, 1.   ])

[10]: sat = {str(Tr)[:8]:(Tr,*Zsat(Tr,w,Y)) for Tr in Trs}
      #sat

[11]: print ('\nTABELA A.15.1 SATURAÇÃO substância simples (w=Y=0) \n'
           'VW 4A edição pág. 552\n')
      print('Tr\t lnPr\t Pr\t Zl\t Zv\t hl\t hv\t sl\t sv\t cf')
      print('*'*100)
      for v in sat.values():
          Tr,lnPr,Pr,Zl,Zv,hl,hv,sl,sv,cf = v
          print(f'{Tr:.4g}\t{lnPr:.4g}\t{Pr:.4g}\t{Zl:.4g}\t{Zv:.4g}\t',
                f'{hl:.4g}\t{hv:.4g}\t{sl:.4g}\t{sv:.4g}\t{cf:.4g}')
      print ('\n' FIM ' ')

```

TABELA A.15.1 SATURAÇÃO substância simples (w=Y=0)
 VW 4A edição pág. 552

Tr	lnPr	Pr	Zl	Zv	hl	hv	sl	sv	cf
----	------	----	----	----	----	----	----	----	----

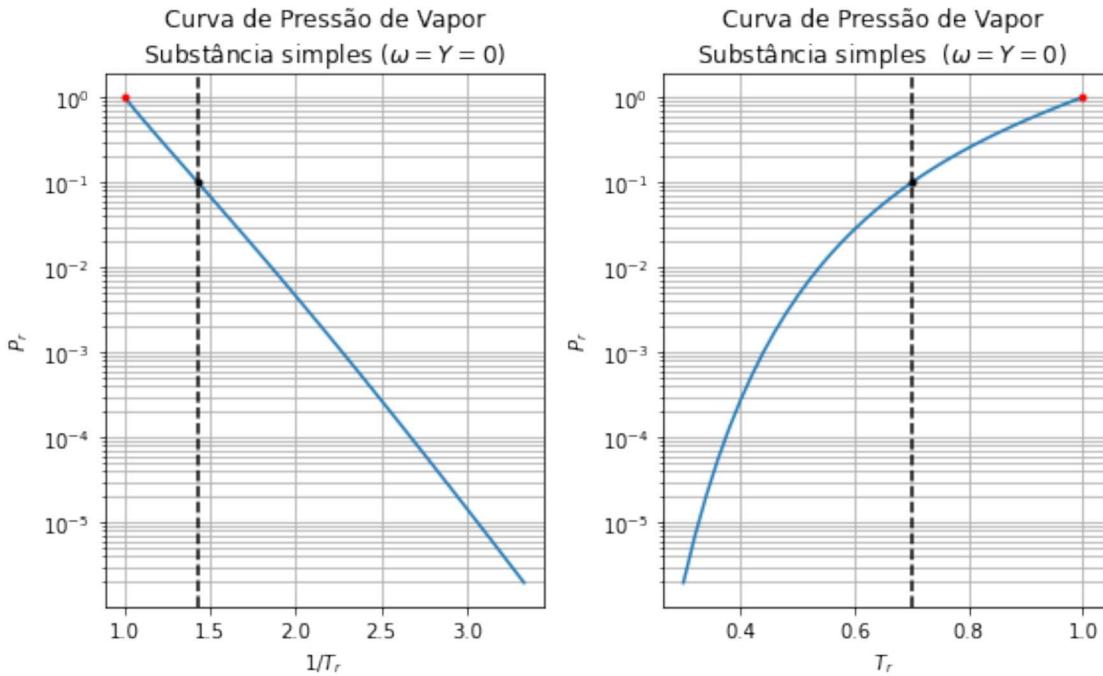
0.3	-13.15	1.952e-06	5.657e-07	1	6.046	2.207e-05		20.15	
5.008e-05		-2.35e-05							
0.32	-11.89	6.842e-06	1.897e-06	0.9999	5.991	6.691e-05			
18.72	0.0001418		-6.733e-05						
0.34	-10.8	2.048e-05	5.45e-06	0.9998	5.935	0.0001751			
17.46	0.0003478		-0.0001671						
0.36	-9.831	5.378e-05	1.377e-05	0.9996	5.879	0.0004057			
16.33	0.0007587		-0.0003682						
0.38	-8.975	0.0001266	3.125e-05	0.9993	5.822	0.0008493			
15.32	0.0015	-0.0007348							
0.4	-8.212	0.0002714	6.477e-05	0.9986	5.764	0.001633			
14.41	0.002732		-0.001349						
0.42	-7.528	0.0005377	0.0001243	0.9977	5.705	0.002919			
13.58	0.004641		-0.002309						
0.44	-6.912	0.0009953	0.0002231	0.9963	5.645	0.004906			
12.83	0.007431		-0.003719						
0.46	-6.355	0.001737	0.0003785	0.9943	5.586	0.00782	12.14		
0.01131	-0.00569								
0.48	-5.849	0.002882	0.0006111	0.9916	5.526	0.01191	11.5		
0.01649	-0.008326								
0.5	-5.387	0.004573	0.0009452	0.9882	5.465	0.01744	10.92		
0.02315	-0.01172								
0.52	-4.965	0.006981	0.001408	0.9839	5.405	0.02467	10.38		
0.03148	-0.01596								
0.54	-4.576	0.0103	0.002031	0.9787	5.345	0.03388	9.877	0.04163	
-0.02112									
0.56	-4.217	0.01474	0.002845	0.9724	5.284	0.04534	9.409	0.05374	
-0.02723									
0.58	-3.886	0.02053	0.003886	0.9651	5.224	0.05932	8.973	0.06793	
-0.03434									
0.6	-3.578	0.02794	0.00519	0.9567	5.163	0.07606	8.563	0.08431	-0.04246
0.62	-3.291	0.03722	0.006795	0.9472	5.103	0.09584	8.178	0.103	
-0.0516									
0.64	-3.023	0.04866	0.00874	0.9365	5.041	0.1189	7.815	0.1241	-0.06175
0.66	-2.772	0.06253	0.01107	0.9246	4.979	0.1455	7.472	0.1476	-0.07289
0.68	-2.537	0.07914	0.01382	0.9115	4.917	0.176	7.146	0.1738	-0.08501
0.7	-2.315	0.09879	0.01704	0.8972	4.853	0.2106	6.835	0.2028	-0.09808
0.72	-2.106	0.12118	0.02077	0.8817	4.788	0.2497	6.538	0.2347	-0.1121
0.74	-1.908	0.1484	0.02508	0.8648	4.722	0.2936	6.254	0.2698	-0.1269
0.76	-1.72	0.1791	0.03002	0.8466	4.654	0.3428	5.98	0.3084	-0.1427
0.78	-1.542	0.214	0.03565	0.827	4.583	0.3978	5.716	0.3507	-0.1593
0.8	-1.372	0.2536	0.04205	0.8059	4.509	0.4592	5.46	0.3973	-0.1767
0.82	-1.21	0.2982	0.04932	0.7832	4.432	0.5278	5.21	0.4488	-0.1949
0.84	-1.055	0.3481	0.05757	0.7586	4.35	0.6047	4.965	0.506	-0.2139
0.86	-0.907	0.4037	0.06695	0.7318	4.263	0.6913	4.724	0.5701	-0.2337
0.88	-0.7646	0.4655	0.07766	0.7026	4.169	0.7894	4.483	0.6426	-0.2544

0.9	-0.6275	0.5339	0.08999	0.6703	4.066	0.9021	4.241	0.7262	-0.2761
0.92	-0.4952	0.6095	0.1044	0.6339	3.949	1.034	3.994	0.8248	-0.2987
0.94	-0.3671	0.6927	0.1216	0.5919	3.814	1.192	3.735	0.9455	-0.3225
0.96	-0.2426	0.7846	0.1433	0.541	3.647	1.393	3.451	1.103	-0.3479
0.98	-0.1207	0.8863	0.1741	0.4722	3.41	1.679	3.105	1.338	-0.3753
0.98	-0.1207	0.8863	0.1741	0.4722	3.41	1.679	3.105	1.338	-0.3753
0.99	-0.06039		0.9414	0.1988	0.4215	3.221	1.905	2.863	1.534
	-0.3902								
0.995	-0.03023		0.9702	0.2191	0.3843	3.067	2.08	2.684	1.692
	-0.3981								
1	7.01e-06		1	0.2833	0.2834	2.626	2.626	2.22	2.22
	-0.4064								

FIM

```
[12]: Prs = np.array([item[2] for item in sat.values()])
marker = '.'
plt.figure(figsize=(8,5))
plt.subplot(1,2,1)
plt.title('Curva de Pressão de Vapor\nSubstância simples $(\omega=Y=0)$')
plt.semilogy(1/Trs,Prs)
plt.semilogy([1/0.7],[sat['0.700000'][2]],'k'+marker)
plt.semilogy([1],[1],'r'+marker)
plt.axvline(x=1/0.7,color='k',linestyle='--')
plt.grid(which='both')
plt.xlabel('$1/T_r$')
plt.ylabel('$P_r$')

plt.subplot(1,2,2)
plt.title('Curva de Pressão de Vapor\nSubstância simples $(\omega=Y=0)$')
plt.semilogy(Trs,Prs)
plt.plot([0.7],[sat['0.700000'][2]],'k'+marker)
plt.plot([1],[1],'r'+marker)
plt.axvline(x=0.7,color='k',linestyle='--')
plt.grid(which='both')
plt.xlabel('$T_r$')
plt.ylabel('$P_r$')
plt.tight_layout();
```



```
[13]: Zls = np.array([item[3] for item in sat.values()])
Zvs = np.array([item[4] for item in sat.values()])

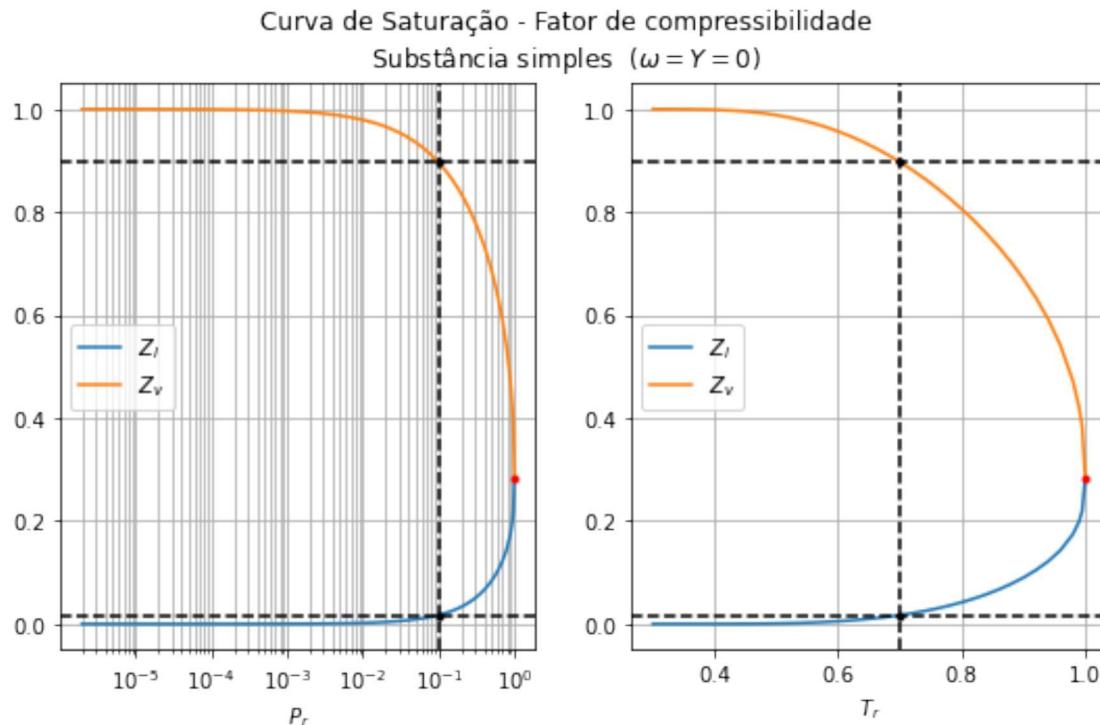
plt.figure(figsize=(9,5))
plt.subplot(1,2,1)
plt.suptitle('Curva de Saturação - Fator de compressibilidade\nSubstância simples $\omega=Y=0$')
plt.semilogx(Prs,Zls)
plt.semilogx(Prs,Zvs)
plt.semilogx([Prs[-1]],[Zvs[-1]],'r'+marker)
plt.semilogx([Zsat(0.7,w,Y)[1]],[Zsat(0.7,w,Y)[2]],'k'+marker)
plt.semilogx([Zsat(0.7,w,Y)[1]],[Zsat(0.7,w,Y)[3]],'k'+marker)
plt.axvline(x=Zsat(0.7,w,Y)[1],color='k',linestyle='--')
plt.axhline(y=Zsat(0.7,w,Y)[2],color='k',linestyle='--')
plt.axhline(y=Zsat(0.7,w,Y)[3],color='k',linestyle='--')
plt.grid(which='both')
plt.xlabel('$Z_l, Z_v$')
plt.xlabel('$P_r$')
plt.legend(['$Z_l$', '$Z_v$'],loc='best');

plt.subplot(1,2,2)
plt.plot(Trs,Zls)
plt.plot(Trs,Zvs)
plt.plot([1],[Zvs[-1]],'r'+marker)
plt.plot([0.7],[Zsat(0.7,w,Y)[2]],'k'+marker)
```

```

plt.plot([0.7],[Zsat(0.7,w,Y)[3]],'k'+marker)
plt.axvline(x=0.7,color='k',linestyle='--')
plt.axhline(y=Zsat(0.7,w,Y)[2],color='k',linestyle='--')
plt.axhline(y=Zsat(0.7,w,Y)[3],color='k',linestyle='--')
plt.grid(which='both')
plt.xlabel('$Z_l, Z_v$')
plt.xlabel('$T_r$')
plt.legend(['$Z_l$', '$Z_v$'],loc='best');

```



