

Tarea 2.1 | Absorción y arrastre, cálculo de etapas y altura de relleno

Esteban Richmond-Salazar

9/9/2019

Contents

1 Problema N° 1	1
1.1 Solución	4
2 Problema N° 2	5
2.1 Solución	11

1 Problema N° 1

Valor: 10 puntos

Referencia: Adaptado de problema 12.D8 Wankat (2008)

Se van a procesar 12 kmol/h de una solución acuosa concentrada de amoníaco en una columna de arrastre. La alimentación contiene 2 kmol/h de amoníaco y 10 kmol/h de agua. Se desea que la corriente de agua a la salida contenga una fracción molar de amoníaco de 0,015. El gas de arrastre es aire puro, con una tasa de flujo de 9 kmol/h. La operación es a 40 °C y a 100 kPa. Suponga que el aire es insoluble, que el agua es no volátil y que el separador de arrastre es isotérmico.

Pesos moleculares: Amoníaco = 17 u, agua = 18 u, aire = 29 u. Usando el método gráfico:

1. Calcule la cantidad de etapas de equilibrio necesarias, incluyendo la fracción.
2. Calcule la tasa mínima de flujo de aire.

Datos de equilibrio

```
X_W = [0.40, 0.30, 0.25, 0.20, 0.15, 0.10, 0.075, 0.05, 0.04, \
0.03, 0.025, 0.020, 0.016, 0.012, 0.0] # kg/kg, relación masa
p_A = [ 719, 454, 352, 260, 179, 110, 79.7, 51, 40.1, \
29.6, 24.4, 19.3, 15.3, 11.5, 0.0] # mmHg
```

html('<h2>Solución</h2>')

```

# Datos del enunciado
L_entra    = 12      # kmol/h
L_NH3_entra = 2      # kmol/h
L_s        = 10      # kmol/h
x_sale     = 0.015
Y_entra    = 0
G_s        = 9       # kmol/h
t          = 40      # °C
p_t        = 100     # kPa
M_NH3      = 17
M_H2O      = 18
M_aire     = 29

# Cálculo de relaciones mol
X_entra = L_NH3_entra/L_s
X_sale  = x_sale/(1-x_sale)

# Conversión de datos de equilibrio a relaciones molares
Xeq     = [X_W[i]*M_H2O/M_NH3 for i in range(len(X_W))]
Yeq     = [p_A[i]/(p_t*760/101.325 - p_A[i]) for i in range(len(p_A)\
)]

# Descarto los datos correspondientes a concentraciones mucho \
mayores a las de operación para ajustar mejor a un polinomio.
i       = 0
n_i     = 0
while i <= len(Xeq):
    if Xeq[i] <= X_entra:
        n_i=i-1
        i = len(Xeq)
    i += 1

datos_eq = zip([Xeq[i] for i in range(n_i, len(Xeq))],[Yeq[i] for i \
in range(n_i, len(Yeq))])

print "Ajuste de los datos de equilibrio"
print "-----"
# Se definen estas variables para ajustar a un polinomio de 3er \
grado
%var X, a1, a2, a3, m

modelo(X) = a1*X+a2*X^2+a3*X^3

ajuste    = find_fit(datos_eq, modelo, solution_dict = True)
Y_eq(X)   = modelo(a1=n(ajuste[a1], digits=4), a2=n(ajuste[a2], \
digits=4), a3=n(ajuste[a3], digits=4))
show(r'$Y^* = %s$' %Y_eq(X))

```

```

print "Parte a): etapas de equilibrio"
print "-----"

Y_op(X)    =  L_s/G_s*X - L_s/G_s*X_sale+Y_entra
Y_sale     =  Y_op(X_entra)

print "Y_sale = %s mol/mol" %numerical_approx(Y_sale, digits=3)

# Cálculo de las etapas ideales
XY_etapas = [(X_sale, Y_entra)]      # Primer punto de la \
    escalera (X,Y)

forget()
Xi        =  X_sale
while Xi < X_entra:
    Yi     =  Y_eq(Xi)
    XY_etapas.append((Xi, Yi))      # Guarda el punto (X,Y*)
    Xi     =  solve(Y_op(X)==Y_eq(Xi), X, to_poly_solve = True)[0].\
rhs()
    XY_etapas.append((Xi, Yi))      # Guarda el punto (X,Y)

N_p       =  (len(XY_etapas)-3)/2 +(X_entra-XY_etapas[len(XY_etapas)\
-2][0])/(XY_etapas[len(XY_etapas)-1][0]-XY_etapas[len(XY_etapas)\
-2][0])

# Gráfica
plot (Y_eq, xmin=0, ymin=0, title = 'Relaciones mol', axes_labels = \
['$X_{NH_3}$', '$Y_{NH_3}$'], legend_label='Curva de equilibrio', \
frame=True) \
+ plot(Y_op, (X_sale, X_entra), color='red', legend_label='Curva \
de operac.') \
+ scatter_plot(datos_eq, xmax = X_entra, ymax = Y_eq(X_entra)) \
+ line([(X_sale, 0), (X_sale, Y_entra), (0, Y_entra)], color='\
purple') \
+ line([(X_entra, 0), (X_entra, Y_eq(X_entra))], color='purple') \
\
+ line(XY_etapas, color='green', legend_label='Etapas') \
+ text (r'$N_p = $ %s' %N_p.n(digits=3), ((X_entra+X_sale)/1.5, (\
Y_entra+Y_sale)/4))

print "Parte b): flujo mínimo de aire"
print "-----"

assume(X>=X_sale); assume(X<=X_entra)      # Define los lí\
    mites de X dónde buscar la relación limitante

