Memoria de cálculo para potencial PSP en Acaray

BERNARDO CAPRILE CANALA-ECHEVARRIA

January 2024

Contents

1 Introducción

Para poder diseñar una chimenea de equilibrio es necesario hacer los siguientes estudios previos: Toma de carga de diferentes caudales, rechazo de carga desde diferentes condiciones estables y una combinación de ambas.
Con estos resultados, uno se asegura que el diseño de la chimenea de equilibrio cumpla con las dimensiones adecuadas y así evita presión negativa dentro del tunel de aducción (succión en el tunel o tubería).

1.1 Herramientas de Cálculo

Para poder hacer los gáficos y cálculos se ocupó Python. Python es un lenguaje de programación de alto nivel, interpretado y versátil, utilizado en diversos campos, que destaca por su sintaxis clara y legible, así como por su amplia comunidad de desarrollo, permitiendo la creación de aplicaciones, gráfcos, scripts y soluciones en áreas como programación web, análisis de datos, inteligencia artificial y automatización.

1.2 Librerías ocupadas

Para resolver las ecuaciones diferenciales, se utilizó las siguientes librerías en Python: matplotlib para la visualización de resultados, scipy.integrate para la integración numérica, numpy para operaciones numéricas eficientes, optimize para métodos de optimización y math para funciones matemáticas básicas.

2 Ecuaciones para la oscilación de masa

Es crucial destacar que en los casos de rechazo de carga, el parámetro Q_{turb} no se tiene en consideración. Por el contrario, al considerar la toma de carga, se utilizan ambos parámetros, Q_{turb} y $Q(0)$.

$$Re = \frac{v \cdot D}{\mu} \tag{1}$$

$$\frac{1}{\sqrt{f}} = -2 \cdot \log_{10}\left(\frac{\frac{\varepsilon}{D}}{3.7} + \frac{2.51}{Re \cdot \sqrt{f}}\right) \tag{2}$$

$$h_l = f \cdot \frac{L \cdot v^2}{2 \cdot D \cdot g} \tag{3}$$

$$\frac{dQ(t)}{dt} = \frac{g \cdot A_t}{L}\Big(-z(t) - c_1 \cdot Q(t) \cdot |Q(t)| - \frac{c_2}{A_s^2}(Q(t) - Q(0) - Q_{turb}) \cdot |Q(t) - Q(0) - Q_{turb}|\Big) \tag{4}$$

$$\frac{dz(t)}{dt} = \frac{Q(t) - Q(0) - Q_{turb}}{A_s} \tag{5}$$

Donde:

Re: Número de Reynolds.

v: Velocidad del fluido.

D: Diámetro característico de la tubería.

μ : Viscosidad dinámica del fluido

f: Factor de fricción.

ϵ : Rugosidad absoluta de la tubería.

h_l: Pérdida de carga.

g: Aceleración debida a la gravedad.

Q(t): Caudal en el tiempo

z(t): Altura en el tiempo

A(t): Área transversal del canal.

L: Longitud del canal.

c_1, c_2 : Constantes.

Q(0): Caudal inicial.

Q_{turb} : Caudal turbinado

A_s: Área de la sección transversal de la chimenea de equilibrio.

3 Pérdida por carga

Para calcular la pérdida por carga, se emplearon las siguientes ecuaciones: 1, 2 Y 3. A continuación, se detallan los valores correspondientes a cada parámetro utilizados en el proceso:

$$
\begin{gathered}
\mathrm{Q} = 200 \\
\mathrm{D} = 10 \\
\mathrm{L} = 700 \\
\mathrm{g} = 9.8 \\
\epsilon = 0.0015 \\
\mu = 0.000001
\end{gathered}
$$

```
# Calcular el área de la sección transversal y la velocidad del flujo
A = math.pi * (D/2)**2
V = Q / A

# Calcular el número de Reynolds
Re = (V * D) / 0.000001  # Viscosidad cinemática del agua asumida como 0.000001 m^2/s

# Estimar el factor de fricción utilizando la ecuación de Colebrook
def colebrook(f, Re, D, epsilon):
    return 1 / math.sqrt(f) + 2 * math.log10((epsilon / (3.7 * D)) + (2.51 / (Re * math.sqrt(f))))

# Utilizar el método de la bisección para encontrar el factor de fricción
f_guess = 0.01  # Supongamos un valor inicial para el factor de fricción
f = optimize.bisect(colebrook, 0.0001, 1, args=(Re, D, epsilon))

# Calcular el headloss utilizando la ecuación de Darcy-Weisbach
hf = f * (L / D) * (V**2 / (2 * g))
```

Sustituyendo los valores en las fórmulas, se concluye que la pérdida por carga asciende a 0.3 metros.