```

[14]: hls = np.array([item[5] for item in sat.values()])
hvs = np.array([item[6] for item in sat.values()])

plt.figure(figsize=(9,5))
plt.subplot(1,2,1)
plt.suptitle('Curva de Saturação - Desvio de entalpia adimensional\nSubstância simples $(\omega=Y=0)$')
plt.semilogx(Prs,hls)
plt.semilogx(Prs,hvs)
plt.semilogx([Prs[-1]],[hvs[-1]],'r'+marker)
plt.semilogx([Zsat(0.7,w,Y)[1]],[Zsat(0.7,w,Y)[4]],'k'+marker)
plt.semilogx([Zsat(0.7,w,Y)[1]],[Zsat(0.7,w,Y)[5]],'k'+marker)
plt.axvline(x=Zsat(0.7,w,Y)[1],color='k',linestyle='--')
plt.axhline(y=Zsat(0.7,w,Y)[4],color='k',linestyle='--')

```

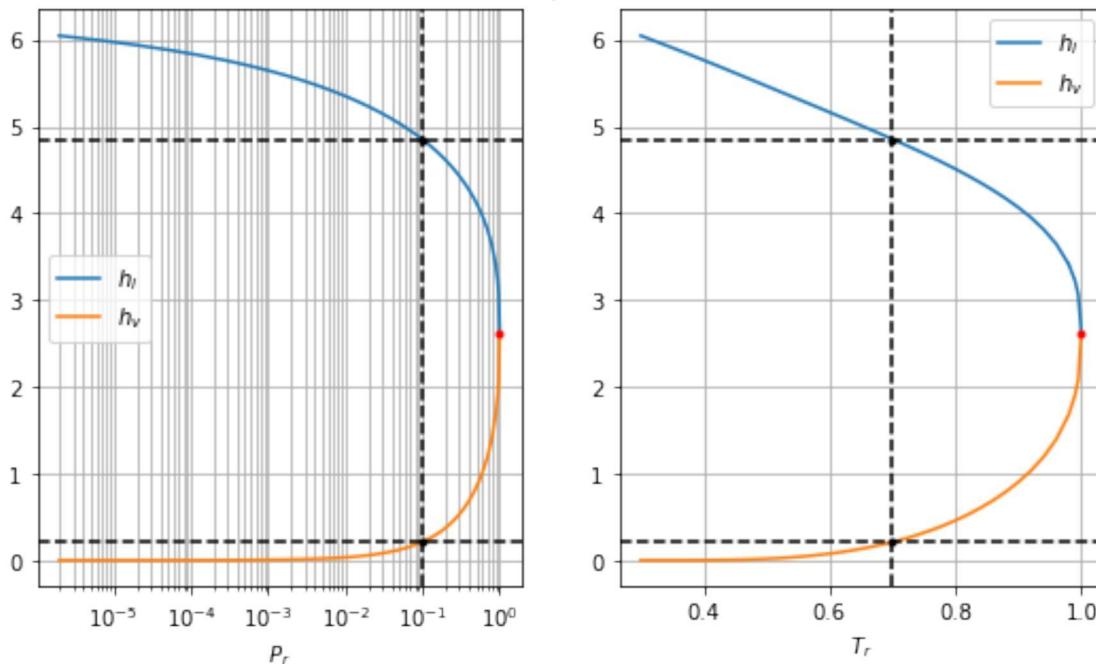
```

plt.axhline(y=Zsat(0.7,w,Y)[5],color='k',linestyle='--')
plt.grid(which='both')
plt.xlabel('$h_l,h_v$')
plt.xlabel('$P_r$')
plt.legend(['$h_l$', '$h_v$'],loc='best');

plt.subplot(1,2,2)
plt.plot(Trs,hls)
plt.plot(Trs,hvs)
plt.plot([1],[hvs[-1]],'r'+marker)
plt.plot([0.7],[Zsat(0.7,w,Y)[4]],'k'+marker)
plt.plot([0.7],[Zsat(0.7,w,Y)[5]],'k'+marker)
plt.axvline(x=0.7,color='k',linestyle='--')
plt.axhline(y=Zsat(0.7,w,Y)[4],color='k',linestyle='--')
plt.axhline(y=Zsat(0.7,w,Y)[5],color='k',linestyle='--')
plt.grid(which='both')
plt.xlabel('$h_l,h_v$')
plt.xlabel('$T_r$')
plt.legend(['$h_l$', '$h_v$'],loc='best');

```

Curva de Saturação - Desvio de entalpia adimensional
Substância simples ($\omega = Y = 0$)



```
[15]: sls = np.array([item[7] for item in sat.values()])
svs = np.array([item[8] for item in sat.values()])
```

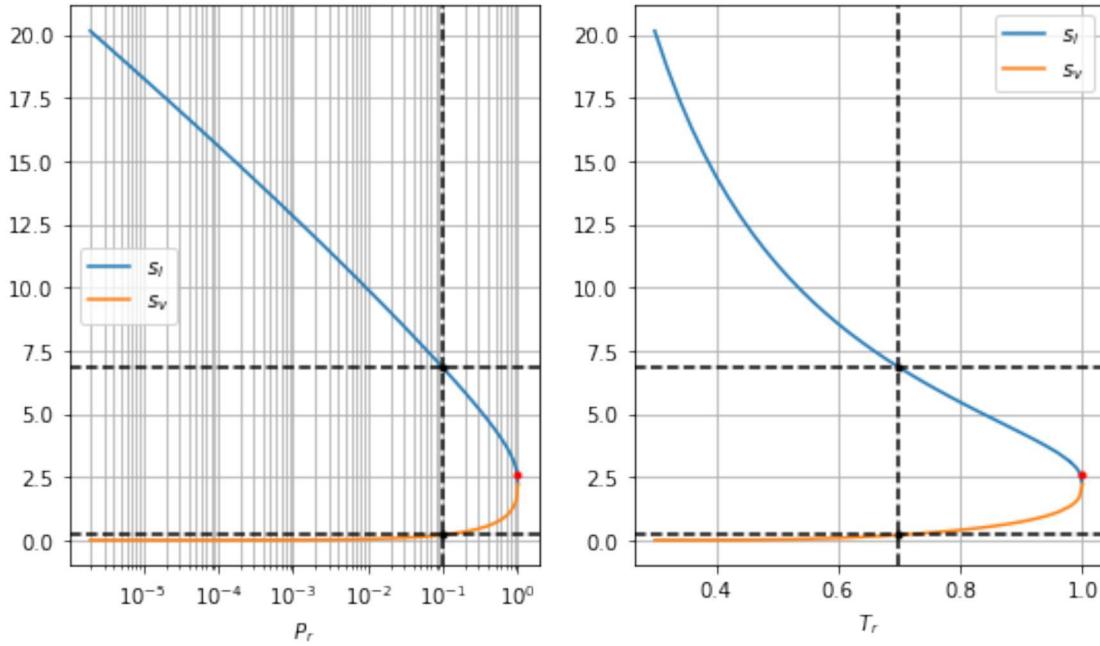
```

plt.figure(figsize=(9,5))
plt.subplot(1,2,1)
plt.suptitle('Curva de Saturação - Desvio de entropia adimensional\nSubstância_\n→simples $(\omega=Y=0)$')
plt.semilogx(Prs,sls)
plt.semilogx(Prs,svs)
plt.semilogx([Prs[-1]],[hvs[-1]],'r'+marker)
plt.semilogx([Zsat(0.7,w,Y)[1]],[Zsat(0.7,w,Y)[6]],'k'+marker)
plt.semilogx([Zsat(0.7,w,Y)[1]],[Zsat(0.7,w,Y)[7]],'k'+marker)
plt.axvline(x=Zsat(0.7,w,Y)[1],color='k',linestyle='--')
plt.axhline(y=Zsat(0.7,w,Y)[6],color='k',linestyle='--')
plt.axhline(y=Zsat(0.7,w,Y)[7],color='k',linestyle='--')
plt.grid(which='both')
# plt.ylabel('$s_l, s_v$')
plt.xlabel('$P_r$')
plt.legend(['$s_l$', '$s_v$'],loc='best');

plt.subplot(1,2,2)
plt.plot(Trs,sls)
plt.plot(Trs,svs)
plt.plot([1],[hvs[-1]],'r'+marker)
plt.plot([0.7],[Zsat(0.7,w,Y)[6]],'k'+marker)
plt.plot([0.7],[Zsat(0.7,w,Y)[7]],'k'+marker)
plt.axvline(x=0.7,color='k',linestyle='--')
plt.axhline(y=Zsat(0.7,w,Y)[6],color='k',linestyle='--')
plt.axhline(y=Zsat(0.7,w,Y)[7],color='k',linestyle='--')
plt.grid(which='both')
# plt.ylabel('$s_l, s_v$')
plt.xlabel('$T_r$')
plt.legend(['$s_l$', '$s_v$'],loc='best');

```

Curva de Saturação - Desvio de entropia adimensional
Substância simples ($\omega = Y = 0$)



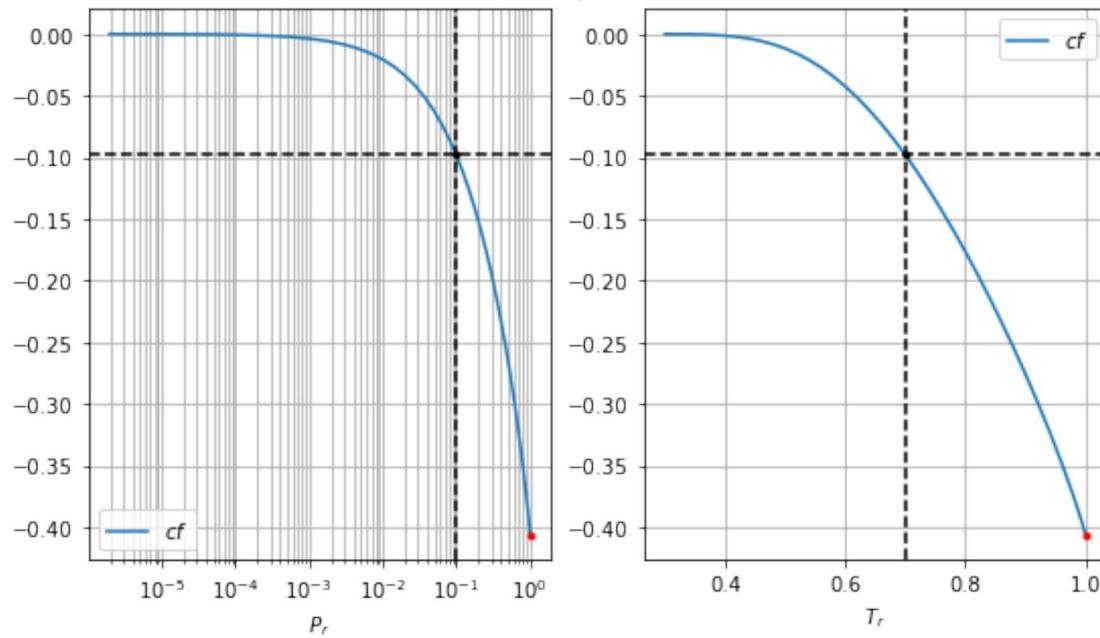
```
[16]: cfs = np.array([item[9] for item in sat.values()])

plt.figure(figsize=(9,5))
plt.subplot(1,2,1)
plt.suptitle('Curva de Saturação - Coeficiente de fugacidade\nSubstância simples $(\omega=Y=0)$')
plt.semilogx(Prs,cfs)
plt.semilogx([Prs[-1]],[cfs[-1]],'r'+marker)
plt.semilogx([Zsat(0.7,w,Y)[1]],[Zsat(0.7,w,Y)[8]],'k'+marker)
plt.axvline(x=Zsat(0.7,w,Y)[1],color='k',linestyle='--')
plt.axhline(y=Zsat(0.7,w,Y)[8],color='k',linestyle='--')
plt.grid(which='both')
plt.xlabel('$P_r$')
plt.legend(['$cf$'],loc='best');

plt.subplot(1,2,2)
plt.plot(Trs,cfs)
plt.plot([1],[cfs[-1]],'r'+marker)
plt.plot([0.7],[Zsat(0.7,w,Y)[8]],'k'+marker)
plt.axvline(x=0.7,color='k',linestyle='--')
plt.axhline(y=Zsat(0.7,w,Y)[8],color='k',linestyle='--')
plt.grid(which='both')
plt.xlabel('$T_r$')
```