```

```

Y_op_lim(m, X) = m*X - m*X_sale+Y_entra # Línea de \
operación en términos de m = Ls/Gs
m_lim = solve(Y_op_lim(m, X)==Y_eq(X), m) [0].rhs() # Como hay \
varios puntos devuelve una función de X
forget() # Elimina los l\
límites de X puestos anteriormente

mX_lim = find_local_minimum(m_lim, X_sale, X_entra) # Devuelve [m_m\
ín, X(m_mín)]
Gs_min = numerical_approx(L_s/mX_lim[0], digits=3)

# Gráfica
plot (Y_eq, xmin=0, ymin=0, title = 'Flujo limitante', axes_labels =\
['$X_{NH_3}$', '$Y_{NH_3}$'], legend_label='Curva de equilibrio '\
, frame=True) \
+ plot(Y_op_lim(mX_lim[0], X), (X_sale,X_entra), color='red', \
legend_label='Curva de operac. limitante') \
+ scatter_plot(datos_eq, xmax = X_entra, ymax = Y_eq(X_entra)) \
+ line([(X_sale, 0), (X_sale, Y_entra), (0, Y_entra)], color='\
purple') \
+ line([(X_entra, 0), (X_entra, Y_eq(X_entra))], color='purple')\
\
+ text (r'$L_{s}/G_{s,min} = $ %s' %n(mX_lim[0], digits=3),((\
X_entra+X_sale)/1.5,(Y_entra+Y_sale)/1.5)) \
+ text (r'$G_{s,min} = $ %s kmol/h' %Gs_min,((X_entra+X_sale)\
/1.5,(Y_entra+Y_sale)/2))

# ***** Final del código *****