4 Parámetros iniciales para las ecuaciones diferenciales

Con el objetivo de maximizar la versatilidad del programa, se han situado los parámetros al inicio. De esta manera, cualquier ajuste necesario en las dimensiones puede llevarse a cabo de manera sencilla y eficiente.

```
# Parámetros
Dh = 0.3 #pérdida por carga (valor calculado anteriormente)
g = 9.8 #aceleración de gravedad
As = 314.16  # sección transversal de la chimenea de equilibrio (surge area) (20m de diámetro)
At = 78.54  # Sección transversal del tunel de aducción (10m de diámetro)
Qtur = 0  # Caudal de la chimenea de equilibrio
L = 700  # Largo del tunel de aducción
c1 = Dh / (200**2) #Constante 1
k = 219.5 #0.5*(As/Ac)^2+(As/Ac-1)^2 #Coeficiente de pérdida
c2 = k / (2 * g) #Constante 2
Q_initials = [50, 100, 150, 200] #caudales de prueba
tiempo_onda= 600 #tiempo total
# Parámetros en forma de diccionario
parameters = {'As': As, 'At': At, 'Qtur': Qtur, 'L': L, 'c1': c1, 'c2': c2}
```

5 Rechazo de carga

En la primera situación, se aborda el rechazo de carga. El objetivo de este análisis es evaluar la altura que alcanzará la columna de agua en la chimenea de equilibrio al momento de cerrar las compuertas. En este escenario, se establece que las compuertas se cierran de manera instantánea.

5.1 Planteamiento de las ecuaciones diferenciales

```
def MassOscillation(y, t, As, At, Qtur, L, c1, c2):
Q, z = y
dQ = g * At / L * (-z - c1 * Q * np.abs(Q) - c2 / As**2 * (Q - Qtur) * np.abs(Q - Qtur))
dz = (Q - Qtur) / As
return [dQ, dz]
```

5.2 Resolución de la ecuación diferencial y ploteo de la función

```
# Tiempos
times = np.linspace(0, tiempo_onda, 1000)
plt.figure(figsize=(10, 6))
for Q_initial in Q_initials:
    yinit = [Q_initial, -c1 * Q_initial**2]
    out = odeint(MassOscillation, yinit, times, args=(As, At, Qtur, L, c1, c2))
    z_values = out[:, 1]
    plt.plot(times, z_values, label=f'Q = ({int(Q_initial)})')
plt.title('Rechazo de carga para diferentes caudales - Altura vs tiempo')
plt.xlabel('Tiempo (s)')
plt.ylabel('z (m)')
plt.legend()
plt.show()
```

5.3 Función de rechazo de carga

A continuación, se presenta el gráfico resultante. Aunque puede ser difícil de apreciar, es importante destacar que la función comienza en z = -0.3, lo cual representa la pérdida inicial por carga.

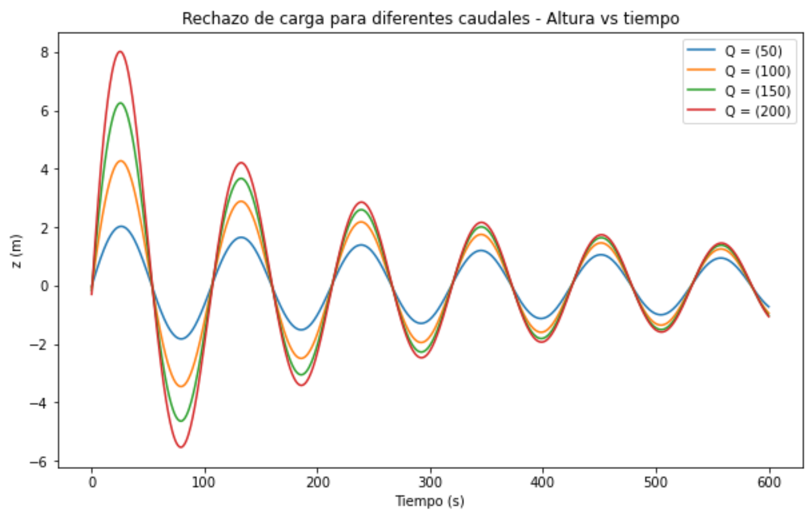

Figure 1: Variación de la altura en el tiempo

Los valores máximos y mínimos del gráfico anterior son los siguientes:

Caudal de 50$\frac{m^3}{s}$: Valor máximo de 2.026m y valor mínimo de -1.827 m
Caudal de 100$\frac{m^3}{s}$: Valor máximo de 4.269m y valor mínimo de -3.458 m
Caudal de 150$\frac{m^3}{s}$: Valor máximo de 6.248m y valor mínimo de -4.641 m
Caudal de 200$\frac{m^3}{s}$: Valor máximo de 8.013m y valor mínimo de -5.535 m

6 Toma de carga

En la segunda situación, se aborda la toma de carga, la cual tiene lugar cuando la central comienza su operación. En este escenario, observamos que el nivel de la chimenea de equilibrio disminuye inicialmente para luego experimentar un ascenso acelerado.

