```
In [2]: import numpy as np
        import matplotlib.pyplot as plt
        from mpl_toolkits.mplot3d import Axes3D
        import matplotlib as mpl
        from matplotlib import rcParams
        %matplotlib inline
        #rcParams["figure.figsize"] = (11,7)
        #rcParams['font.family'] = 'serif'
        #rcParams['font.size'] = 14
```

Fluido 1 en dirección x lateral a tubos (gas)

Fluido 2 en dirección y, area transversal de tubos en plano xz (oil)

```
In [ ]:
```

Parámetros Intercambiador

```
In [3]: def propiedades_tubos():
            kt = 40
            do = 48.26
            di = 40.89
            do *= 10**(-3)
            di *= 10**(-3)
            ro = do/2
            ri = di/2
            return kt,do,di,ro,ri

        def propiedades_aletas():
            kf = 386
            de = 79
            t = 1
            sigmaf = 275
            de *= 10**(-3)
            t *= 10**(-3)
            re = de/2
            N_aletas = sigmaf*Ly
            return kf,de,t,sigmaf,re,N_aletas


        def calcular_etaf():
            r_pp = re/ro
            Le = re - ro + t/2
            m_pp = np.sqrt(2*h1/kf*t)
            if r_pp<=2:
                b = 0.9107 + 0.0893*r_pp
            else:
                b = 0.9706 + 0.17125*np.log(r_pp)
            n_pp = np.exp(0.13*m_pp*Le-1.3863)
            phi_pp = m_pp*Le*(r_pp**n_pp)
            if phi_pp <= 0.6 + 2.257*r_pp**(-0.445):
                eta_0 = np.tanh(phi_pp)/phi_pp
            else:
                eta_0 = r_pp**(-0.246)*(m_pp*Le)**(-b)
            Ab = 2*np.pi*ro*(Ly - t*N_aletas)
            A0 = np.pi*re**2-np.pi*ro**2 + 2*np.pi*re*t
            eta_f = (Ab + eta_0*N_aletas*A0)/(Ab + N_aletas*A0)
            return eta_f
```

```
In [ ]:
```

Propiedades de los gases

```
In [4]: def cp_gas(T):
            a = 1.0147 + 3.3859*10**(-4)*T - 1.6631*10**(-6)*(T**2) + 3.5452*10**(-9)*T**3
            b = - 2.8851*10**(-12)*T**4 + 8.1731*10**(-16)*T**5
            return (a+b)*10**3

        def viscosidad_gas(T):
            a = 1.751*10**(-6) + 6.4458*10**(-8)*T - 4.1435*10**(-11)*T**2
            b = 1.8385*10**(-14)*T**3 - 3.2051*10**(-18)*T**4
            return a+b

        def conductividad_gas(T):
            a = 1.9955*10**(-4) + 9.9744*10**(-5)*T - 4.1435*10**(-8)*T**2
            b = 1.8385*10**(-14)*T**3 - 3.2051*10**(-18)*T**4
            return a+b

        def densidad_gas(T):
            P = 120
            R = 0.297
            return P/(R*T)

        def Pr_gas(T):
            return cp_gas(T)*viscosidad_gas(T)/conductividad_gas(T)
```

```
In [5]: #h1 constantee
        def calcular_h1():
            u_max = m1/(rho1*Ly*(Nf*(Z-do) - sigmaf*(de-do)*t))
            Re = rho1*u_max*do/miu1
            if 0<= Re and Re < 10**2:
                C = 0.9
                alpha = 0.4
                beta = 0.36
            elif 10**2 <= Re and Re < 10**3:
                C = 0.52
                alpha = 0.5
                beta = 0.36
            elif 10**3 <= Re and Re < 2*10**5:
                C = 0.27
                alpha = 0.63
                beta = 0.36
            elif 2*10**5 <= Re and Re < 2*10**6:
                C = 0.033
                alpha = 0.8
                beta = 0.4
            else:
                print("Fuera de rango")
                print(Re)

            Nu = C*Re**alpha*Pr1**beta
            h = k1*Nu/do
            return h
```