```

1.1 Solución

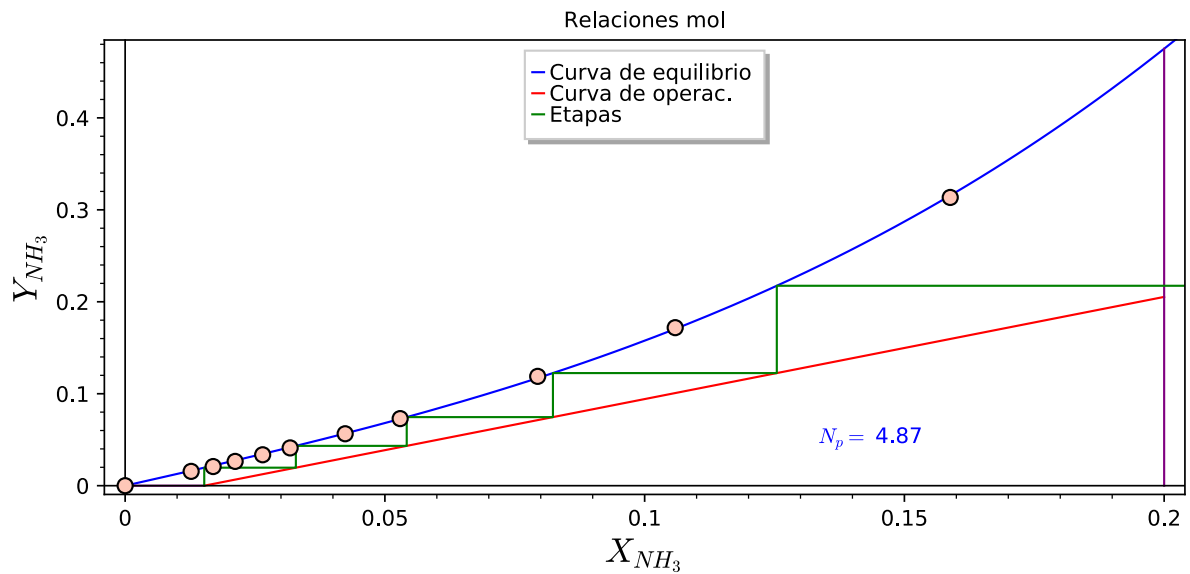
Ajuste de los datos de equilibrio

$$Y^* = 24.72 * X^3 + 0.5735 * X^2 + 1.272 * X$$

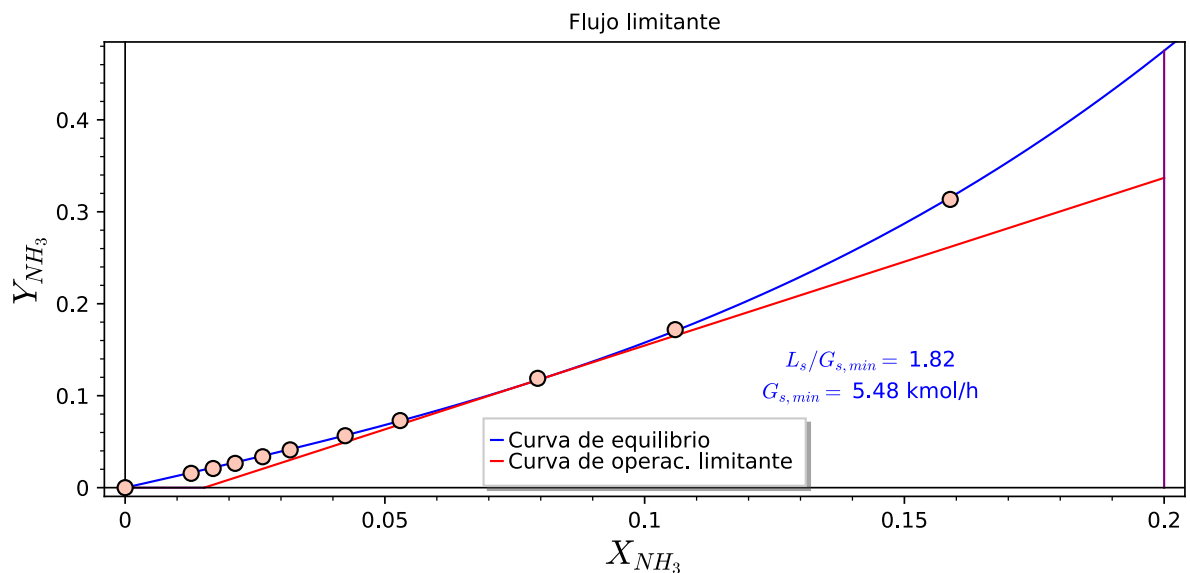
Parte a): etapas de equilibrio

$$Y_{\text{sale}} = 0.205 \text{ mol/mol}$$

2 Problema N° 2



Parte b): flujo mínimo de aire



2 Problema N° 2

Valor: 15 puntos

Referencia: Adaptado de Problema 10.6-6 Geankoplis (1998)

Una corriente de gas contiene 4,0 % molar de NH₃ y su contenido de amoníaco se reduce a 0,5 % molar en una torre de absorción rellena que opera a 20 °C y 1 atm. El flujo de agua pura de entrada es de 70,0 kmol/h y el flujo total de gas de entrada es de 60,0 kmol/h. El diámetro de la torre es 0,747 m. Los coeficientes de transferencia de masa de película son $F_{Ga} = 0,074 \text{ kmol}/(\text{s m}^3)$ y $F_{La} = 0,17 \text{ kmol}/(\text{s m}^3)$. Determine lo siguiente por método gráfico:

1. Calcule la altura de la torre usando coeficientes individuales de transferencia de masa, $k_y a$ (o F_{Ga}).
2. Calcule la altura de la torre usando coeficientes globales de transferencia de masa, $K_y a$ (o F_{OGa}).

```

html('<h2>Solución</h2>')

y_entra = 0.04
y_sale = 0.005
T = 20 # °C
p_t = 1 # atm
L_s = 70.0 # kmol/h
x_entra = 0
G_entra = 60.0 # kmol/h
D_c = 0.747 # m
FGa = 0.074 # kmol/(s m³)
FLa = 0.17 # kmol/(s m³)

# Datos de equilibrio
datos_x_eq = [0, 0.0208, 0.0258, 0.0309, 0.0405, 0.0503, \
              0.0737, 0.096, 0.137, 0.175, 0.210, 0.241, 0.297]
datos_p_NH3 = [0, 12.0, 15.0, 18.2, 24.9, 31.7, 50.0 \
               , 69.6, 114, 166, 227, 298, 470] # mmHg

# Conversión de presiones parciales a fracciones molares
datos_y_eq = [pA/(p_t*760) for pA in datos_p_NH3]

# Datos de equilibrio en relaciones molares
datos_X_eq = [x/(1-x) for x in datos_x_eq]
datos_Y_eq = [pA/(p_t*760-pA) for pA in datos_p_NH3]

Y_entra = y_entra/(1-y_entra)
Y_sale = y_sale/(1-y_sale)
X_entra = x_entra/(1-x_entra)

# Descarto los datos correspondientes a concentraciones mucho \
mayores a las de operación para ajustar mejor a un polinomio.
i = 0
n_i = len(datos_y_eq)
while i < n_i: #len(datos_y_eq):
    if datos_y_eq[i] >= y_entra:
        n_i = i
        #i = len(datos_y_eq)
    i += 1

Datos_eq = zip([datos_X_eq[i] for i in range(n_i+1)], [datos_Y_eq[i] \
for i in range(n_i+1)])