```
plt.legend(['$cf$'], loc='best');
```

Curva de Saturação - Coeficiente de fugacidade
Substância simples ($\omega = Y = 0$)



[]:

Algumas Relações Termodinâmicas - não numérico - 2020

December 27, 2020

Algumas Relações Termodinâmicas Seja a definição do coeficiente de Joule-Thomson:

$$\mu_j = \left. \frac{\partial T}{\partial P} \right)_h$$

Aplicando-se a Regra Cíclica com $T - P - h$ nessa ordem:

$$\left. \frac{\partial T}{\partial P} \right)_h \left. \frac{\partial P}{\partial h} \right)_T \left. \frac{\partial h}{\partial T} \right)_P = -1$$

Isolando-se o μ_j :

$$\mu_j = \left. \frac{\partial T}{\partial P} \right)_h = - \left. \frac{\frac{\partial h}{\partial T}}{\frac{\partial h}{\partial P}} \right)_P = \frac{T \left. \frac{\partial v}{\partial T} \right)_P - v}{C_p} = \frac{v(\alpha_P T - 1)}{C_p}$$

Se for nos dada a equação de estado explícita em v , $v = v(T, P)$, então as derivadas

$$\left. \frac{\partial v}{\partial T} \right)_P = v\alpha_p$$

e

$$\left. \frac{\partial v}{\partial P} \right)_T = -v\beta_T$$

são cálculáveis. A única derivada $P - v - T$ que resta-nos determinar é

$$\left. \frac{\partial P}{\partial T} \right)_v$$

$$\left. \frac{\partial P}{\partial T} \right)_v = ?$$

$$\left. \frac{\partial P}{\partial T} \right)_v \left. \frac{\partial T}{\partial v} \right)_P \left. \frac{\partial v}{\partial P} \right)_T = -1$$

$$\left. \frac{\partial P}{\partial T} \right)_v = - \frac{\left. \frac{\partial v}{\partial T} \right)_P}{\left. \frac{\partial v}{\partial P} \right)_T} = - \frac{v \alpha_p}{-v \beta_T} = \frac{\alpha_p}{\beta_T}$$

O quadrado da velocidade do som c^2 em um meio não dispersivo é definido como: $\left. \frac{\partial P}{\partial \rho} \right)_s$.

$$c^2 = \left. \frac{\partial P}{\partial \rho} \right)_s$$

Mas:

$$\rho = \frac{1}{v}$$

Derivando-se:

$$d\rho = -\frac{dv}{v^2}$$

Substituindo:

$$c^2 = -v^2 \left. \frac{\partial P}{\partial v} \right)_s = \frac{v}{\beta_s}$$

Finalmente, a velocidade do som em meio não dispersivo fica:

$$c = \sqrt{\frac{v}{\beta_s}}$$

A compressibilidade adiabática ou isentrópica é definida como:

$$\beta_s = -\frac{1}{v} \left. \frac{\partial v}{\partial P} \right)_s$$

Infelizmente a compressibilidade isentrópica não é facilmente mensurável. Poderemos escrever em função de outras propriedades?

Pela Regra Cíclica $v - P - s$ nesta ordem:

$$\left. \frac{\partial v}{\partial P} \right)_s \left. \frac{\partial P}{\partial s} \right)_v \left. \frac{\partial s}{\partial v} \right)_P = -1$$

Explicitando-se a derivada de interesse:

$$\left. \frac{\partial v}{\partial P} \right)_s = -\frac{\left. \frac{\partial s}{\partial P} \right)_v}{\left. \frac{\partial s}{\partial v} \right)_P}$$

As derivadas parciais com P, v não se prestam para as Relações de Maxwell. Então, aplicando-se a Regra da Quarta Propriedade, com T como a quarta propriedade:

$$\left. \frac{\partial v}{\partial P} \right)_s = - \frac{\left. \frac{\partial s}{\partial T} \right)_v \left. \frac{\partial T}{\partial P} \right)_v}{\left. \frac{\partial s}{\partial T} \right)_P \left. \frac{\partial T}{\partial v} \right)_P} = - \frac{C_v}{T} \frac{\left. \frac{\partial T}{\partial P} \right)_v}{\frac{C_p}{T} \left. \frac{\partial T}{\partial v} \right)_P}$$

Levando-se em conta que

$$C_v = T \left. \frac{\partial s}{\partial T} \right)_v$$

,

$$C_p = T \left. \frac{\partial s}{\partial T} \right)_P$$

,

$$\alpha_p = \frac{1}{v} \left. \frac{\partial v}{\partial T} \right)_P$$

,

$$\left. \frac{\partial P}{\partial T} \right)_v = \frac{\alpha_p}{\beta_T}$$

,

podemos substituir e simplificar:

$$\left. \frac{\partial v}{\partial P} \right)_s = - \frac{C_v}{C_p} \frac{\frac{\beta_T}{\alpha_p}}{\frac{1}{v \alpha_p}} = -v \frac{C_v}{C_p} \beta_T$$

Finalmente, obtém-se para β_s :

$$\beta_s = - \frac{1}{v} \left. \frac{\partial v}{\partial P} \right)_s = \frac{C_v}{C_p} \beta_T$$

Uma expressão útil envolvendo os calores específicos é:

$$C_p - C_v = \frac{T v \alpha_p^2}{\beta_T} \geq 0$$

[]:

Relações Termodinâmicas - Exemplos - 2020

December 27, 2020

0.0.1 Exemplo 1

$$\left. \frac{\partial h}{\partial T} \right)_v = ?$$

Pela Relação Fundamental na forma de entalpia:

$$dh = Tds + vdP$$

Substituindo-se as derivadas parciais:

$$\left. \frac{\partial h}{\partial T} \right)_v = T \left. \frac{\partial s}{\partial T} \right)_v + v \left. \frac{\partial P}{\partial T} \right)_v$$

então:

$$\left. \frac{\partial h}{\partial T} \right)_v = C_v + v \frac{\alpha_p}{\beta_T}$$

Expandindo-se:

$$\left. \frac{\partial h}{\partial T} \right)_v = C_p - \frac{T v \alpha_p^2}{\beta_T} + v \frac{\alpha_p}{\beta_T} = C_p + \frac{v \alpha_p}{\beta_T} (1 - \alpha_p T)$$

0.0.2 Exemplo 2

$$\left. \frac{\partial s}{\partial P} \right)_v = ?$$

Usando-se a Regra da Quarta Propriedade com T e expandindo-se:

$$\left. \frac{\partial s}{\partial P} \right)_v = \frac{\left. \frac{\partial s}{\partial T} \right)_v}{\left. \frac{\partial P}{\partial T} \right)_v} = \frac{C_v \beta_T}{T \alpha_p} = \left(C_p - \frac{T v \alpha_p^2}{\beta_T} \right) \frac{\beta_T}{T \alpha_p} = \frac{\beta_T C_p}{T \alpha_p} - v \alpha_p$$

0.0.3 Exemplo 3

$$\left. \frac{\partial a}{\partial s} \right)_P = ?$$

A) Primeiro procedimento: Pela Relação Fundamental na forma de energia livre de Helmholtz:

$$da = -sdT - Pdv$$

Substituindo-se as derivadas parciais:

$$\left. \frac{\partial a}{\partial s} \right)_P = -s \left. \frac{\partial T}{\partial s} \right)_P - P \left. \frac{\partial v}{\partial s} \right)_P$$

Substituindo-se por C_p e α_p e expandindo-se:

$$\left. \frac{\partial a}{\partial s} \right)_P = -\frac{sT}{C_p} - P \frac{\frac{\partial v}{\partial T}}{\left. \frac{\partial s}{\partial T} \right)_P} = -\frac{sT}{C_p} - P \frac{v\alpha_p}{\frac{T}{C_p}} = -\frac{T}{C_p} (s + Pv\alpha_p)$$

B) Segundo procedimento: Usando-se a Regra da Quarta Propriedade com T:

$$\left. \frac{\partial a}{\partial s} \right)_P = \frac{\left. \frac{\partial a}{\partial T} \right)_P}{\left. \frac{\partial s}{\partial T} \right)_P} = \frac{T}{C_p} \left. \frac{\partial a}{\partial T} \right)_P$$

Substituindo-se as derivadas parciais na Relação Fundamental na forma de energia livre de Helmholtz:

$$\left. \frac{\partial a}{\partial T} \right)_P = -s - P \left. \frac{\partial v}{\partial T} \right)_P = -s - Pv\alpha_p$$

Substituindo-se por C_p e as derivadas parciais, obtém-se, conforme esperado a mesma expressão:

$$\left. \frac{\partial a}{\partial s} \right)_P = -\frac{T}{C_p} (s + Pv\alpha_p)$$

0.0.4 Exemplo 4

$$\left. \frac{\partial T}{\partial v} \right)_u = ?$$

A) Primeiro procedimento: Usando-se a Regra Cíclica em T-v-u:

$$\left. \frac{\partial T}{\partial v} \right)_u \left. \frac{\partial v}{\partial u} \right)_T \left. \frac{\partial u}{\partial T} \right)_v = -1$$

Isolando-se a derivada de interesse:

$$\left. \frac{\partial T}{\partial v} \right)_u = - \frac{\left. \frac{\partial u}{\partial v} \right)_T}{\left. \frac{\partial u}{\partial T} \right)_v}$$

Pela Relação Fundamental na forma de Energia Interna:

$$du = Tds - Pdv$$

Substituindo-se as derivadas parciais:

$$\left. \frac{\partial u}{\partial v} \right)_T = T \left. \frac{\partial s}{\partial v} \right)_T - P$$

Pela Relação de Maxwell:

$$\left. \frac{\partial s}{\partial v} \right)_T = \left. \frac{\partial P}{\partial T} \right)_v = \frac{\alpha_p}{\beta_T}$$

Finalmente, substituindo-se na derivada de interesse:

$$\left. \frac{\partial T}{\partial v} \right)_u = - \frac{T \frac{\alpha_p}{\beta_T} - P}{C_p - \frac{T v \alpha_p^2}{\beta_T}} = - \frac{T \alpha_p - P \beta_T}{C_p \beta_T - T v \alpha_p^2} = \frac{P \beta_T - T \alpha_p}{C_p \beta_T - T v \alpha_p^2}$$

B) Segundo procedimento: Pelo método das Tabelas de Bridgman:

$$\left. \frac{\partial T}{\partial v} \right)_u = \frac{\partial T)_u}{\partial v)_u}$$

Das Tabelas de Bridgman, obtemos:

$$\partial T)_u = - \partial u)_T = -T \left. \frac{\partial v}{\partial T} \right)_P - P \left. \frac{\partial v}{\partial P} \right)_T = -T v \alpha_p + P v \beta_T$$

e também:

$$\partial v)_u = -\partial u)_v = -C_p \left. \frac{\partial v}{\partial P} \right)_T - T \left. \frac{\partial v}{\partial T} \right)_P^2 = C_p v \beta_T - T v^2 \alpha_p^2$$

Substituindo-se, obtemos a mesma expressão, como já era esperado:

$$\left. \frac{\partial T}{\partial v} \right)_u = \frac{P v \beta_T - T v \alpha_p}{C_p v \beta_T - T v^2 \alpha_p^2} = \frac{P \beta_T - T \alpha_p}{C_p \beta_T - T v \alpha_p^2}$$

Problema com estados adimensionais Tr,Pr,x - 2020

December 27, 2020

0.0.1 Exemplo de problema completamente adimensional:

Uma substância pura, cuja assinatura é (M, T_c, P_c, ω, Y) , que está em um volume rígido no estado original a $(P_r^{(1)}, x^{(1)})$, sofre um recebimento de calor até a temperatura $T_r^{(2)}$ a partir de uma fonte a T_{fr} . São fornecidos $\frac{C_{vo}}{R}$, T_{or} , P_{or} e a fonte a T_{fr} . Determine a pressão final $P_r^{(2)}$, $\frac{q}{RT_c}$, $\frac{s_{gen}}{R}$, $\frac{w_{rev}}{RT_c}$, etc.

Análise do problema O volume específico permanecerá constante entre 1 e 2:

$$v^{(1)}(P^{(1)}, x^{(1)}) = v^{(2)}((T^{(2)}, P^{(2)})$$

Adimensionalizando-se:

$$\frac{Z^{(1)}(P_r^{(1)}, x^{(1)}, \omega, Y) T_{rsat}(P_r^{(1)})}{P_r^{(1)}} = \frac{Z^{(2)}(T_r^{(2)}, P_r^{(2)}, \omega, Y) T_r^{(2)}}{P_r^{(2)}}$$

Aplicando-se a primeira lei:

$$q = u^{(2)}(T^{(2)}, P^{(2)}) - u^{(1)}(P^{(1)}, x^{(1)})$$

Adimensionalizando-se:

$$\left(\frac{q}{RT_c} \right) = \frac{(u^{(2)} - u^{(1)})}{RT_c}$$

Aplicando-se a segunda lei:

$$s_{gen} = s^{(2)}(T^{(2)}, P^{(2)}) - s^{(1)}(P^{(1)}, x^{(1)}) - \frac{q}{T_f}$$

Adimensionalizando-se:

$$\left(\frac{s_{gen}}{R} \right) = \frac{(s^{(2)} - s^{(1)})}{R} - \left(\frac{q}{RT_c} \right) \frac{1}{T_f}$$

O trabalho reversível ficará:

$$w_{rev} = T_o s_{gen}$$

Adimensionalizando-se:

$$\left(\frac{w_{rev}}{RT_c} \right) = T_{or} \left(\frac{s_{gen}}{R} \right)$$

Os desvios adimensionais de entalpia se dividem com o título do vapor da forma convencional:

$$\left[\frac{(h^* - h)}{RT_c} \right] (P_r^{(1)}, x^{(1)}, \omega, Y) = (1 - x^{(1)}) \left[\frac{(h^* - h)}{RT_c} \right]_l (P_r^{(1)}, \omega, Y) + x^{(1)} \left[\frac{(h^* - h)}{RT_c} \right]_v (P_r^{(1)}, \omega, Y)$$

A relação entre os desvios *adimensionais* de energia interna e de entalpia fica:

$$\left[\frac{u^* - u}{RT_c} \right] = \left[\frac{h^* - h}{RT_c} \right] - T_r (1 - Z)$$

Expandindo-se a variação de energia interna adimensional:

$$\left(\frac{q}{RT_c} \right) = \left[\frac{(u^* - u)}{RT_c} \right] (P_r^{(1)}, x^{(1)}, \omega, Y) - \left[\frac{(u^* - u)}{RT_c} \right] (T_r^{(2)}, P_r^{(2)}, \omega, Y) + \left(\frac{C_{vo}}{R} \right) (T_r^{(2)} - T_{rsat}(P_r^{(1)}))$$

Uma vez que:

$$\frac{C_{vo}}{R} = \frac{C_{po}}{R} - 1$$

Podemos expandir, alternativamente a primeira lei em desvios de entalpia:

$$\begin{aligned} \left(\frac{q}{RT_c} \right) &= \\ \left[\frac{(h^* - h)}{RT_c} \right] (P_r^{(1)}, x^{(1)}, \omega, Y) &- \left[\frac{(h^* - h)}{RT_c} \right] (T_r^{(2)}, P_r^{(2)}, \omega, Y) + \left(\frac{C_{po}}{R} \right) (T_r^{(2)} - T_{rsat}(P_r^{(1)})) + \\ T_{rsat}(P_r^{(1)})Z^{(1)}(P_r^{(1)}, x^{(1)}, \omega, Y) &- T_r^{(2)}Z^{(2)}(T_r^{(2)}, P_r^{(2)}, \omega, Y) \end{aligned}$$

Expendendo-se a segunda lei:

$$\begin{aligned} \left(\frac{s_{gen}}{R} \right) &= \\ \left[\frac{(s^* - s)}{R} \right] (P_r^{(1)}, x^{(1)}, \omega, Y) &- \left[\frac{(s^* - s)}{R} \right] (T_r^{(2)}, P_r^{(2)}, \omega, Y) + \\ \left(\frac{C_{po}}{R} \right) \ln \left(\frac{T_r^{(2)}}{T_{rsat}(P_r^{(1)})} \right) &- \ln \left(\frac{P_r^{(2)}}{P_r^{(1)}} \right) \end{aligned}$$

```
[1]: from LK_proptermo import *
```

Para cálculos SEM o computador

```
[2]: w,Y = 0.,0. # exemplo com substância simples
Cvo_R = 1.7354/0.51835 # metano! Cvo/R
Cpo_R = 1 + Cvo_R # Cpo/R
Cvo_R,Cpo_R # constantes
```

```
[2]: (3.3479309346966337, 4.347930934696634)
```

```
[3]: # Dados
Pr1,x1 = 0.03,0.4
Tr2 = 1.2
Tfr = 5
```

```
[4]: # Estado adimensional em 1
er1 = lk.Nondim_State(Pr=Pr1,x=x1,w=w,Y=Y).prop
```

```
[5]: er1
```

```
[5]: {'Tr': 0.6048286216208224,
'Pr': 0.03,
'h1': 5.148766709444532,
'hv': 0.08055177098862677,
'h': 3.1214807340621697,
'u': 2.749593933629975,
'ul': 4.547293858110277,
'uv': 0.053044046909522344,
'sl': 8.46819537886024,
'sv': 0.08860691764736535,
's': 5.11635999437509,
'x': 0.4,
'Z1': 0.005548299413435112,
'Zv': 0.954519804295324,
'Z': 0.38513690136619066,
'fase': 'saturated mixture'}
```

```
[6]: er1['h'] - er1['Tr']*(1 - er1['Z']),er1['u']
```

```
[6]: (2.749593933629975, 2.749593933629975)
```

```
[7]: def residuo(pr2,args):
    er2 = lk.Nondim_State(Tr=Tr2,Pr=pr2,w=w,Y=Y).prop
    return pr2 - Pr1*er2['Z']/er1['Z']*(Tr2/er1['Tr'])
```

```
[8]: Pr2,ite,res = robustNewton(residuo,1)
Pr2,ite,res
```

```
[8]: (0.15006204479177002, 2, 1.248921771557221e-09)
```

```
[9]: er2 = lk.Nondim_State(Tr=Tr2,Pr=Pr2,w=w,Y=Y).prop
er2
```

```
[9]: {'Tr': 1.2,
      'Pr': 0.15006204479177002,
      'h': 0.11006475706292101,
      'u': 0.07525561420152861,
      's': 0.0629407490424819,
      'Z': 0.9709923809488397,
      'cf': -0.02877988184328562,
      'fase': 'supercritical'}
```

```
[10]: q_RTC = er1['u'] - er2['u'] + Cvo_R*(Tr2 - er1['Tr'])
q_RTC
```

```
[10]: 4.66693098855013
```

```
[11]: sgenR = er1['s'] - er2['s'] + Cpo_R*np.log(Tr2/er1['Tr']) - np.log(Pr2/Pr1) - u
      ↳ q_RTC/Tfr
sgenR
```

```
[11]: 5.489086849022513
```

0.0.2 Metano: Estado adimensional

```
[12]: w,Y = 0.011,0 # metano
Tr,Pr = 0.95,1.40
```

```
[13]: er = lk.Nondim_State(Pr=Pr,Tr=Tr,w=w,Y=Y).prop
```

```
[14]: er
```

```
[14]: {'Tr': 0.95,
      'Pr': 1.4,
      'h': 3.934110754325196,
      'u': 3.2007130795247596,
      's': 3.2746672494725666,
      'Z': 0.22800244757848792,
      'cf': -0.8665019656065875,
      'fase': 'supercritical'}
```