6.1 Planteamiento de ecuaciones diferenciales

```
def GradualStart(y, t, As, At, Qtur, L, c1, c2, Qturb):
Q, z = y
dQ = g * At / L * (-z - c1 * Q * np.abs(Q) - c2 / As**2 * (Q - Qturb) * np.abs(Q - Qturb))
dz = (Q - Qtur - Qturb) / As
return [dQ, dz]
```

6.2 Resolución de la ecuación diferencial y ploteo de la función

Es crucial notar que las condiciones iniciales varían, dado que la altura inicial es -0.3, indicativa de la pérdida de carga, y el caudal inicial es nulo en todos los casos. Además, en este contexto, se asume que las compuertas se abren de manera instantánea y con el caudal turbinado deseado.

```
# Condiciones iniciales y tiempos
y0 = [Dh, 0]  # Altura y caudal inicial igual a 0
times = np.linspace(0, tiempo_onda, 1000)
plt.figure(figsize=(10, 6))
z_crit=[]
for caudal in Q_initials:
    sol = odeint(GradualStart, y0, times, args=(As, At, Qtur, L, c1, c2, caudal))
    plt.plot(times, sol[:, 1], label=f'Q={caudal}')
    z_values = sol[:, 1]
    min_z_index = np.argmin(z_values)
    time_of_min_z = times[min_z_index]
    min_z_value = z_values[min_z_index]
    z_crit.append(min_z_value)
    tcrit=time_of_min_z
plt.title('Toma de carga para diferentes caudales - Altura vs Tiempo')
plt.xlabel('Tiempo (s)')
plt.ylabel('Altura (m)')
plt.legend()
plt.show()
```

6.3 Función de toma de carga

A continuación, se presenta el gráfico resultante. Aunque pueda resultar difícil de percibir, es importante destacar que la función inicia en z = -0.3, reflejando la pérdida de carga inicial. Posteriormente, la función experimenta un descenso inicial debido a la apertura de las compuertas.

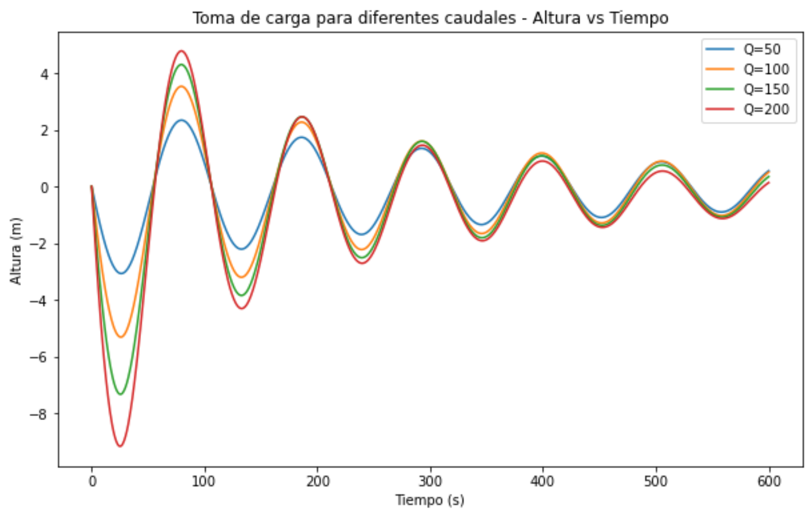

Figure 2: Variación de la altura en el tiempo

Los valores máximos y mínimos del gráfico anterior son los siguientes:

Caudal de $50\frac{m^3}{s}$: Valor máximo de 2.131m y valor mínimo de -2.519 m
Caudal de $100\frac{m^3}{s}$: Valor máximo de 3.486m y valor mínimo de -4.752 m
Caudal de $150\frac{m^3}{s}$: Valor máximo de 4.367m y valor mínimo de -6.762 m
Caudal de $200\frac{m^3}{s}$: Valor máximo de 4.929m y valor mínimo de -8.591 m

7 Toma y rechazo de carga

En la última situación, se aborda la combinación de toma y rechazo de carga. Este escenario se materializa cuando la central comienza su operación y, en el momento en que el agua en la chimenea de equilibrio alcanza su punto más bajo, se procede al cierre inmediato de las compuertas, generando un aumento abrupto en el nivel del agua. El propósito de este estudio es analizar los extremos tanto en la disminución como en el aumento del nivel del agua en esta situación específica.