```

```

datos_eq = zip([datos_x_eq[i] for i in range(n_i+1)], [datos_y_eq[i] \
    for i in range(n_i+1)])

# Ajuste de datos de equilibrio a polinomio de 3er grado
%var x, X, xI, yI, a0, a1, a2, a3
modelo(x) = a1*x + a2*x^2 + a3*x^3

ajuste = find_fit(datos_eq, modelo, solution_dict=True) # \
    En fracciones molares
y_eq(x) = modelo(a1=ajuste[a1], a2=ajuste[a2], a3=ajuste[a3])

ajuste = find_fit(Datos_eq, modelo, solution_dict=True) # \
    En relaciones molares
Y_eq(X) = modelo(a1=ajuste[a1], a2=ajuste[a2], a3=ajuste[a3])

# Área de la columna
A_c = pi/4 * D_c^2 # m^2
show(r'$A_c = \frac{\pi}{4} D_c^2 = %s \text{ m}^2$' % (n(A_c, \
    digits=4)))

# Balance de masa
G_s = G_entra*(1-y_entra)
G_sale = G_s*(1+Y_sale)
Y_op(X) = L_s/G_s * X - L_s/G_s * X_entra + Y_sale
y_op(x) = Y_op(x/(1-x))/(1+Y_op(x/(1-x)))

X_sale = numerical_approx(solve(Y_op(X)==Y_entra, X)[0].rhs())
x_sale = X_sale/(1+X_sale)

print "G_s = %.4g kmol/h" %G_s
print "G_sale = %.4g kmol/h" %G_sale
print "X_sale = %.4g kmol/kmol" %X_sale
print "x_sale = %.4g kmol/kmol" %x_sale

# ***** Gráfica en relaciones molares *****
plot(Y_op(X), xmin=X_entra, ymin=Y_sale, xmax=X_sale, ymax=Y_entra, \
    legend_label='Curva de operac.', color='red') \
    + plot(Y_eq(X), xmin=0, ymin=0, xmax=datos_X_eq[n_i], ymax=\
    datos_Y_eq[n_i], title='Relaciones mol', axes_labels=['$X_{\
    NH_3}$', '$Y_{NH_3}$'], legend_label='Curva de equilibrio') \
    + scatter_plot(Datos_eq, ymin=0)

# Cálculo de las composiciones en la interfase
x_BM(xI, xL) = ((1-xL) - (1-xI))/ln((1-xL)/(1-xI))
y_BM(yI, yG) = ((1-yG) - (1-yI))/ln((1-yG)/(1-yI))

kpxa(xI, xL) = FLa/x_BM(xI, xL)

```

```

kpya(yI ,yG) = FGa/y_BM(yI ,yG)

n_seg = 4          # Cantidad de segmentos para calcular Delta_y
#@interact
#def interactive_function(n_seg = slider(start = 3, stop = 20, step \
    = 1, default=10, label = "Cantidad de segmentos")):

Delta_y  = (y_entra - y_sale)/n_seg
lista_y  = [y_sale+Delta_y*i for i in range(n_seg+1)]
lista_x  = [find_root(y_op == yop, x_entra, x_sale) for yop in \
    lista_y]

lista_xI = []
lista_yI = []

# Para cada par (x,y) calcula (xI ,yI)
for i in range(n_seg+1):
    xop = lista_x[i]
    yop = lista_y[i]

    # Valor preliminar de xI, yI
    sol_I = solve([yI == -FLa/FGa * xI + FLa/FGa * xop + yop, yI == \
        y_eq(xI)], xI,yI, solution_dict = True)
    xI0   = sol_I[0][xI]
    yI0   = sol_I[0][yI]

    # Itera para encontrar un mejor estimado de xI, yI
    err_x = 1      # Error para x (primer valor es arbitrario)
    err_y = 1      # Error para y
    tol   = 1e-16  # Tolerancia deseada
    it    = 0      # Número de iteraciones

    while it < 10 and (err_x > tol or err_y > tol):
        it += 1
        sol_I = solve([yI == -kpxa(xI0 ,xop)/kpya(yI0 ,yop) * xI + \
            kpxa(xI0 ,xop)/kpya(yI0 ,yop) * xop + yop, yI == y_eq(xI)], xI,yI, \
            solution_dict = True)
        err_x = abs(xI0 - sol_I[0][xI])
        err_y = abs(yI0 - sol_I[0][yI])
        xI0   = sol_I[0][xI]
        yI0   = sol_I[0][yI]

    lista_xI.append(xI0)
    lista_yI.append(yI0)
    if(it==10): print "Excesivas iteraciones"