```
'Tr': 0.95, 'Pr': 1.4, 'h': 3.8931603159263872, 'u': 3.1605478907173232, 's': 3.234558928759381,  
'Z': 0.22882902609572192, 'cf': -0.8635045616894483, 'fase': 'supercritical'
```

```
[15]: np.log(er['Pr'])
```

```
[15]: 0.3364722366212129
```

0.0.3 Água: Estado dimensional e adimensional

```
[16]: sub = Substance('water')  
b_sub = Base(sub,Tb=300.,Pb=0.1) # base qualquer
```

Path absoluto da tabela: /home/emanuel/tabelaA1A3.db

```
[17]: T,P = 200+273.15,1. # estado qualquer
```

```
[18]: e_ag = State(sub,b_sub,T=T,P=P)  
e_ag.e
```

```
[18]: {'Tr': 0.7309593696894794,  
       'Pr': 0.045207956600361664,  
       'h': 0.17402293134582286,  
       'u': 0.1326065739605745,  
       's': 0.18410445526988595,  
       'Z': 0.9433397270728706,  
       'cf': -0.05397024294453048,  
       'fase': 'superheated vapor'}
```

```
[19]: # Estado adimensional da água no estado fornecido a T,P:  
er_ag = lk.Nondim_State(Tr=T/sub.Tc,Pr=P/sub.Pc,w=sub.w,Y=sub.Y).prop
```

```
[20]: er_ag
```

```
[20]: {'Tr': 0.7309593696894794,  
       'Pr': 0.045207956600361664,  
       'h': 0.17402293134582286,  
       'u': 0.1326065739605745,  
       's': 0.18410445526988595,  
       'Z': 0.9433397270728706,  
       'cf': -0.05397024294453048,  
       'fase': 'superheated vapor'}
```

```
[21]: # Os estados termodinâmicos adimensionais são EXATAMENTE os mesmos  
e_ag.e == er_ag
```

```
[21]: True
```

```
[22]: # As instâncias são DISTINTAS  
e_ag.e is er_ag
```

```
[22]: False
```

```
[ ]:
```

Problema 11.56 e 11.64 4a VW -misturas - bases fornecidas - si, gp, ps - 2020

July 26, 2020

0.0.1 Análise do problema

O objetivo do problema é a determinação do valor para a entropia molar de uma mistura de metano (A) e etano (B), cujas condições do estado i , dadas por $T^{(i)}, P^{(i)}, y_A^{(i)}$, são conhecidas, admitindo-se um modelo de pseudo-substância pura (ou mistura de gases perfeitos) naquele estado e tal que as respectivas bases de referência para A e para B puros são fornecidas.

Para simplificar um pouco a notação, façamos $y_A = y_A^{(i)}$. Sabemos que, de um modo geral, a entropia específica molar da mistura homogênea binária de A,B no estado i , isto é, em $T^{(i)}, P^{(i)}, y_A$, em termos das entropias parciais molares de A e de B é dada por:

$$\bar{s}^{(i)}(T^{(i)}, P^{(i)}, y_A) = y_A \bar{S}_A^{(i)}(T^{(i)}, P^{(i)}, y_A) + y_B \bar{S}_B^{(i)}(T^{(i)}, P^{(i)}, y_A)$$

(1)

Como você aprendeu, para nos livrarmos das propriedades parciais molares precisamos de um modelo para a mistura. Nesse caso, como pede o problema, usaremos o modelo de pseudo-substância pura, mas com dois modelos pseudo-críticos distintos: a Regra de Kay e o modelo de mistura de Lee-Kesler-Wu-Stiel. Qualquer que seja o modelo pseudo-crítico, a entropia da mistura será expressa por:

$$\bar{s}^{(i)}(T^{(i)}, P^{(i)}, y_A) = \bar{s}^{*(i)}(T^{(i)}, P^{(i)}, y_A) - (y_A + y_B) \left[\frac{\bar{s}^* - \bar{s}}{\bar{R}} \right]_m^{(i)} (T_{rm}^{(i)}, P_{rm}^{(i)}, \omega_m, Y_m) \bar{R}$$

(2)

onde, a partir da assinatura da pseudo-substância pura ($M_m^{(i)}, T_{Cm}^{(i)}, P_{Cm}^{(i)}, \omega_m^{(i)}, Y_m^{(i)}$) obtida por meio de algum modelo pseudo-crítico aplicado ao estado i da mistura, têm-se que:

$$M_m^{(i)} = y_A^{(i)} M_A + y_B^{(i)} M_B$$

$$T_{rm}(i) = T^{(i)} / T_{Cm}^{(i)}$$

$$P_{rm}(i) = P^{(i)} / P_{Cm}^{(i)}$$

Pela equação (2) a entropia específica molar da mistura como uma pseudo-substância pura pode ser expressa como uma mistura de gases perfeitos corrigida pela entropia residual no estado i da mistura como pseudo-substância pura.

Note que a mistura, como uma mistura de gases perfeitos em i , é expressa como:

$$\bar{s}^{*(i)}(T^{(i)}, P^{(i)}, y_A) = y_A \bar{s}_A^{*(i)}(T^{(i)}, P^{(i)}) + y_B \bar{s}_B^{*(i)}(T^{(i)}, P^{(i)}) - \bar{R} [y_A \ln(y_A) + y_B \ln(y_B)]$$

(3)

Entretanto, depois de substituir (2) em (3), você poderia se inquirir a respeito dos valores de $\bar{s}_A^{*(i)}(T^{(i)}, P^{(i)})$ e de $\bar{s}_B^{*(i)}(T^{(i)}, P^{(i)})$, cada uma delas um valor do gás perfeito no estado i para as substâncias puras A e B respectivamente. Pelo que aprendeu em Termodinâmica para substâncias puras, estes valores em qualquer estado dependem de suas respectivas bases de referência que são completamente independentes entre si, ou seja, deve estar claro agora para você que, assim como o valor isolado de uma propriedade não mensurável de uma substância pura não tem significado, assim também o valor isolado das propriedades não mensuráveis da mistura também não. Você deve ter notado adicionalmente que sequer tem sentido o conceito de uma base de referência para as misturas.

Trata o problema então de se determinar o valor da entropia da mistura em um estado e composição dadas $T^{(i)}, P^{(i)}, y_A$, assumindo-se o modelo de pseudo-substância pura (ou de misturas de gases perfeitos) e fornecendo-se as bases de referência independentes para as entropias de A e de B puros. Com essas bases, você poderá determinar os valores de $\bar{s}_A^{*(i)}(T^{(i)}, P^{(i)})$ e de $\bar{s}_B^{*(i)}(T^{(i)}, P^{(i)})$, que substituídos em (3), lhe permitirá obter um valor em (2) para $\bar{s}^{*(i)}(T^{(i)}, P^{(i)}, y_A)$, objetivo do problema.

Vamos denominar por conveniência as bases de referência para A e B puros como bA e bB respectivamente. Assim, T_{bA} seria a temperatura referente ao estado da base bA e assim por diante. Da Termodinâmica das substâncias puras aprendemos que podemos expandir qualquer entropia de uma substância pura na forma de diferenças. De modo geral, para entropia da substância pura A no estado i :

$$\bar{s}_A^{*(i)}(T^{(i)}, P^{(i)}) = \bar{s}_{bA}(T_{bA}, P_{bA}) + (\bar{s}^* - \bar{s})(T_{bA}, P_{bA}) - (\bar{s}^* - \bar{s})(T^{(i)}, P^{(i)}) + \bar{s}_A^{*(i)}(T^{(i)}, P^{(i)}) - \bar{s}_A^*(T_{bA}, P_{bA})$$

(4)

e, analogamente para a entropia da substância pura B no estado i :

$$\bar{s}_B^{*(i)}(T^{(i)}, P^{(i)}) = \bar{s}_{bB}(T_{bB}, P_{bB}) + (\bar{s}^* - \bar{s})(T_{bB}, P_{bB}) - (\bar{s}^* - \bar{s})(T^{(i)}, P^{(i)}) + \bar{s}_B^{*(i)}(T^{(i)}, P^{(i)}) - \bar{s}_B^*(T_{bB}, P_{bB})$$

. (5)

Em (4) e (5), $\bar{s}_{bA}(T_{bA}, P_{bA})$ e $\bar{s}_{bB}(T_{bB}, P_{bB})$ representam os valores **arbitrários** nos estados **arbitrários** para as bases bA e bB respectivamente.

Veja bem, se forem fornecidos os estados da bases dados por (T_b, P_b) , ou (T_b, x_b) , ou (P_b, x_b) e seus valores para entalpia h_b e entropia s_b , você deve se recordar, como aprendeu em Termodinâmica das substâncias puras, que você *não precisa* de fato fazer estas expansões, pois elas são realizadas

interna e automaticamente pelo programa *LK_proptermo*. Mas mostraremos aqui e numericamente como você pode realizá-las.

No caso particular do problema no qual foi escolhido o modelo de pseudo-substância pura (ou mistura de gases perfeitos) para a mistura no estado i , precisamos determinar valores para $\bar{s}_A^{*(i)}(T^{(i)}, P^{(i)})$ e para $\bar{s}_B^{*(i)}(T^{(i)}, P^{(i)})$, que são valores do gás perfeito para A e B puros no estado i . Então com estas condições as entropias residuais de A e de B no estado i são nulas. Observe novamente e cuidadosamente que não estamos nos referindo aqui aos **reais** estados de A e B puros em i e ainda muito menos da mistura. Aplicando-se em (4) e em (5), obtemos, se T_{bA}, P_{bA} forem independentes entre si:

$$\bar{s}_A^{*(i)}(T^{(i)}, P^{(i)}) = \bar{s}_{bA}(T_{bA}, P_{bA}) + (\bar{s}^* - \bar{s})(T_{bA}, P_{bA}) + \bar{s}_A^{*(i)}(T^{(i)}, P^{(i)}) - \bar{s}_A^*(T_{bA}, P_{bA}) \quad (6)$$

e, da mesma forma, se T_{bB}, P_{bB} forem independentes entre si:

$$\bar{s}_B^{*(i)}(T^{(i)}, P^{(i)}) = \bar{s}_{bB}(T_{bB}, P_{bB}) + (\bar{s}^* - \bar{s})(T_{bB}, P_{bB}) + \bar{s}_B^{*(i)}(T^{(i)}, P^{(i)}) - \bar{s}_B^*(T_{bB}, P_{bB}) \quad . \quad (7)$$

Estamos quase lá. Agora precisamos analisar com muito cuidado os dados específicos fornecidos pelo problema para cada base: $\bar{s}_{bA}(T_{bA}, P_{bA})$ e $\bar{s}_{bB}(T_{bB}, P_{bB})$.

No caso do metano (A), a base é um valor escolhido para a entropia do LÍQUIDO SATURADO REAL $\bar{s}_{bA_l}(T_{bA})$ na temperatura T_{bA} , portanto as propriedades primárias são T_{bA}, x_{bA} com $x_{bA} = 0$.

Já no caso do etano (B), a base será um outro valor escolhido para a entropia do vapor como GÁS PERFEITO na temperatura T_{bB} , mas **não foi** fornecida a pressão. Nesse caso então adotaremos VAPOR SATURADO em T_{bB} como GÁS PERFEITO hipotético $\bar{s}_{bB}^*(T_{bB}, P_{sat}(T_{bB}))$, portanto as propriedades primárias serão T_{bB}, x_{bB} com $x_{bB} = 1$. Porém, como se refere o problema a uma entropia de um gás perfeito para B e a entropia do gás perfeito de uma substância pura é sempre uma função de T, P , precisamos incluir na forma funcional, além de T_{bB} , a pressão $P_{sat}(T_{bB})$. Como ficarão as expressões (6) e (7) agora?

Vejamos: Em A, haverá a **entropia residual do líquido saturado em T_{bA}** :

$$\bar{s}_A^{*(i)}(T^{(i)}, P^{(i)}) = \bar{s}_{bA_l}(T_{bA}) + (\bar{s}^* - \bar{s})_{A_l}(T_{bA}) + \bar{s}_A^{*(i)}(T^{(i)}, P^{(i)}) - \bar{s}_{bA}^*(T_{bA}, P_{sat}(T_{bA})) \quad (8)$$

Você deverá ter concluído que não restará nenhuma entropia residual em B, pois todos os termos são gases perfeitos:

$$\bar{s}_B^{*(i)}(T^{(i)}, P^{(i)}) = \bar{s}_{bB}^*(T_{bB}, P_{sat}(T_{bB})) + \bar{s}_B^{*(i)}(T^{(i)}, P^{(i)}) - \bar{s}_{bB}^*(T_{bB}, P_{sat}(T_{bB})) \quad . \quad (9)$$

O único desvio de entropia, aquele em A, pode ainda ser substituído pela forma adimensional redimensionalizada:

(10)

Em (10) a temperatura reduzida será definida por:

$$T_{rbA} = T_{bA}/T_{CA}$$

Se quisermos substituir todas as expressões em uma só, teremos para a mistura no estado i a assustadora expressão:

$$\begin{aligned} \bar{s}^{(i)}(T^{(i)}, P^{(i)}, y_A) = \\ y_A \left[\bar{s}_{bA_l}(T_{bA}) + \left[\frac{\bar{s}^* - \bar{s}}{\bar{R}} \right]_{A_l} (T_{rbA}, \omega_A, Y_A) \bar{R} + \bar{s}_A^{*(i)}(T^{(i)}, P^{(i)}) - \bar{s}_{bA}^*(T_{bA}, P_{sat}(T_{bA})) \right] + \\ y_B \left[\bar{s}_{bB}^*(T_{bB}, P_{sat}(T_{bB})) + \bar{s}_B^{*(i)}(T^{(i)}, P^{(i)}) - \bar{s}_{bB}^*(T_{bB}, P_{sat}(T_{bB})) \right] - \\ \bar{R} [y_A \ln(y_A) + y_B \ln(y_B)] - (y_A + y_B) \left[\frac{\bar{s}^* - \bar{s}}{\bar{R}} \right]_m^{(i)} (T_{rm}^{(i)}, P_{rm}^{(i)}, \omega_m, Y_m) \bar{R} \end{aligned} \quad (11)$$

Obviamente a **única** alteração se for assumida uma mistura de gases perfeitos em i é que não haverá o último termo.

Você conseguiria escrever como ficaria esta expressão se, com as mesmas bases fornecidas, a mistura fosse uma **solução ideal** em i ? Isso mesmo, haveria a necessidade de se obterem as entropias de A e B puros $\bar{s}_A^{(i)}(T^{(i)}, P^{(i)})$ e $\bar{s}_B^{(i)}(T^{(i)}, P^{(i)})$ no estado i e portanto, dever-se-iam incluir as entropias residuais $(\bar{s}^* - s)_A^{(i)}(T^{(i)}, P^{(i)})$ de A puro em (6) e $(\bar{s}^* - s)_B^{(i)}(T^{(i)}, P^{(i)})$ de B puro em (7) a serem determinadas no estado i . Finalmente, é claro, o último termo, a entropia residual da pseudo-substância pura em i , seria suprimido.

Não é **muito** mais tranquilizador saber que, como veremos, o programa *LK_propterm* manipula boa parte disso tudo para você? Entretanto, para poder utilizar o programa de forma correta você **precisa** entender a lógica discutida parte por parte nesse texto e só então aplicá-la numericamente.

[1]: `from LK_proptermo import *`

[2]: `A = Substance('methane')`
`B = Substance('ethane')`

Path absoluto da tabela: /home/emanuel/tabelaA1A3.db
Path absoluto da tabela: /home/emanuel/tabelaA1A3.db

Dados das bases fornecidas

Base do Metano Entropia do LÍQUIDO SATURADO:

$$\bar{s}_{bA_l}(T_{bA}) = 100 \text{ kJ/kmolK} \text{ em } T_{bA} = -100^\circ\text{C}$$