Propiedades del aceite

```
In [6]: def cp_oil(T):
            return (0.81554 + 0.00364*T)*10**3

        def densidad_oil(T):
            return 1040.3 - 0.60477*T

        def viscosidad_oil(T):
            a = 81.541 - 0.61584*T + 1.7943*10**(-3)*T**2
            b = -2.3678*10**(-6)*T**3 + 1.1763*10**(-9)*T**4
            return np.exp(a+b)*10**(-3)

        def conductividad_oil(T):
            a = -7.5722*10**(-2) + 2.5952*10**(-3)*T - 1.2033*10**(-5)*T**2
            b = 2.653*10**(-8)*T**3 - 2.8663*10**(-11)*T**4 + 1.2174*10**(-14)*T**5
            return a+b

        def Pr_oil(T):
            return cp_oil(T)*viscosidad_oil(T)/conductividad_oil(T)
```

```
In [13]: def h2_ij(i,j):
             Re = 4*m2[i]/(np.pi*di*miu2)
             f = 0.00512 + 0.4572*Re**(-0.311)
             Nu = ((f/8)*(Re-1000)*Pr_oil(T2[i,j]))/(1+12.7*(f/8)**0.5*(Pr_oil(T2[i,j])**(2/3)-1))
             h = conductividad_oil(T2[i,j])*Nu/di
             return h
```

```
In [8]: def calcular_constantes():
            dy = Ly/Ny
            dA1b = 2*np.pi*ro*(dy-N_aletas*t)
            dA1f = 2*np.pi*N_aletas*(re**2-ro**2+re*t)
            dA1 = dA1b + dA1f
            dA2 = 2*np.pi*ri*dy
            Rw = np.log(ro/ri)/(2*np.pi*kt*dy)
            return dy,dA1,dA2,Rw

        def calcular_Uij(i,j):
            U = (dA2/(h1*eta_f*dA1) + dA2*Rw + 1/h2_ij(i,j))**(-1)
            return U
```

Gráficos

```
In [9]: def graficar():

            x = np.linspace(1,Nc,Nc)
            plt.plot(x,T2[:,Ny],c='k', ls='-', marker='s',label="T2[i]")
            plt.title("Distribución de Temperatura del aceite a la salida")
            plt.ylabel("T2 [K]")
            plt.xlabel("Columnas")
            plt.legend(loc="best")
            plt.show()

            x = np.linspace(1,Nc,Nc)
            plt.plot(x,T1[0:Nc],c='k', ls='-', marker='s', label="T1[i]")
            plt.title("Distribución de Temperatura del gas a lo largo del eje x")
            plt.ylabel("T1 [K]")
            plt.xlabel("Columnas")
            plt.legend(loc="best")
            plt.show()


            fig = plt.figure(figsize=(11,7), dpi=100)
            ax = fig.gca(projection='3d')
            y = np.linspace(1,Nc,Nc)
            x = np.linspace(0,Ly,Ny+1)
            X,Y = np.meshgrid(x,y)
            surf = ax.plot_surface(Y,X,T2, cmap='coolwarm');
            fig.colorbar(surf, shrink=0.5, aspect=5, label="T2 [K]")
            plt.title("Distribución de la Temperatura del Aceite a lo Largo del Intercambiador");
            plt.ylabel("Distancia a lo Largo del Tubo")
            plt.xlabel("Columnas")
            #ax.set_zlabel('T2')
            fake2Dline = mpl.lines.Line2D([0],[0], linestyle="none", c='k', marker = 's')
            ax.legend([fake2Dline], ['Temperatura del Aceite'], numpoints = 1);
```