```



```

# Composiciones en los extremos
x_tope = x_entra
x_fondo = x_sale
xI_tope = lista_xI[0]
xI_fondo = lista_xI[n_seg]
y_tope = y_sale
y_fondo = y_entra
yI_tope = lista_yI[0]
yI_fondo = lista_yI[n_seg]
yeq_tope = y_eq(x_tope)
yeq_fondo = y_eq(x_fondo)
xeq_tope = find_root(y_eq == y_tope, 0, 1)
xeq_fondo = find_root(y_eq == y_fondo, 0, 1)

print "***** Parte (a), usando coeficientes individuales *****"
# ***** Gráfica en fracciones molares *****
# Líneas que muestran pendiente -k'x/k'y
lxyI = line([(lista_x[0], lista_y[0]), (lista_xI[0], lista_yI[0])\
], color = 'gray')
for i in range(n_seg):
    lxyI = lxyI + line([(lista_x[i+1], lista_y[i+1]), (lista_xI[i\
+1], lista_yI[i+1])], color = 'gray')

show(plot(y_op(x), xmin=x_entra, ymin=y_sale, xmax=x_sale, ymax=\
y_entra, legend_label='Curva de operac.', color = 'red') \
+ plot(y_eq(x), xmin=0, ymin=0, xmax=datos_x_eq[n_i], ymax=\
datos_y_eq[n_i], title = 'Fracciones mol', axes_labels = ['$x_{\
NH_3}$', '$y_{NH_3}$'], legend_label='Curva de equilibrio') \
+ scatter_plot(datos_eq, ymin=0) \
+ lxyI )

f = [1/(lista_y[i]-lista_yI[i]) for i in range(n_seg+1)]
datos_f = zip(lista_y, f)

f_int = spline(datos_f)
integral_f = numerical_integral(f_int, y_sale, y_entra)
corr = 1/2*ln((1-y_sale)/(1-y_entra))
N_tG = integral_f[0]+corr

show(plot(f_int, xmin=y_sale, xmax=y_entra, axes_labels = ['$y_{NH_3}\
}$', '$\frac{1}{y_{NH_3}-y_{NH_3, I}}$'], fill = 'axis') \
+ scatter_plot(datos_f) )

show (r'$N_{tG} = \int_{y_{sale}}^{y_{entra}} \frac{1}{y-y_I} + \frac{1}{2} \ln \frac{1-y_{entra}}{1-y_{sale}} = %s + %s = %s$' \
%(n(integral_f[0], digits=4), n(corr, digits=2), n(N_tG, digits=4))\

```

```

)

H_tG = (G_entra/(3600*kpya(x_fondo,xI_fondo)*A_c) + G_sale/(3600*\
    kpya(x_tope,xI_tope)*A_c))/2 # m
h_c = H_tG * N_tG # m

show (r'$H_{tG,prom} = %s \text{ m}$' %(n(H_tG, digits=4)))
show (r'$h_{c} = H_{tG} N_{tG} = %s \text{ m}$' %(n(h_c, digits=3)))

print "***** Parte (b), usando coeficientes globales *****"
# ***** Gráfica en fracciones molares *****
# Líneas verticales
lxyI = line([(lista_x[0], lista_y[0]), (lista_x[0], y_eq(lista_x\
    [0]))], color = 'gray')
for i in range(n_seg):
    lxyI = lxyI + line([(lista_x[i+1], lista_y[i+1]), (lista_x[i+1],\
        y_eq(lista_x[i+1]))], color = 'gray')

show(plot(y_op, xmin=x_entra, ymin=y_sale, xmax=x_sale, ymax=y_entra\
    , legend_label='Curva de operac.', color = 'red') \
    + plot(y_eq(x), xmin=0, ymin=0, xmax=datos_x_eq[n_i], ymax=\
    datos_y_eq[n_i], title = 'Fracciones mol', axes_labels = ['$x_{\
    NH_3}$', '$y_{NH_3}$'], legend_label='Curva de equilibrio') \
    + scatter_plot(datos_eq, ymin=0) \
    + lxyI )

f = [1/(lista_y[i]-y_eq(lista_x[i])) for i in range(n_seg+1)]
datos_f = zip(lista_y, f)

f_int = spline(datos_f)

integral_f = numerical_integral(f_int, y_sale, y_entra)
corr = 1/2*ln((1-y_sale)/(1-y_entra))
N_tOG = integral_f[0]+corr

show( plot(f_int, xmin=y_sale, xmax=y_entra, axes_labels = ['$y_{\
    NH_3}$', '$\frac{1}{y_{NH_3}-y_{NH_3}^*}$'], fill = 'axis') \
    + scatter_plot(datos_f))

show (r'$N_{tOG} = \int_{y_{sale}}^{y_{entra}} \{\frac{1}{y-y^*}\} + \frac{1}{2} \ln \{\frac{1-y_{entra}}{1-y_{sale}}\} = %s + %s = %s$' \
    %(n(integral_f[0], digits=4),n(corr, digits=2),n(N_tOG, digits=4)\
    ))

# Cálculo del coeficiente global
show(r'$\frac{1}{F_{OG}} = \frac{1}{F_G} \frac{(1-y_A)_{iM}}{(1-y_A)\