```
[3]: # Em LK_proptermo as bases são definidas em base de massa
TbA = -100+273.15 # K
sbA = 100./A.M # kJ/kg K
bA = Base(A,Tb=TbA,xb=0,sb=sbA) # liq saturado hb = 0.
bA.e
```

```
[3]: {'Tr': 0.9086853844135396,
      'Pr': 0.56272294387265,
      'hl': 4.062539793898433,
      'hv': 0.9628637630884888,
      'h': 4.062539793898433,
      'u': 3.240213769813718,
      'ul': 3.240213769813718,
      'uv': 0.650090028045398,
      's1': 4.185469219810102,
      'sv': 0.7743039378937479,
      's': 4.185469219810102,
      'x': 0,
      'Z1': 0.09503769050336375,
      'Zv': 0.6557953496248284,
      'Z': 0.09503769050336375,
      'fase': 'saturated mixture'}
```

Base do Etano Entropia do VAPOR SATURADO como gás perfeito, adotada arbitrariamente, pois não foi informada a pressão:

$$\bar{s}_{bB}^*(T_{bB}, P_{sat}(T_{bB})) = 200 \text{ kJ/kmolK para } T_{bB} = -100^\circ\text{C}$$

```
[4]: TbB = -100.+273.15
sbB = 200./B.M # kJ/kg K
bB = Base(B,Tb=TbB,xb=1.,sb=sbB,gp=1) # gás perfeito hb=0.
bB.e
```

```
[4]: {'Tr': 0.5670913437919628,
      'Pr': 0.01051106813666906,
      'hl': 6.054377424423015,
      'hv': 0.03880462454041287,
      'h': 0.0,
      'u': 0.0,
```

```

'u1': 5.488378512408265,
'uv': 0.026540287491972292,
'sl': 10.654783976628813,
'sv': 0.04701705204349395,
's': 0.0,
'x': 1.0,
'Z1': 0.0019263770981025512,
'Zv': 0.9783732599999979,
'Z': 1.0,
'fase': 'saturated mixture',
'cf': 0.0}

```

Dados: (T,P) e a composição da mistura, ou seja $T^{(i)}, P^{(i)}, y_A$

[5]: $T, P = 20 + 273.15, 4. \text{ # } K, MPa$
 $y_A = 0.5$
 $y_B = 1 - y_A$

Assinaturas das substâncias puras A e B

[6]: $(A.M, B.M), (A.Tc, B.Tc), (A.Pc, B.Pc), (A.w, B.w), (A.Y, B.Y)$

[6]: $((16.0436, 30.069),$
 $(190.55, 305.33),$
 $(4.599, 4.871),$
 $(0.011, 0.099),$
 $(0.0, 0.0))$

0.0.2 Solução ideal

Estado real de A puro a T,P na base fornecida

[7]: $eA = \text{State}(A, bA, T=T, P=P, gp=0)$
 $eA.e \text{ # estado adimensional não depende de base}$

[7]: $\{'Tr': 1.5384413539753343,$
 $'Pr': 0.8697542944118286,$
 $'h': 0.41391763131238535,$
 $'u': 0.30602047529018206,$
 $'s': 0.19794498919092607,$
 $'Z': 0.9298659284324379,$
 $'cf': -0.07110500107540536,$
 $'fase': 'supercritical'\}$

[8]: $\# entropia real de A puro a T,P na base fornecida$
 $smA = eA.s*A.M \text{ # este valor depende da base}$

```
smA # kJ/kmol K
```

[8]: 147.32513810109506

```
[9]: (sbA*A.M + bA.e['s']*A.Ru -  
eA.e['s']*A.Ru +  
A.sstarm(T,P) - A.sstarm(TbA,bA.Pb) ),smA # mesma coisa!
```

[9]: (147.3251381010951, 147.32513810109506)

Estado real de B puro a T,P na base fornecida

```
[10]: eB = State(B,bB,T=T,P=P,gp=0)  
eB.e # estado adimensional não depende de base
```

```
[10]: {'Tr': 0.9601087348115154,  
'Pr': 0.821186614658181,  
'h': 4.0283980109275666,  
'u': 3.2052892874452525,  
's': 3.789857177279507,  
'Z': 0.14269218304330555,  
'cf': -0.4059155147775077,  
'fase': 'compressed liquid'}
```

```
[11]: # Entropia real de B puro a T,P na base fornecida  
smB = eB.s*B.M  
smB # kJ/kmol K
```

[11]: 154.58887338263932

```
[12]: (sbB*B.M - eB.e['s']*B.Ru +  
B.sstarm(T,P) - B.sstarm(TbB,bB.Pb) ),smB # mesma coisa!
```

[12]: (154.58887338263932, 154.58887338263932)

Modelo de Solução ideal

```
[13]: sm_si = (yA*(smA - A.Ru*np.log(yA)) +  
yB*(smB - B.Ru*np.log(yB)) )  
sm_si # kJ/kmol K
```

[13]: 156.72018490610466

```
[14]: vm_si = yA*eA.v*A.M + yB*eB.v*B.M  
vm_si # m3/kmol
```

[14]: 0.32678145578765605

```
[15]: ((yA*eA.e['Z'] +
yB*eB.e['Z'])*A.Ru*T/(1000*P)),vm_si # mesma coisa!
```

```
[15]: (0.3267814557876561, 0.32678145578765605)
```

0.0.3 Regra de Kay

```
[16]: Tcmk = yA*A.Tc + yB*B.Tc
Pcmk = yA*A.Pc + yB*B.Pc
wmk = yA*A.w + yB*B.w
Ymk = yA*A.Y + yB*B.Y
Tcmk,Pcmk,wmk,Ymk
```

```
[16]: (247.94, 4.735, 0.055, 0.0)
```

```
[17]: ek_mist = lk.Nondim_State(Tr=T/Tcmk,Pr=P/Pcmk,w=wmk,Y=Ymk).prop
ek_mist
```

```
[17]: {'Tr': 1.1823425022182785,
'Pr': 0.8447729672650475,
'h': 0.7459610483936489,
'u': 0.5271696165955648,
's': 0.4555048524956102,
'Z': 0.8149508865767799,
'cf': -0.175413047346523,
'fase': 'supercritical'}
```

```
[18]: vm_k = ek_mist['Z']*A.Ru*T/(1000.*P)
vm_k # m3/kmol
```

```
[18]: 0.4965900388270458
```

0.0.4 Modelo Pseudo-Critico de Lee-Kesler-Wu-Stiel

```
[19]: Tcm,Pcm,wm,Ym = pseudocrit((yA,yB),
(A.Tc,B.Tc),
(A.Pc,B.Pc),
(A.w,B.w),
(A.Y,B.Y))
Tcm,Pcm,wm,Ym
```

```
[19]: (250.151350367483, 4.847377401612279, 0.055, 0.0)
```

```
[20]: elk_mist = lk.Nondim_State(Tr=T/Tcm,Pr=P/Pcm,w=wm,Y=Ym).prop
elk_mist
```

```
[20]: {'Tr': 1.1718905357470593,
       'Pr': 0.8251884820582704,
       'h': 0.7429470783311117,
       'u': 0.5240320713210977,
       's': 0.4573055051392195,
       'Z': 0.8131950038571993,
       'cf': -0.1766675970136161,
       'fase': 'supercritical'}
```

```
[21]: vm_lk = elk_mist['Z']*A.Ru*T/(1000.*P)
      vm_lk # m3/kmol
```

```
[21]: 0.4955200923035749
```

0.0.5 Estados do gás perfeito a T,P nas bases fornecidas

Estado de A puro como GÁS PERFEITO a T,P na base fornecida

$$\bar{s}_A^{*(i)}(T^{(i)}, P^{(i)}) = \bar{s}_{bA_l}(T_{bA}) + \left[\frac{\bar{s}^* - \bar{s}}{\bar{R}} \right]_{A_l} (T_{rbA}, \omega_A, Y_A) \bar{R} + \bar{s}_A^{*(i)}(T^{(i)}, P^{(i)}) - \bar{s}_{bA}^*(T_{bA}, P_{sat}(T_{bA}))$$

```
[22]: # Entropia de A como gás perfeito a T,P ba base fornecida
smgpA = smA + eA.e['s']*A.Ru

egpA = State(A,bA,T=T,P=P, gp=1) # gás perfeito
egpA.s*A.M, smgpA # mesma coisa!
```

```
[22]: (148.97095369317293, 148.9709536931729)
```

```
[23]: egpA.e
```

```
[23]: {'Tr': 1.5384413539753343,
       'Pr': 0.8697542944118286,
       'h': 0.0,
       'u': 0.0,
       's': 0.0,
       'Z': 1.0,
       'cf': 0.0,
       'fase': 'supercritical'}
```

Estado de B puro como GÁS PERFEITO a T,P na base fornecida

$$\bar{s}_B^{*(i)}(T^{(i)}, P^{(i)}) = \bar{s}_{bB}(T_{bB}, P_{sat}(T_{bB})) + \bar{s}_B^{*(i)}(T^{(i)}, P^{(i)}) - \bar{s}_{bB}^*(T_{bB}, P_{sat}(T_{bB}))$$

```
[24]: # Entropia de B puro como gás perfeito na base fornecida
smgpB = smB + eB.e['s']*B.Ru

egpB = State(B,bB,T=T,P=P,gp=1) # gás perfeito
egpB.s*B.M,smgpB # mesma coisa!
```

[24]: (186.09967878170156, 186.09967878170156)

```
[25]: egpB.e
```

```
[25]: {'Tr': 0.9601087348115154,
'Pr': 0.821186614658181,
'h': 0.0,
'u': 0.0,
's': 0.0,
'Z': 1.0,
'cf': 0.0,
'fase': 'compressed liquid'}
```

Modelo da Regra de Kay

$$\bar{s}^{(i)}(T^{(i)}, P^{(i)}, y_A) = \bar{s}^{*(i)}(T^{(i)}, P^{(i)}, y_A) - (y_A + y_B) \left[\frac{\bar{s}^* - \bar{s}}{\bar{R}} \right]_m^{(i)} (T_{rm}^{(i)}, P_{rm}^{(i)}, \omega_m, Y_m) \bar{R}$$

```
[26]: sm_k = (yA*(egpA.s*A.M - A.Ru*np.log(yA)) +
            yB*(egpB.s*B.M - B.Ru*np.log(yB)) -
            ek_mist['s']*A.Ru)
sm_k # kJ/kmol K
```

[26]: 169.51119575055142

Modelo de Lee-Kesler

$$\bar{s}^{(i)}(T^{(i)}, P^{(i)}, y_A) = \bar{s}^{*(i)}(T^{(i)}, P^{(i)}, y_A) - (y_A + y_B) \left[\frac{\bar{s}^* - \bar{s}}{\bar{R}} \right]_m^{(i)} (T_{rm}^{(i)}, P_{rm}^{(i)}, \omega_m, Y_m) \bar{R}$$

```
[27]: sm_lk = (yA*(egpA.s*A.M - A.Ru*np.log(yA)) +
              yB*(egpB.s*B.M - B.Ru*np.log(yB)) -
              elk_mist['s']*A.Ru)
sm_lk # kJ/kmol K
```

[27]: 169.4962242061396

0.0.6 Mistura de Gases Perfeitos

$$\bar{s}^{*(i)}(T^{(i)}, P^{(i)}, y_A) = y_A \bar{s}_A^{*(i)}(T^{(i)}, P^{(i)}) + y_B \bar{s}_B^{*(i)}(T^{(i)}, P^{(i)}) - \bar{R} [y_A \ln(y_A) + y_B \ln(y_B)]$$

```
[28]: vm_gp = A.Ru*T/(1000.*P)
      vm_gp # m3/kmol
```

[28]: 0.609349651625

```
[29]: sm_gp = (yA*(egpA.s*A.M - A.Ru*np.log(yA)) +
      yB*(egpB.s*B.M - B.Ru*np.log(yB)) )
      sm_gp # kJ/kmol K
```

[29]: 173.2984954016747

0.0.7 Sumário dos resultados

```
[30]: Tcmk,Pcmk,wmk,Ymk # Kay
```

[30]: (247.94, 4.735, 0.055, 0.0)

```
[31]: Tcm,Pcm,wm,Ym # Lee-Kesler
```

[31]: (250.151350367483, 4.847377401612279, 0.055, 0.0)

```
[32]: vm_si,vm_k,vm_lk,vm_gp # m3/kmol
```

[32]: (0.32678145578765605, 0.4965900388270458, 0.4955200923035749, 0.609349651625)