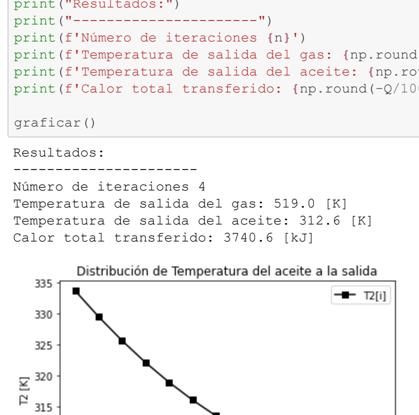
Temperaturas

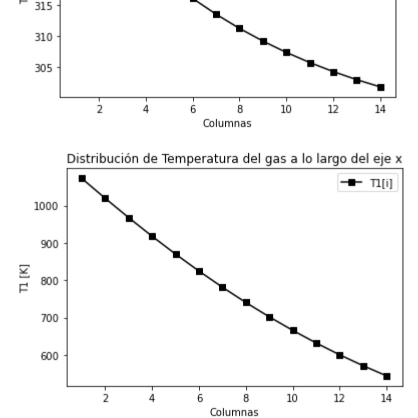
```
In [10]: def pedir_parametros():
             print("Ingresar parámetros para intercambiador de calor: ")
             Nf = input("Número de filas de tubos: ")
             Nc = input("Número de columnas de tubos: ")
             Ly = input("Largo de tubos [m]: ")
             Z = input("Separación vertical de tubos [m]: ")
             Ny = input("Elementos discretización longitudinal: ")
             m1 = input("Flujo másico de gas [kg/s]: ")
             m2 = input("Flujo másico de aceite [kg/s]: ")
             print("Distribución de flujo másico")
             print("0: uniforme, 1: triangular, 2: creciente, 3: decreciente")
             dis = input("Selección: ")
             return (Nf,Nc,Ly,Z,Ny,m1,m2,dis)
```

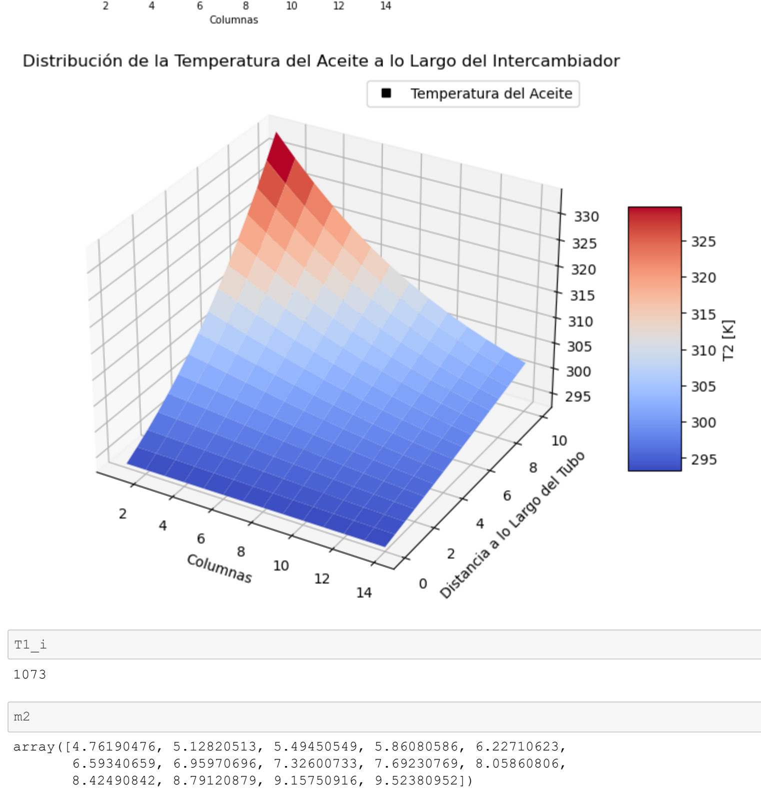
```
In [28]: def pedir_parametros():
             Nf = 12
             Nc = 14
             Ly = 10
             Z = 0.2
             Ny = 15
             m1 = 6
             m2 = 100
             dis = 2
             return (Nf,Nc,Ly,Z,Ny,m1,m2,dis)
```