7.1 Planteamiento de ecuaciones diferenciales

```
    # Función de la ecuación diferencial para el arranque gradual con flujo turbinado
def GradualStart(y, t, As, At, Qtur, L, c1, c2, Qturb):
    Q, z = y
    # Ecuaciones diferenciales
    dQ = g * At / L * (-z - c1 * Q * np.abs(Q) - c2 / As**2 * (Q - Qturb) * np.abs(Q - Qturb))
    dz = (Q - Qtur - Qturb) / As
    return [dQ, dz]

# Función de la ecuación diferencial
def MassOscillation(y, t, As, At, Qtur, L, c1, c2):
    Q, z = y
    dQ = g * At / L * (-z - c1 * Q * np.abs(Q) - c2 / As**2 * (Q - Qtur) * np.abs(Q - Qtur))
    dz = (Q - Qtur) / As
    return [dQ, dz]
```

7.2 Resolución de las ecuaciones diferenciales y ploteo de la función

```
colors = ['r', 'g', 'b', 'c']
# Condiciones iniciales y tiempos
y0 = [Dh, 0]  # Altura y caudal inicial igual a 0
times1 = np.linspace(0, tcrit, 1000) # de 0 al timepo crítico
plt.figure(figsize=(10, 6))
for i, caudal in enumerate(Q_initials):
    sol = odeint(GradualStart, y0, times1, args=(As, At, Qtur, L, c1, c2, caudal))
    plt.plot(times1, sol[:, 1], color=colors[i])
    z_values = sol[:, 1]
z_initials = z_crit #El rechazo de carga parte desde el punto mínimo
times2 = np.linspace(tcrit, tiempo_onda, 1000)# tiempo desde el crítico
for i, caudal in enumerate(Q_initials):
    yinit = [caudal, z_initials[i]]
    out = odeint(MassOscillation, yinit, times2, args=(As, At, Qtur, L, c1, c2))
    z_values = out[:, 1]
    plt.plot(times2, z_values, label=f'Q= ({caudal})', color=colors[i])
plt.title('Toma y rechazo de carga para diferentes caudales - Altura vs Tiempo')
plt.xlabel('Tiempo (s)')
plt.ylabel('z (m)')
plt.legend()
plt.show()
```

7.3 Función de toma y rechazo de carga

A continuación, se presenta el gráfico resultante. Aunque puede resultar difícil de percibir, la función comienza en z=-0,3 debido a la pérdida de carga. Inicialmente, la función experimenta un descenso debido a la apertura de las compuertas, seguido de un ascenso brusco ocasionado por el cierre inmediato de las mismas.

Los valores máximos y mínimos del gráfico anterior son los siguientes:

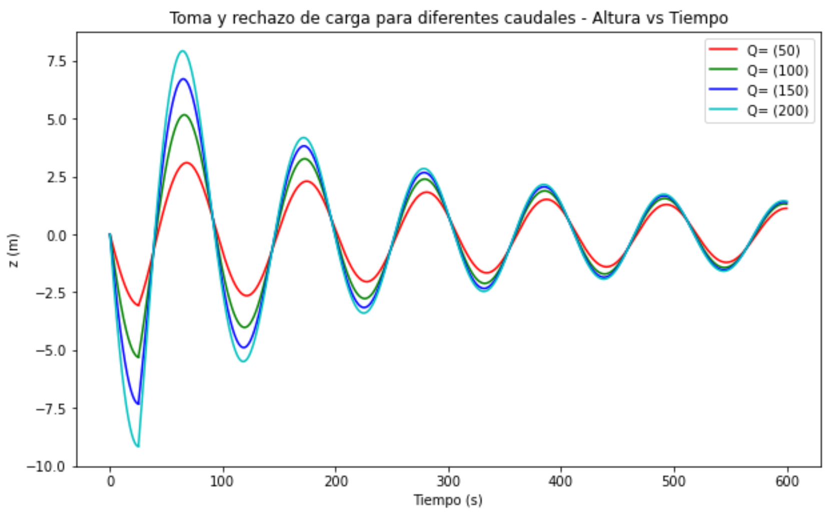

Figure 3: Variación de la altura en el tiempo

Caudal de 50$\frac{m^3}{s}$: Valor máximo de 3.093m y valor mínimo de -2.519 m

Caudal de 100$\frac{m^3}{s}$: Valor máximo de 5.234m y valor mínimo de -4.752 m

Caudal de 150$\frac{m^3}{s}$: Valor máximo de 6.821m y valor mínimo de -6.762 m

Caudal de 200$\frac{m^3}{s}$: Valor máximo de 8.055m y valor mínimo de -8.591 m