```
[33]: sm_si,sm_k,sm_lk,sm_gp # kJ/kmolK
```

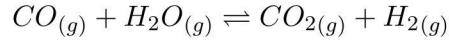
[33]: (156.72018490610466, 169.51119575055142, 169.4962242061396, 173.2984954016747)

Observe que nesse caso particular os modelos pseudo-críticos se equivalem, ao passo que os modelos de solução ideal e de mistura de gases perfeitos estão muito distantes.

Determinação da constante de equilíbrio - reação água-gás 2020

December 27, 2020

Reação gás-água (Water-Gas Shift Reaction):



A constante de equilíbrio é definida por:

$$\ln(K_T) = -\frac{\Delta G^o(T)}{RT}$$

Precisamos expressar $\Delta G^o(T)$, na forma de $\Delta H^o(T)$ e $\Delta S^o(T)$:

$$\begin{aligned}\Delta G^o(T) &= G^o_P(T, P^o) - G^o_R(T, P^o) = \\ \Delta H^o(T) - T\Delta S^o(T) &= \\ (H^o_P(T, P^o) - H^o_R(T, P^o)) - T(S^o_P(T, P^o) - S^o_R(T, P^o))\end{aligned}$$

Expandindo-se nos “produtos”:

$$G^o_P(T, P^o) = \bar{h}_{CO_2}^o(T, P^o) + \bar{h}_{H_2}^o(T, P^o) - T(\bar{s}_{CO_2}^o(T, P^o) + \bar{s}_{H_2}^o(T, P^o))$$

Analogamente com os “reagentes”:

$$G^o_R(T, P^o) = \bar{h}_{CO}^o(T, P^o) + \bar{h}_{H_2O}^o(T, P^o) - T(\bar{s}_{CO}^o(T, P^o) + \bar{s}_{H_2O}^o(T, P^o))$$

Cada uma das entalpias deve ser reduzida à base das entalpias de formação:

$$\bar{h}_{CO}^o(T, P^o) = \bar{h}_{fCO}^{o*}(T^o) + (\bar{h}_{CO}^*(T) - \bar{h}_{CO}^*(T^o))$$

$$\bar{h}_{CO_2}^o(T, P^o) = \bar{h}_{fCO_2}^{o*}(T^o) + (\bar{h}_{CO_2}^*(T) - \bar{h}_{CO_2}^*(T^o))$$

$$\bar{h}_{H_2}^o(T, P^o) = \bar{h}_{fH_2}^{o*}(T^o) + (\bar{h}_{H_2}^*(T) - \bar{h}_{H_2}^*(T^o))$$

$$\bar{h}_{H_2O}^o(T, P^o) = \bar{h}_{fH_2O}^{o*}(T^o) + (\bar{h}_{H_2O}^*(T) - \bar{h}_{H_2O}^*(T^o))$$

Cada uma das entropias deve ser expressa na base das entropias absolutas:

$$\bar{s}_{CO}^o(T, P^o) = \bar{s}_{CO}^{o*}(T^o, P^o) + (\bar{s}_{CO}^*(T, P^o) - \bar{s}_{CO}^*(T^o, P^o))$$

$$\bar{s}_{CO_2}^o(T, P^o) = \bar{s}_{CO_2}^{o*}(T^o, P^o) + (\bar{s}_{CO_2}^*(T, P^o) - \bar{s}_{CO_2}^*(T^o, P^o))$$

$$\bar{s}_{H_2}^o(T, P^o) = \bar{s}_{H_2}^{o*}(T^o, P^o) + (\bar{s}_{H_2}^*(T, P^o) - \bar{s}_{H_2}^*(T^o, P^o))$$

$$\bar{s}_{H_2O}^o(T, P^o) = \bar{s}_{H_2O}^{o*}(T^o, P^o) + (\bar{s}_{H_2O}^*(T, P^o) - \bar{s}_{H_2O}^*(T^o, P^o))$$

```
[1]: %matplotlib inline
from LK_proptermo import *
```

```
[2]: hfm_co2 = -393522. # kJ/kmol
hfm_h2o = -241826. # gás perfeito
hfm_h2 = 0.
hfm_co = -110527.
```

```
[3]: som_co2 = 213.974 # kJ/kmol K
som_h2o = 188.835
som_h2 = 130.678
som_co = 197.653
```

```
[4]: To,Po = 25+273.15,0.1
```

```
[5]: co2 = Substance(formula='CO2')
h2o = Substance('water')
h2 = Substance(formula='H2')
co = Substance(formula='CO')
```

Path absoluto da tabela: /home/emanuel/tabelaA1A3.db
Path absoluto da tabela: /home/emanuel/tabelaA1A3.db
Path absoluto da tabela: /home/emanuel/tabelaA1A3.db
Path absoluto da tabela: /home/emanuel/tabelaA1A3.db

```
[6]: b_co2 = Base(co2,Tb=To,Pb=Po,hb=hfm_co2/co2.M,sb=som_co2/co2.M,gp=1)
b_h2o = Base(h2o,Tb=To,Pb=Po,hb=hfm_h2o/h2o.M,sb=som_h2o/h2o.M,gp=1)
b_h2 = Base(h2,Tb=To,Pb=Po,hb=hfm_h2/h2.M,sb=som_h2/h2.M,gp=1)
b_co = Base(co,Tb=To,Pb=Po,hb=hfm_co/co.M,sb=som_co/co.M,gp=1)
```

```
[7]: Ts = np.r_[298.15,400,500:1100:7j,1200:4200:200,4500:6000:4j]
Ts #K
```

```
[7]: array([ 298.15,  400. ,  500. ,  600. ,  700. ,  800. ,  900. ,
   1000. , 1100. , 1200. , 1400. , 1600. , 1800. , 2000. ,
  2200. , 2400. , 2600. , 2800. , 3000. , 3200. , 3400. ,
  3600. , 3800. , 4000. , 4500. , 5000. , 5500. , 6000. ])
```

```
[8]: # Wikipedia 600 a 2000K
K = 10**(-2.4198 + 0.0003855*Ts + 2180.6/Ts )
#K = np.where(Ts>2000,np.nan,K)
K
```

```
[8]: array([1.02071633e+05, 1.53426366e+03, 1.36191499e+02, 2.79147237e+01,
 9.22981204e+00, 4.11481202e+00, 2.23892161e+00, 1.40055446e+00,
 9.69646951e-01, 7.24380360e-01, 4.75851485e-01, 3.62973568e-01,
 3.05875325e-01, 2.76375792e-01, 2.62713154e-01, 2.59407981e-01,
 2.63749902e-01, 2.74401016e-01, 2.90781429e-01, 3.12778937e-01,
 3.40606508e-01, 3.74733244e-01, 4.15855170e-01, 4.64889781e-01,
 6.30271657e-01, 8.78658268e-01, 1.25003660e+00, 1.80564901e+00])
```

T (°C)	200	250	300	350	400	450	500	550
K _{eq}	210.82	83.956	38.833	20.303	11.723	7.3369	4.9035	3.4586

The equilibrium constant for the reaction derived from thermodynamics is given in equation 16.

$$\ln(K_{eq}) = \frac{5693.5}{T} + 1.077 \ln(T) + 5.44 \times 10^{-4} T - 1.125 \times 10^{-7} T^2 - \frac{49170}{T^2} - 13.148 \quad (16)$$

The equilibrium constant $K(T)$ equation, Equation 3.69, is a curve fit of JANAF Table data for $400 < T < 3200$:

$$\ln K(T) = 2.743 - \frac{1.761}{t} - \frac{1.611}{t^2} - \frac{0.2803}{t^3} \quad \left(t = \frac{T}{1000} \right) \quad (3.69)$$

```
[9]: lnKeq = 5693.5/Ts + 1.077*np.log(Ts) + 5.44e-4*Ts - 1.125e-7*Ts**2 - 49170/
      - Ts**2 - 13.148
lnKeq
```

```
[9]: array([ 1.16834630e+01,  7.43084482e+00,  4.97932792e+00,  3.37997657e+00,
 2.26641301e+00,  1.45457371e+00,  8.42061568e-01,  3.67482435e-01,
 -8.15077361e-03, -3.10749748e-01, -7.63167986e-01, -1.08052319e+00,
 -1.31272170e+00, -1.48937055e+00, -1.62908353e+00, -1.74411253e+00,
 -1.84282771e+00, -1.93112628e+00, -2.01327213e+00, -2.09241715e+00,
 -2.17093606e+00, -2.25064802e+00, -2.33296707e+00, -2.41900666e+00,
 -2.65578713e+00, -2.93074973e+00, -3.24990251e+00, -3.61707178e+00])
```

```
[10]: # NOTA: O sinal negativo do ÚLTIMO termo do ferguson 3rd está errado!
# No ferguson 2nd o sinal estava CORRETO (+).
```

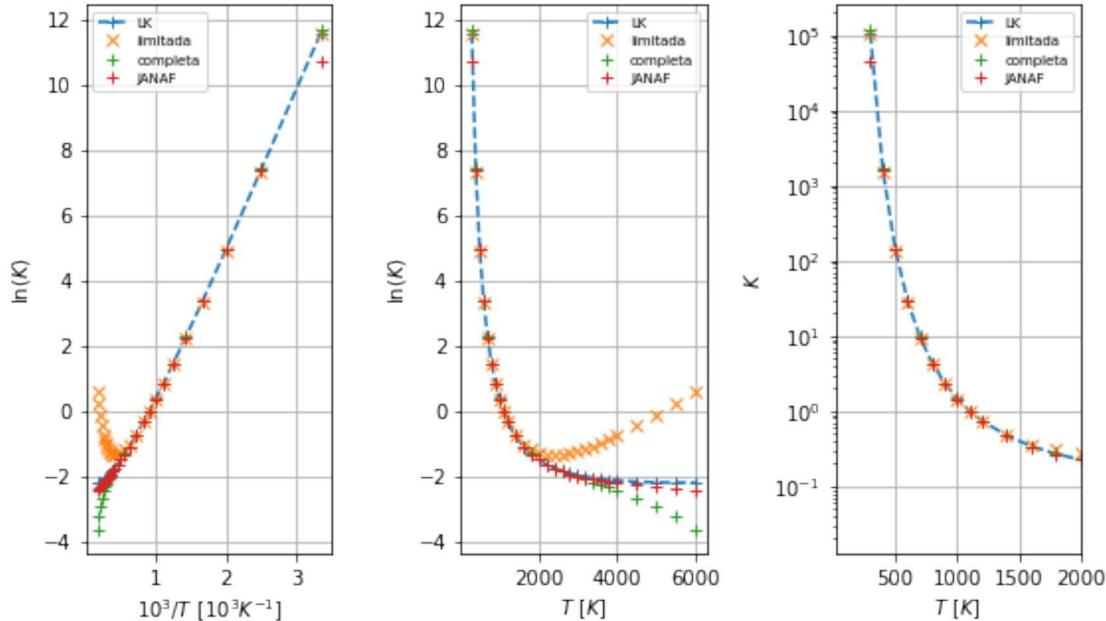
```

plt.subplot(1,3,2)
plt.plot(Ts,np.log(K),'+--',label='LK')
plt.plot(Ts,np.log(K),'x',label='limitada')
plt.plot(Ts,np.log(Keq),'+',label='completa')
plt.plot(Ts,np.log(Kjan),'+',label='JANAF')
plt.grid(True)
plt.ylabel('$\ln(K)$')
plt.xlabel('$T \ [K]$')
plt.legend(loc='best',fontsize='x-small')

plt.subplot(1,3,3)
plt.semilogy(Ts,np.exp(lnK),'+--',label='LK')
plt.semilogy(Ts,K,'x',label='limitada')
plt.semilogy(Ts,np.exp(lnKeq),'+',label='completa')
plt.semilogy(Ts,np.exp(lnKjan),'+',label='JANAF')
plt.grid(True)
plt.ylabel('K')
plt.xlabel('$T \ [K]$')
plt.xlim(right=2000.)
plt.legend(loc='best',fontsize='x-small')
plt.tight_layout()

```

Water-Gas-Shift Reaction - Equilibrium Constant K(T)



Adiabatic Flame Temperature - Dissociation - 2020

December 27, 2020

```
[1]: %matplotlib inline
from LK_proptermo import *
from equilibriumNR import equilibrium
```

0.0.1 Combustível e ar

```
[2]: alfa,beta,gama,delta = 2,2,0,0
phi,T,P = 1.2,3000.,0.10132 # 1 atm # T,P dos produtos
```

0.0.2 Equilíbrio da combustão do combustível dado a T,P

```
[3]: props,ite,res = equilibrium(alfa,beta,gama,delta,phi,
                                T,P,jacob = 1)
```

```
[4]: #fracs,n,N,cp,h,s,v,M,x,ite,res
props,ite,res
```

```
[4]: {'yi': {'CO2': 0.05128192305994846,
           'H2O': 0.049958852566446776,
           'N2': 0.6721708175587002,
           'O2': 0.01863604760274963,
           'CO': 0.12208223237385288,
           'H2': 0.016497618694114122,
           'H': 0.020074210589464313,
           'O': 0.015243712784131586,
           'OH': 0.02037700232321525,
           'NO': 0.013677582447376801},
      'ni': {'CO2': 0.5916092969925416,
           'H2O': 0.5763458131404036,
           'N2': 7.754438232941029,
           'O2': 0.21499308846303877,
           'CO': 1.4083907030074583,
           'H2': 0.1903232955259162,
           'H': 0.23158432652047958,
           'O': 0.17585772267615443,
```

```

'OH': 0.23507745614688103,
'NO': 0.15779020078460856},
'N': 11.53641013619851,
'Cpom': 38.789372252198845,
'hm': 60786.47410478268,
'sm': 278.69416342509317,
'vm': 246.18564942755629,
'M': 27.056660116691273,
'x': 0.0},
8,
array([ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
       0.00000000e+00,  0.00000000e+00,  2.77555756e-16,  0.00000000e+00,
       0.00000000e+00, -4.44089210e-16]))
```

0.0.3 Adiabatic Flame Temperature TAC

```
[5]: subs = (
    Substance('ethanol'), (2,6,1,0),-235000.,232.444),
    Substance('gasoline'),(7,17,0,0),-208600.,466.514),
    Substance('ethane'),(2,6,0,0),52467.,219.330),
    Substance('n-hexane'),(6,14,0,0),-167300.,387.979),
    Substance('methane'),(1,4,0,0),-74873.,186.251),
    #(Substance(formula='H2'),(0,2,0,0),0.,0.), som não é nulo!
    Substance('acetylene'),(2,2,0,0),226731.,200.958)
)
```

```
Path absoluto da tabela: /home/emanuel/tabelaA1A3.db
```

```
[6]: plt.figure()
for sub,coefs,hfom,som in subs:
    alfa,beta,gama,delta = coefs
    print (sub.name)

    To,Po = 298.,0.10132 # 1 atm
    Ts = []
    # Para várias relações de equivalência:
    phis = np.r_[0.2:0.9:20j,0.95,1.0,1.025,
                1.05,1.08,1.12:1.4:5j]

    for phi in phis:
        def residuo(T,args):
```

```

    props,ite,res = equilibrium(alfa,beta,
                                gama,delta,
                                phi,T,P, jacob = 1)
    fracs,n,N,cpm,hm,sm,vm,M,x = props.values()
    return N*hm - hfom

T,ite,args = robustNewton(residuo,2000.)
Ts.append(T)
Ts = np.array(Ts)