Loop para resolver

```
In [29]: import numpy as np

         T1_i = 800+273
         T1_o_exit = T1_i - 100
         T1_m = (T1_i + T1_o_exit)/2

         T2_i = 20+273

         Nf,Nc,Ly,Z,Ny,m1,m2,dis = map(float,pedir_parametros())

         Nf,Nc,Ny,dis = int(Nf),int(Nc),int(Ny),int(dis)

         kt,do,di,ro,ri = propiedades_tubos()

         kf,de,t,sigmaf,re,N_aletas = propiedades_aletas()

         dy,dA1,dA2,Rw = calcular_constantes()

         #gas
         T1 = np.zeros(Nc+1)
         T1[0] = T1_i
         T1[Nc] = T1_o_exit

         #aceite
         if dis == 0:
             m2 = np.full(Nc,m2/Nc)
         elif dis ==1:
             m2x1 = np.linspace(1,2,Nc//2)
             m2x2 = np.linspace(2,1,Nc//2)
             if Nc%2 != 0:
                 m2x = np.concatenate((m2x1,np.array([2+2/Nc]),m2x2))
             m2 = m2x*m2/np.sum(m2x)
         elif dis == 2:
             m2x = np.linspace(1,2,Nc)
             m2 = m2x*m2/np.sum(m2x)
         elif dis == 3:
             m2x = np.linspace(2,1,Nc)
             m2 = m2x*m2/np.sum(m2x)
         T2 = np.zeros((Nc,Ny+1))
         T2[:,0] = np.full(Nc,T2_i)


         n = 0
         error = 0.001
         e = error + 1


         while e > error:
             n += 1
             T1_o = T1_o_exit
             T1_m = (T1_i + T1_o)/2
             for i in range(Nc):
                 cp1 = cp_gas(T1[i])
                 rho1 = densidad_gas(T1_m)
                 k1 = conductividad_gas(T1_m)
                 miu1 = viscosidad_gas(T1_m)
                 Pr1 = Pr_gas(T1_m)
                 h1 = calcular_h1()
                 eta_f = calcular_etaf()
                 suma = 0
                 for j in range(Ny):
                     cp2 = cp_oil(T2[i,j])
                     rho2 = densidad_oil(T2[i,j])
                     k2 = conductividad_oil(T2[i,j])
                     miu2 = viscosidad_oil(T2[i,j])
                     U_ij = calcular_Uij(i,j)
                     T2[i,j+1] = T2[i,j] + U_ij*dA2/(cp2*m2[i])*(T1[i]-T2[i,j])
                     suma += U_ij*(T1[i]-T2[i,j])
                 T1[i+1] = T1[i] - dA2/(cp1*m1)*suma
             T1_o_exit = T1[Nc]
             e = np.abs(T1_o - T1_o_exit)

         T1_o = T1_o_exit
         S1 = 0
         S2 = 0
         for i in range(Nc):
             S1 += m2[i]*cp_oil(T2[i,Ny])*T2[i,Ny]
             S2 += m2[i]*cp_oil(T2[i,Ny])
         T2_o = S1/S2

         Q = 0
         for i in range(Nc):
             Q += m1*cp_gas(T1[i])*(T1[i+1]-T1[i])

         print("")
         print("Resultados:")
         print("----------------------")
         print(f'Número de iteraciones {n}')
         print(f'Temperatura de salida del gas: {np.round(T1_o, decimals=1)} [K]')
         print(f'Temperatura de salida del aceite: {np.round(T2_o, decimals=1)} [K]')
         print(f'Calor total transferido: {np.round(-Q/1000, decimals=1)} [kJ]')

         graficar()

Resultados:
----------------------
Número de iteraciones 4
Temperatura de salida del gas: 519.0 [K]
Temperatura de salida del aceite: 312.6 [K]
Calor total transferido: 3740.6 [kJ]
```








```
In [27]: T1_i
Out[27]: 1073

In [20]: m2
Out[20]: array([4.76190476, 5.12820513, 5.49450549, 5.86080586, 6.22710623,
                6.59340659, 6.95970696, 7.32600733, 7.69230769, 8.05860806,
                8.42490842, 8.79120879, 9.15750916, 9.52380952])

In [633]: np.sum(m2)
Out[633]: 35.00000000000001

In [544]: y
Out[544]: array([  1.,   2.,   3.,   4.,   5.,   6.,   7.,   8.,   9.,  10.,  11.,
                  12.,  13.,  14.,  15.,  16.,  17.,  18.,  19.,  20.,  21.,  22.,
                  23.,  24.,  25.,  26.,  27.,  28.,  29.,  30.,  31.,  32.,  33.,
                  34.,  35.,  36.,  37.,  38.,  39.,  40.,  41.,  42.,  43.,  44.,
                  45.,  46.,  47.,  48.,  49.,  50.,  51.,  52.,  53.,  54.,  55.,
                  56.,  57.,  58.,  59.,  60.,  61.,  62.,  63.,  64.,  65.,  66.,
                  67.,  68.,  69.,  70.,  71.,  72.,  73.,  74.,  75.,  76.,  77.,
                  78.,  79.,  80.,  81.,  82.,  83.,  84.,  85.,  86.,  87.,  88.,
                  89.,  90.,  91.,  92.,  93.,  94.,  95.,  96.,  97.,  98.,  99.,
                 100., 101.])

In [ ]:

In [474]: 320-273
Out[474]: 47

In [406]: x = np.linspace(1,3,3)

In [407]: x
Out[407]: array([1., 2., 3.])

In [ ]:
```