```

```

_{*M}} + \frac{m\prime(1-x_A)_{iM}}{F_L (1-y_A)_{*M}}$'

m_tope = (yI_tope - yeq_tope) / (xI_tope - x_tope)
m_fondo = (yI_fondo - yeq_fondo) / (xI_fondo - x_fondo)

FOGa_tope = 1 / (1 / FGa * y_BM(yI_tope, y_tope) / y_BM(yeq_tope, y_tope) \
) + m_tope / FLa * x_BM(xI_tope, x_tope) / y_BM(yeq_tope, y_tope) )
FOGa_fondo = 1 / (1 / FGa * y_BM(yI_fondo, y_fondo) / y_BM(yeq_fondo, \
y_fondo) + m_fondo / FLa * x_BM(xI_fondo, x_fondo) / y_BM(yeq_fondo, \
y_fondo))

show (r '$m\prime_{tope} = %s$' %n(m_tope, digits=4))
show (r '$(F_{OG})a_{tope} = %s$ kmol/(s m³)' %n(FOGa_tope, digits\
=4))
show (r '$m\prime_{fondo} = %s$' %n(m_fondo, digits=4))
show (r '$(F_{OG})a_{fondo} = %s$ kmol/(s m³)' %n(FOGa_fondo, digits\
=4))

H_tOG = (G_entra / (3600 * FOGa_fondo * A_c) + G_sale / (3600 * FOGa_tope * \
A_c)) / 2 # m, promediando en los extremos
h_c = H_tOG * N_tOG # m

show (r '$H_{tOG, prom} = %s \text{ m}$' %n(H_tOG, digits=4))
show (r '$h_c = H_{tOG} N_{tOG} = %s \text{ m}$' %n(h_c, digits=3)\
))

# ***** Final del código *****

```

2.1 Solución

$$A_c = \frac{\pi}{4} D_c^2 = 0.4383 \text{ m}^2$$

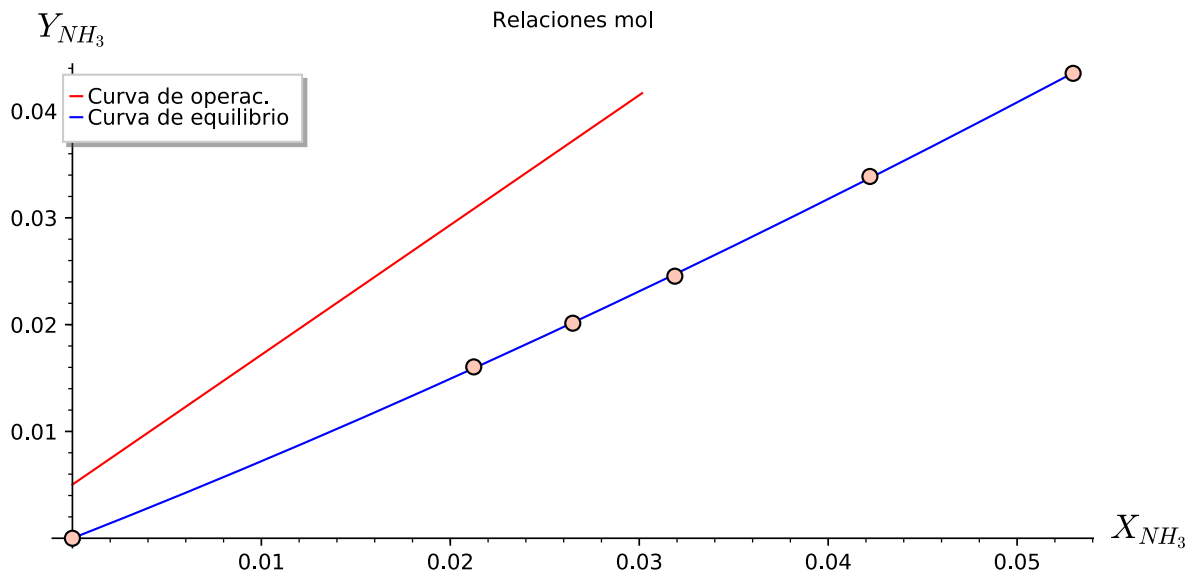
$$G_s = 57.6 \text{ kmol/h}$$

$$G_{\text{sale}} = 57.89 \text{ kmol/h}$$

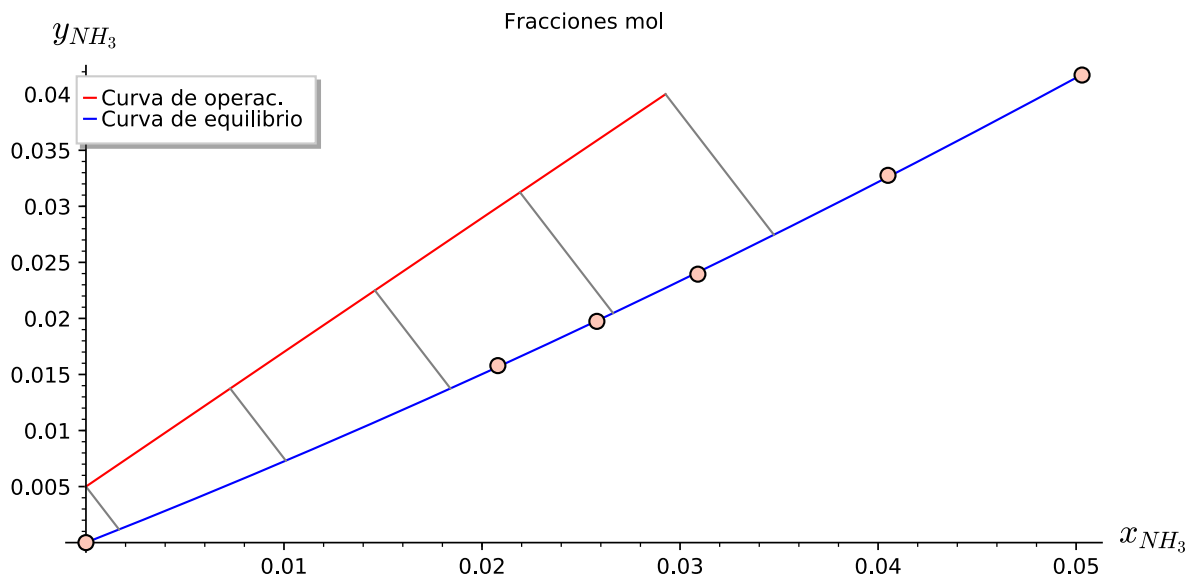
$$X_{\text{sale}} = 0.03015 \text{ kmol/kmol}$$

$$x_{\text{sale}} = 0.02927 \text{ kmol/kmol}$$

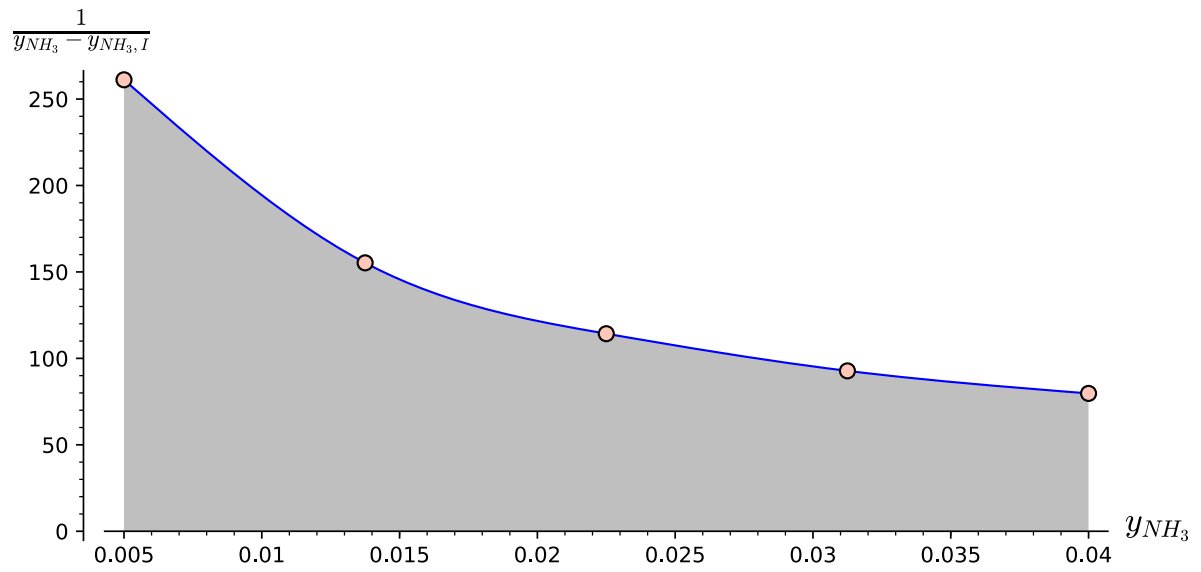
2 Problema N° 2



***** Parte (a), usando coeficientes individuales *****



2 Problema N° 2

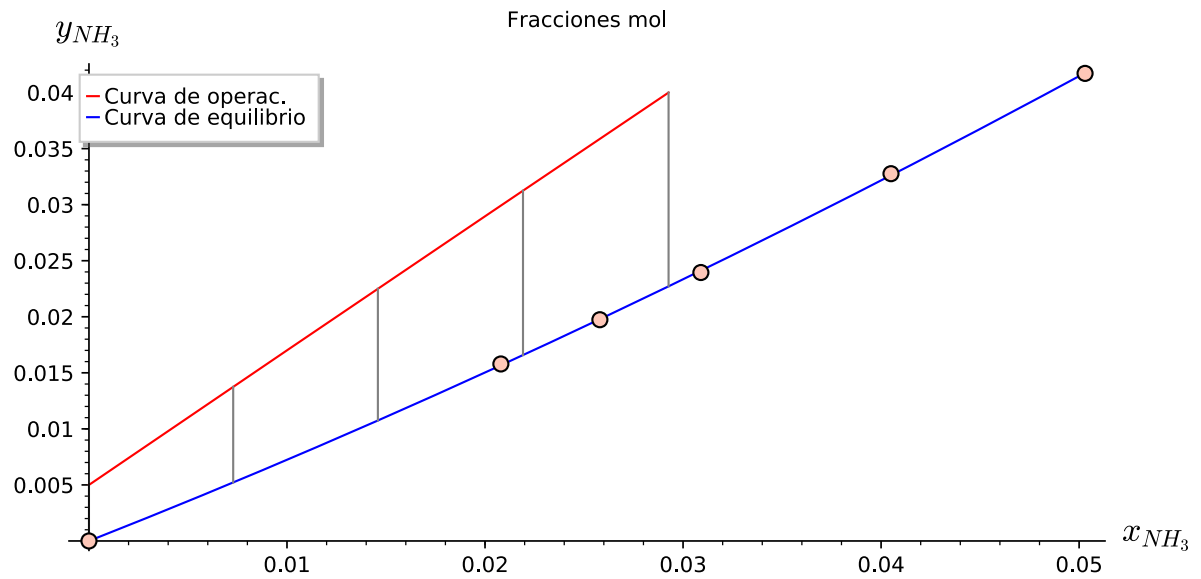


$$N_{tG} = \int_{y_{sale}}^{y_{entra}} \frac{1}{y-y_I} + \frac{1}{2} \ln \frac{1-y_{entra}}{1-y_{sale}} = 4.580 + 0.018 = 4.598$$

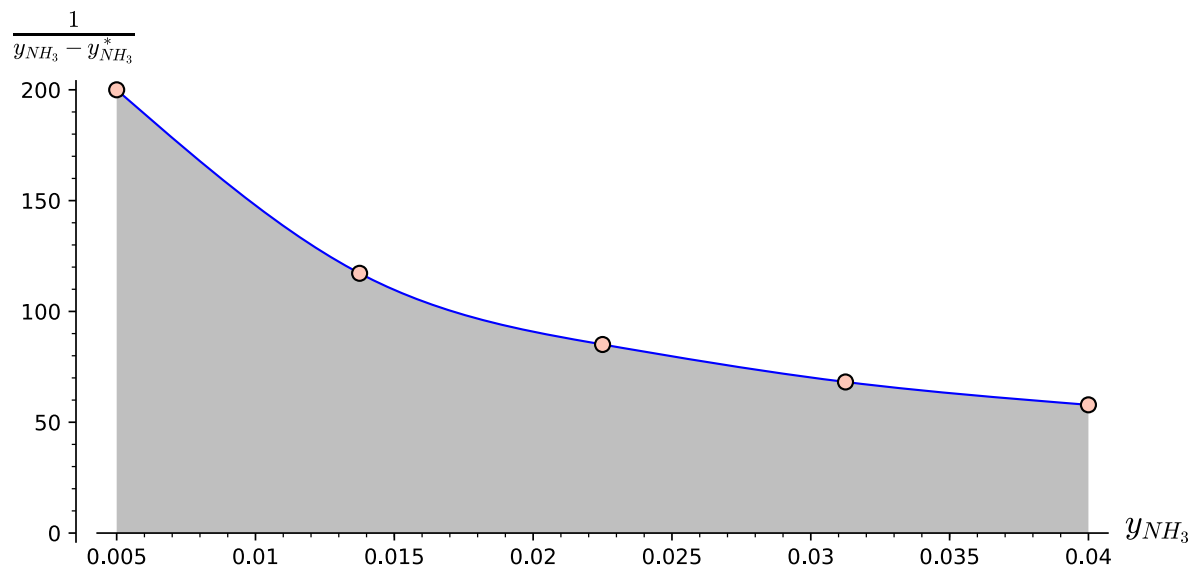
$$H_{tG,prom} = 0.4964 \text{ m}$$

$$h_c = H_{tG} N_{tG} = 2.28 \text{ m}$$

***** Parte (b), usando coeficientes globales *****



2 Problema N° 2



$$N_{tOG} = \int_{y_{sale}}^{y_{entra}} \frac{1}{y-y^*} + \frac{1}{2} \ln \frac{1-y_{entra}}{1-y_{sale}} = 3.431 + 0.018 = 3.449$$

$$\frac{1}{F_{OG}} = \frac{1}{F_G} \frac{(1-y_A)iM}{(1-y_A)^*M} + \frac{m(1-x_A)iM}{F_L(1-y_A)^*M}$$

$$m_{tope} = 0.7006$$

$$(FOGa)_{tope} = 0.05671 \text{ kmol}/(\text{s m}^3)$$

$$m_{fondo} = 0.8671$$

$$(FOGa)_{fondo} = 0.05383 \text{ kmol}/(\text{s m}^3)$$

$$H_{tOG,prom} = 0.6768 \text{ m}$$

$$h_c = H_{tOG} N_{tOG} = 2.33 \text{ m}$$