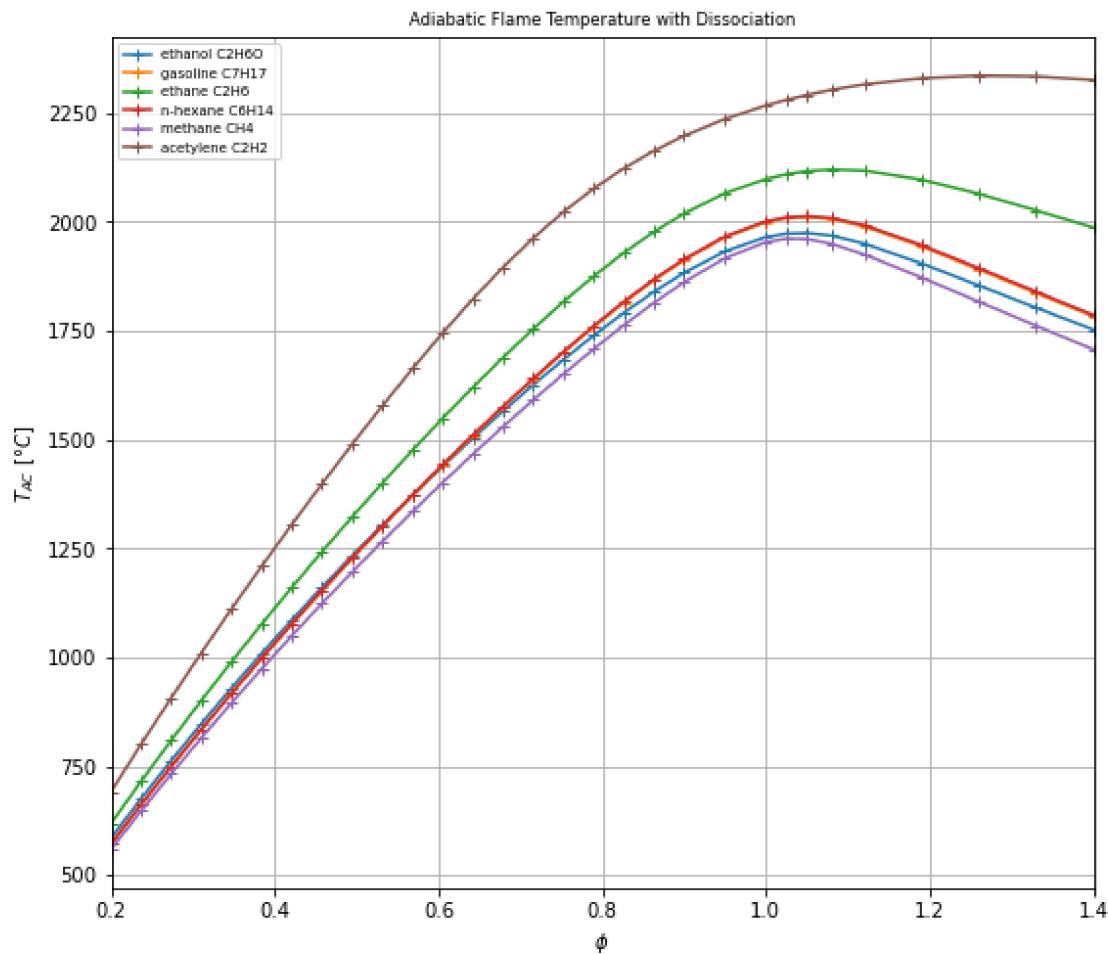
Ts

plt.plot(phis,Ts-273.15,'+-', label='%'s %s'%(sub.name,sub.formula))
plt.ylabel('$T_{AC} \ [^{\circ}\text{C}]$')

plt.grid(True)
plt.title('Adiabatic Flame Temperature with Dissociation',
          fontsize='small')
plt.xlabel('$\phi$')
plt.xlim(phis[0],phis[-1])
plt.xticks()
plt.legend(loc='best',fontsize='x-small');

```

ethanol
gasoline
ethane
n-hexane
methane
acetylene



[]:

Ferguson figura 3-2 rev2020

December 27, 2020

```
[1]: %matplotlib inline
```

```
[2]: import matplotlib.pyplot as plt
from equilibriumNR import equilibrium
import numpy as np

P = 5 # MPa ou 50 bar
alfa,beta,gama,delta = 8,18,0,0
x=0.
jacob = True # None
Ts = np.r_[1000.:4000:61j]
print (Ts)

phi_range = [0.8,1.,1.2]

plt.figure(figsize=(9,7))
for k,phi in enumerate(phi_range):
    plt.subplot(1,len(phi_range),k+1,aspect='auto')

    fracs_dict = {}

    for i,T in enumerate(Ts):

        props,ite,F = equilibrium(alfa,beta,gama,delta,phi,
                                  T,P,x,
                                  nitermax = 200,
                                  jacob = jacob)
        y,n,N,cp,h,s,v,M,x = props.values()
        fracs_dict[i] = y

##      print fracs_dict

    for item in ['CO2', 'H2O', 'N2', 'O2',
                 'CO', 'H2', 'H', 'O', 'OH', 'NO']:

        fracs = np.array([fracs_dict[i][item] for i,T in enumerate(Ts)])
```

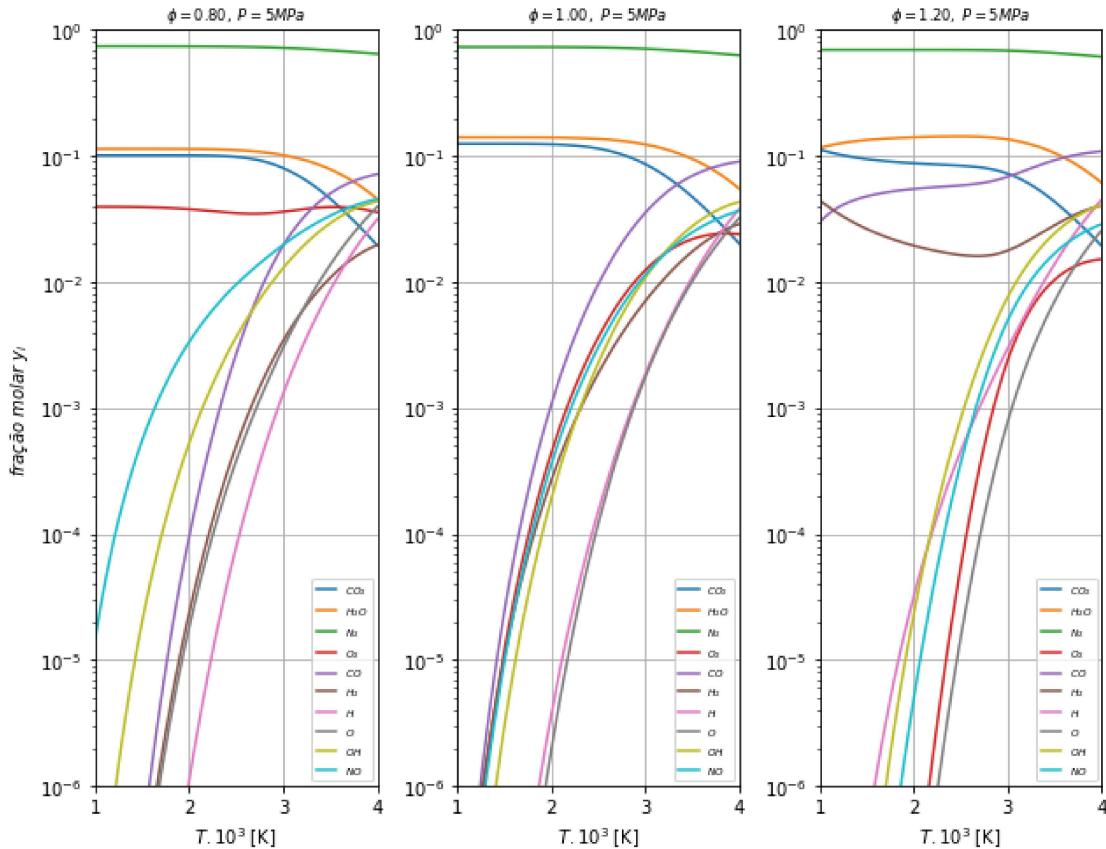
```

plt.semilogy(Ts/1000.,fracs,label=item)
plt.title('$\phi=%2f, \ P=%g \ MPa$'%(phi,P),fontsize='small')
plt.xlabel('$T \cdot 10^3 \ [K]$')
if k == 0:
    plt.ylabel(u'$\frac{\text{molar}}{\text{molar}_i}$')

plt.legend(['CO_2', 'H_2O', 'N_2', 'O_2',
           'CO$', 'H_2$', 'H$', 'O$', 'OH$', 'NO$'],
           loc='best', fontsize='xx-small')
plt.ylim(bottom=1e-6,top=1.)
plt.xlim(Ts[0]/1e3,Ts[-1]/1e3)
plt.tight_layout()
plt.grid()

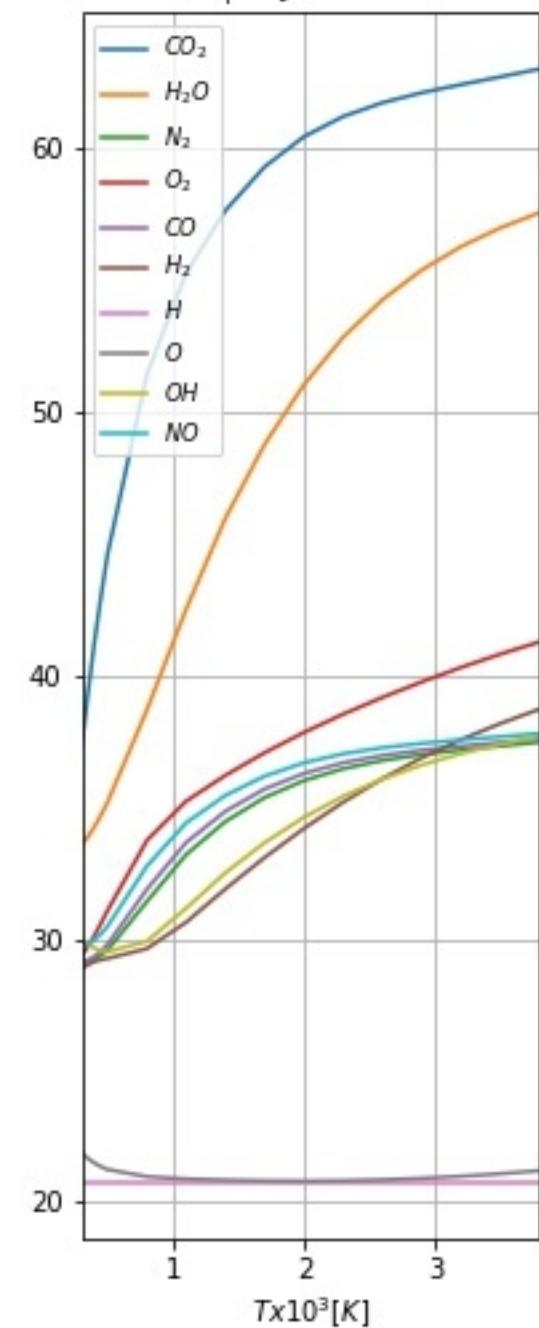
```

[1000. 1050. 1100. 1150. 1200. 1250. 1300. 1350. 1400. 1450. 1500. 1550.
1600. 1650. 1700. 1750. 1800. 1850. 1900. 1950. 2000. 2050. 2100. 2150.
2200. 2250. 2300. 2350. 2400. 2450. 2500. 2550. 2600. 2650. 2700. 2750.
2800. 2850. 2900. 2950. 3000. 3050. 3100. 3150. 3200. 3250. 3300. 3350.
3400. 3450. 3500. 3550. 3600. 3650. 3700. 3750. 3800. 3850. 3900. 3950.
4000.]

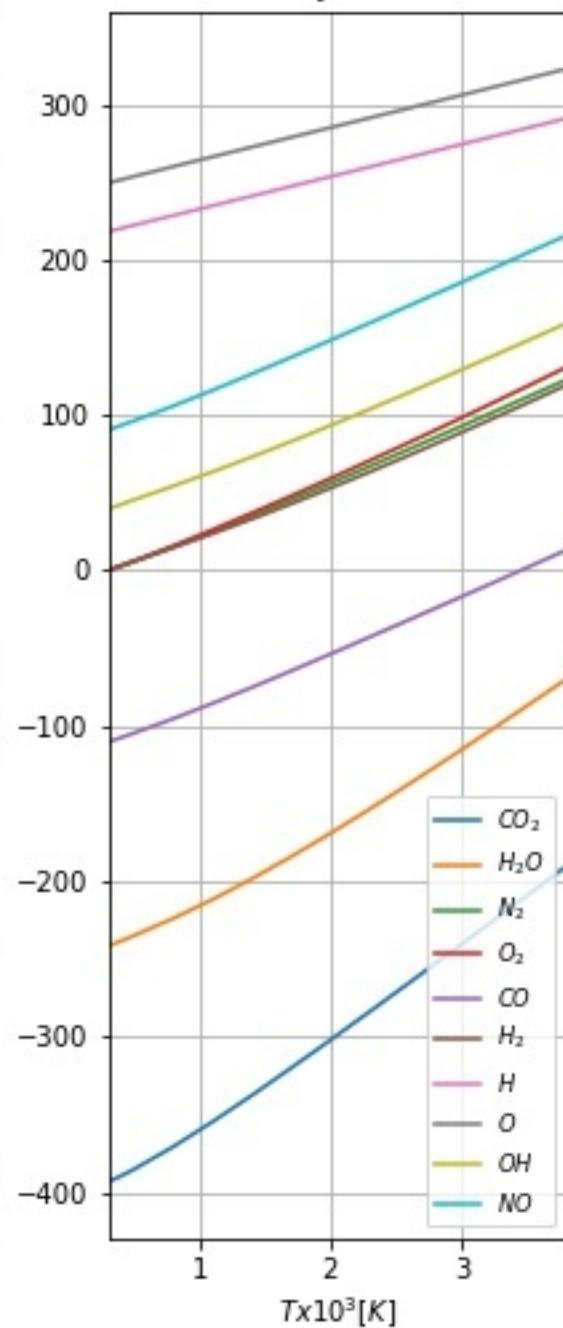


Dez produtos de combustão (gases perfeitos):
Cp_o,entalpia,entropia e K_i(T) (Ferguson)

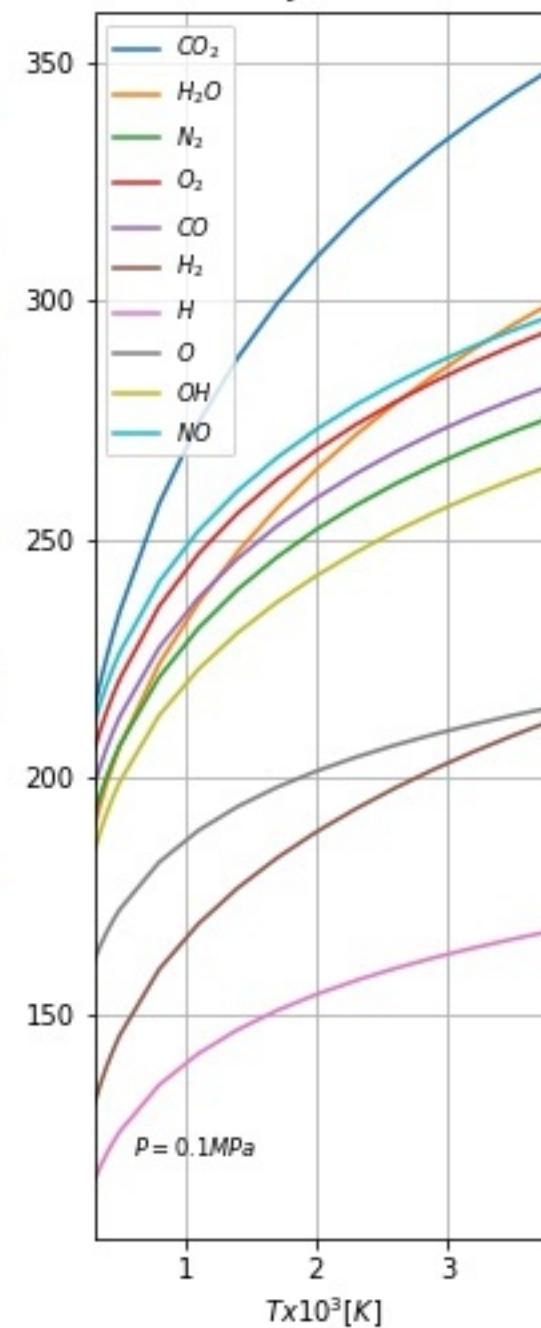
$C_{p0} [kJ/kmolK]$



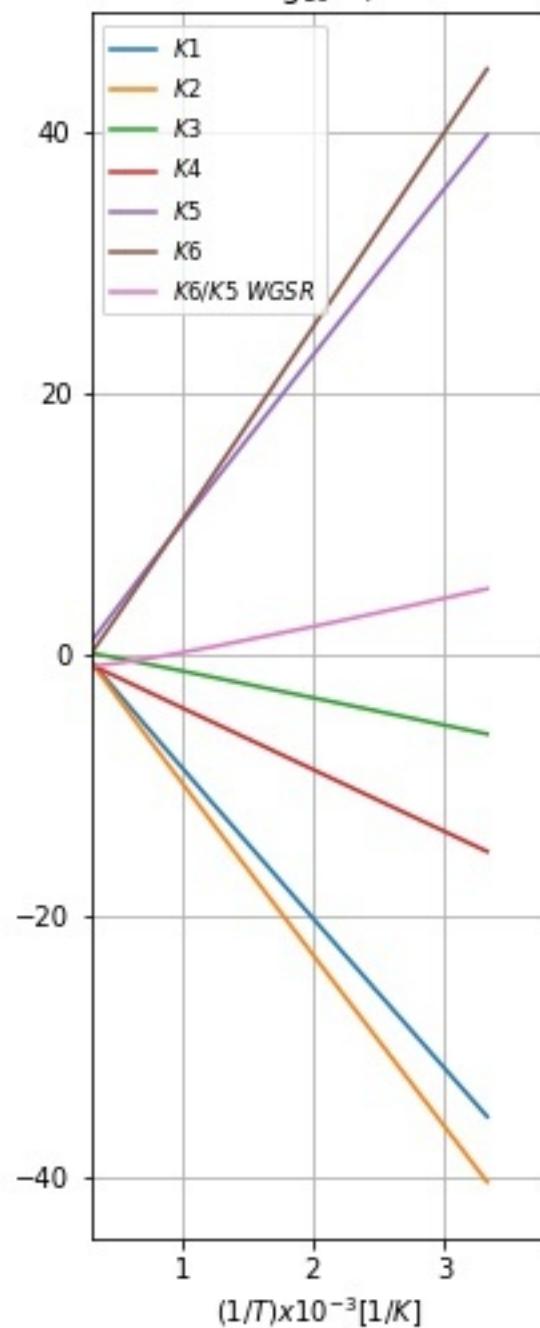
$h^* [MJ/kmol]$



$s^* [kJ/kmolK]$



$\log_{10}(K_i)$



equilibriumNR rev2019

December 27, 2020

```
[1]: ### %load '/home/erwoiski/MyScripts3/equilibriumNR.py'
"""
DTERMINAÇÃO DO EQUILÍBRIO DE DEZ PRODUTOS DE COMBUSTÃO
E DAS PROPRIEDADES TERMODINÂMICAS DA MISTURA
Base: entalpia de formação e entropia absoluta a 298.15K e
100 kPa.
Autor: Emanuel R Woiski UNESP - Ilha Solteira 2019
"""

from numpy.linalg import solve
import numpy as np
from numpy import log as ln
from constants import *
from robustNR_args import robustNewton
import proptermo2 as p2
import sys

def equilibrium(alfa,beta,gama,delta,phi,T,P,x=0.,
                Po= 0.1,nitermax = 200,xtol=1.e-8, jacob = 1):
    """
    Para 1 kmol de combustível composto de C alfa H beta O gama e N delta e
    razão de equivalência phi, acrescentando-se x kmoles de H2O/kmol de combustível
    →nos
    reagentes, esta função calcula o equilíbrio dos dez produtos de combustão mais
    comuns, à temperatura e pressão T (de 300 a 3000K) e P (MPa):
    CO2, H2O, N2, O2, CO, H2, H, O, OH e NO,
    usando 6 equações de equilíbrio, com jacobiano algébrico ou numérico.
    A função retorna um dicionário com as frações molares,
    o número de kmoles de cada produto por kmol de combustível,
    o número total de kmoles por kmol de combustível,
    além das propriedades *molares* da mistura (por kmol de mistura),
    considerada como gás perfeito:
    calor específico, entalpia, entropia, volume específico,
    massa molecular, bem como o número de iterações e o vetor dos resíduos
    (que devem ser próximos de zero) do sistema de equações,
    cujas raízes devem ser obtidas por um esquema de Newton-Raphson modificado
    que denominamos de robustNewton.
    """
```

*NOTA IMPORTANTE: Se for utilizado jacobiano *numérico* E se phi for *igual* a 1.0 e só nesse caso, o jacobiano resultará *singular* para valores de T abaixo de 900K... (versão 15-11-2019).*

"""

```
As = alfa + beta/4 - gama/2 # relação ar-combustível molar estequiométrica

PPo = P/Po          # relação de pressões, onde Po é a pressão de referência
# (kPa)

K = p2.calcula_constEq(T) # constantes de equilíbrio em função da
# temperatura T

# definições por conveniência:
LK1 = ln(K[1]*K[1]/PPo)
LK2 = ln(K[2]*K[2]/PPo)
LK3 = ln(K[3]*K[3])
LK4 = ln(K[4]*K[4])
LK5 = ln(K[5]*K[5]*PPo)
LK6 = ln(K[6]*K[6]*PPo)
asphi = 2*As/phi

def equations(n,args=None):
    """
    Monta o sistema de equações, cujas raízes
    devem ser obtidas - n é o vetor número de moles
    """
    n1,n2,n3,n4,n5,n6,n7,n8,n9,n10 = n
    ln1,ln2,ln3,ln4,ln5,ln6,ln7,ln8,ln9,ln10 = ln(n)
    N = n.sum()
    lnN = ln(N)

    eqs = np.array((
        (n1 + n5) - alfa,
        (2*n2 + 2*n6 + n7 + n9) - beta - 2*x, # x moles de H2O
        (2*n1 + n2 + 2*n4 + n5 + n8 + n9 + n10) - (gama + asphi + x),
        (2*n3 + n10) - (delta + 3.76*asphi),
        2*ln7 - ln6 - lnN - LK1,
        2*ln8 - ln4 - lnN - LK2,
        2*ln9 - ln4 - ln6 - LK3,
        2*ln10 - ln4 - ln3 - LK4,
        2*ln2 - ln4 - 2*ln6 + lnN - LK5,
        2*ln1 - ln4 - 2*ln5 + lnN - LK6,
        ))
    return eqs
```

```

def jacobiano(n,args=None):
    """
    Calcula o jacobiano do sistema de equações -
    As derivadas são calculadas ao longo das linhas.
    """
    n1,n2,n3,n4,n5,n6,n7,n8,n9,n10 = n
    N = n.sum()

    jac = array((
        [1,0,0,0,1,0,0,0,0,0],
        [0,2,0,0,0,2,1,0,1,0],
        [2,1,0,2,1,0,0,1,1,1],
        [0,0,2,0,0,0,0,0,0,1],
        [0,0,0,0,0,-1/n6,2/n7,0,0,0],
        [0,0,0,-1/n4,0,0,0,2/n8,0,0],
        [0,0,0,-1/n4,0,-1/n6,0,0,2/n9,0],
        [0,0,-1/n3,-1/n4,0,0,0,0,0,2/n10],
        [0,2/n2,0,-1/n4,0,-2/n6,0,0,0,0],
        [2/n1,0,0,-1/n4,-2/n5,0,0,0,0,0]
    ))
    jac += np.reshape(
        np.array([0,0,0,0,-1/N,-1/N,0,0,1/N,1/N]),(10,1)
    )
    return jac

def calcula_props_mist(T,P,y): # MPa
    """
    Através dessa função são determinadas as propriedades da mistura de
    produtos de combustão, como gás perfeito, a T e P, conhecidas as
    respectivas frações molares do equilíbrio (dicionário y).
    """
    cp,h,s,Ms = p2.calcula_props(T,P)

    hmist = (h['CO2']*y['CO2'] + h['H2O']*y['H2O'] +
              h['N2']*y['N2'] + h['O2']*y['O2'] +
              h['CO']*y['CO'] + h['H2']*y['H2'] +
              h['H']*y['H'] + h['O']*y['O'] +
              h['OH']*y['OH'] + h['NO']*y['NO'])

    Mmist = (Ms['CO2']*y['CO2'] + Ms['H2O']*y['H2O'] +
              Ms['N2']*y['N2'] + Ms['O2']*y['O2'] +
              Ms['CO']*y['CO'] + Ms['H2']*y['H2'] +
              Ms['H']*y['H'] + Ms['O']*y['O'] +
              Ms['OH']*y['OH'] + Ms['NO']*y['NO'])

```

```

smist = (s['CO2']*y['CO2'] + s['H2O']*y['H2O'] +
          s['N2']*y['N2'] + s['O2']*y['O2'] +
          s['CO']*y['CO'] + s['H2']*y['H2'] +
          s['H']*y['H'] + s['O']*y['O'] +
          s['OH']*y['OH'] + s['NO']*y['NO']) - \
Ru*(y['CO2']*ln(y['CO2']) + y['H2O']*ln(y['H2O']) +
   y['N2']*ln(y['N2']) + y['O2']*ln(y['O2']) +
   y['CO']*ln(y['CO']) + y['H2']*ln(y['H2']) +
   y['H']*ln(y['H']) + y['O']*ln(y['O']) +
   y['OH']*ln(y['OH']) + y['NO']*ln(y['NO']) )

cpmist = (cp['CO2']*y['CO2'] + cp['H2O']*y['H2O'] +
          cp['N2']*y['N2'] + cp['O2']*y['O2'] +
          cp['CO']*y['CO'] + cp['H2']*y['H2'] +
          cp['H']*y['H'] + cp['O']*y['O'] +
          cp['OH']*y['OH'] + cp['NO']*y['NO'])

vmist = Ru*T/(1000*P)

return cpmist,hmist,smist,vmist,Mmist

*****
# corpo da função equilibrium:
# condições iniciais:

n_init = np.ones((10))

if jacob is not None:
##    print 'Com Jacobiano...'
    jacob = jacobian

try:
    n,ite,F = robustNewton(equations,n_init,
                           jacob = jacob,nitermax = nitermax,
                           xtol = xtol,args=None)

##    print ite
except LinAlgError:
    print('T: ',T,'P: ',P,'phi: ', phi)
    print("Jacobiano singular ou não fornecido!")
    eval(input('Aperte RETURN para sair...'))
    sys.exit()

N = n.sum()
y = n/N
yd = dict(list(zip(subst,y)) )
nd = dict(list(zip(subst,n)) )

```

```

props = {}

cp,h,s,v,M = calcula_props_mist(T,P,yd)
props['yi'] = yd
props['ni'] = nd
props['N'] = N
props['Cpom'] = cp # kJ/kmol K de mistura de produtos
props['hm'] = h # kJ/kmol de mistura de produtos
props['sm'] = s # kJ/kmol K de mistura de produtos
props['vm'] = v # m3 por kmol de mistura de produtos
props['M'] = M
props['x'] = x

return props,ite,F

```

```

[11]: if __name__ == '__main__':
    print (equilibrium.__doc__, '\n')

    alfa,beta,gama,delta = 2.,6.,1,0 # fuel composition!
    phi = 1.0 # equivalence ratio
    T = 3000.
    P = 0.1 # MPa
    x = 0      # additional steam
    jacob = True

    if jacob is None:
        print('Sem Jacobiano...')
    else:
        print('Com Jacobiano...')

    props,ite,res = equilibrium(alfa,beta,gama,delta,phi,
                                T,P,x,
                                nitermax = 200,
                                jacob = jacob)

    M = props['M']
    print('niter:',ite)
    print('T: ',T,'P: ',P,'x:',x,'phi: ',phi)

    print('\n\nprops:\n',props,'\n\n')
    print('ntotal: ', props['N'], '\nfracs:\n',props['yi'])
    print('propriedades da mistura:\n')
    Cp: %f kJ/kgK, h: %f kJ/kg, s: %f kJ/kgK, v:%f m3/kg, M: %f kg/kmol'%
          (props['Cpom']/M,props['hm']/M,

```

```

    props['sm']/M,props['vm']/M,M)) # valores em massa!
print('resíduos:\n',res)

```

Para 1 kmol de combustível composto de C alfa H beta O gama e N delta e razão de equivalência phi, acrescentando-se x kmoles de H₂O/kmol de combustível nos

reagentes, esta função calcula o equilíbrio dos dez produtos de combustão mais comuns, à temperatura e pressão T (de 300 a 3000K) e P (MPa):

C_{O2}, H₂O, N₂, O₂, CO, H₂, H, O, OH e NO,

usando 6 equações de equilíbrio, com jacobiano algébrico ou numérico.

A função retorna um dicionário com as frações molares,

o número de kmoles de cada produto por kmol de combustível,

o número total de kmoles por kmol de combustível,

além das propriedades *molares* da mistura (por kmol de mistura),

considerada como gás perfeito:

calor específico, entalpia, entropia, volume específico,

massa molecular, bem como o número de iterações e o vetor dos resíduos

(que devem ser próximos de zero) do sistema de equações,

cujas raízes devem ser obtidas por um esquema de Newton-Raphson modificado

que denominamos de robustNewton.

NOTA IMPORTANTE: Se for utilizado jacobiano *numérico* E se phi for *igual* a 1.0 e só nesse caso, o jacobiano resultará *singular* para valores de T abaixo de 900K... (versão 15-11-2019).

Com Jacobiano...

niter: 7

T: 3000.0 P: 0.1 x: 0 phi: 1.0

props:

```

{'yi': {'C02': 0.03895995945242775, 'H2O': 0.10918553383239565, 'N2':
0.6228542912413446, 'O2': 0.03050636510705498, 'CO': 0.07296858421158428, 'H2':
0.028366331412034654, 'H': 0.026495799082061542, 'O': 0.019631644192616107,
'OH': 0.03418610142111397, 'NO': 0.01684539004736663}, 'ni': {'C02':
0.6961577123593792, 'H2O': 1.950986410761309, 'N2': 11.129498711446356, 'O2':
0.5451042979461829, 'CO': 1.3038422876406208, 'H2': 0.5068650137570793, 'H':
0.4734413263089858, 'O': 0.3507888792254195, 'OH': 0.6108558246542378, 'NO':
0.3010025771072882}, 'N': 17.868543041206856, 'Cpom': 39.43242723175766, 'hm':
60860.44506476328, 'sm': 278.67927850784724, 'vm': 249.4353, 'M':
25.634512086176276, 'x': 0}

```

```
ntotal: 17.868543041206856
fracs:
{'CO2': 0.03895995945242775, 'H2O': 0.10918553383239565, 'N2':
0.6228542912413446, 'O2': 0.03050636510705498, 'CO': 0.07296858421158428, 'H2':
0.028366331412034654, 'H': 0.026495799082061542, 'O': 0.019631644192616107,
'OH': 0.03418610142111397, 'NO': 0.01684539004736663}
propriedades da mistura:Cp: 1.538255 kJ/kgK, h: 2374.160462 kJ/kg, s: 10.871253
kJ/kgK, v:9.730449 m3/kg, M: 25.634512 kg/kmol
resíduos:
[ 0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00
2.62012634e-14 8.88178420e-16 2.81441537e-14 -1.33226763e-14
5.32907052e-14 1.77635684e-15]
```

[]:

[]: