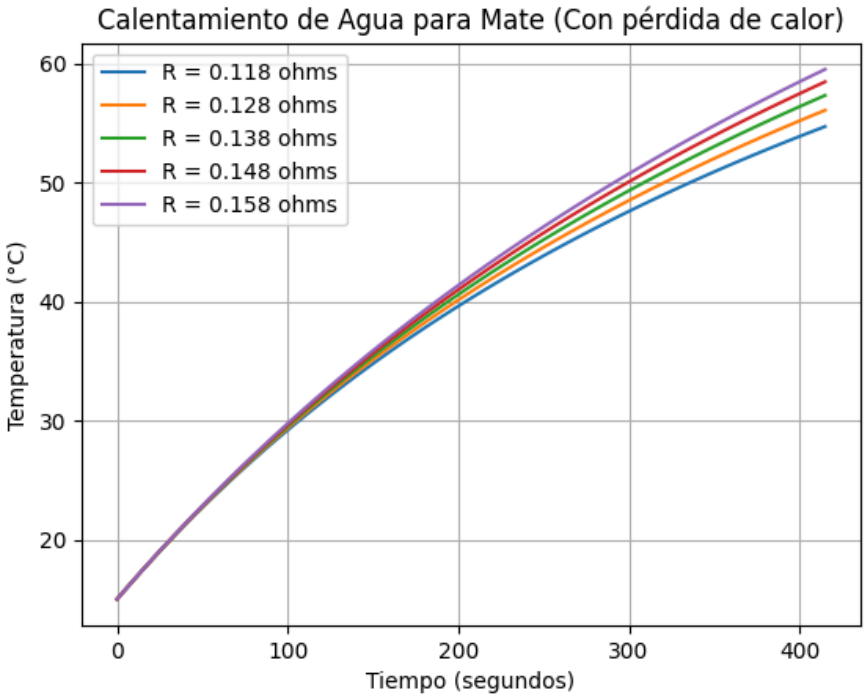
TP5: Realizar familias de curvas con:

a. Distribución uniforme de 5 valores de resistencia (5 valores distintos cercanos al elegido)

   En el siguiente gráfico podemos observar como cambia el calentamiento del agua al ponerle diferentes valores de resistencia y cómo esto puede mejorar o empeorar el tiempo de calentamiento

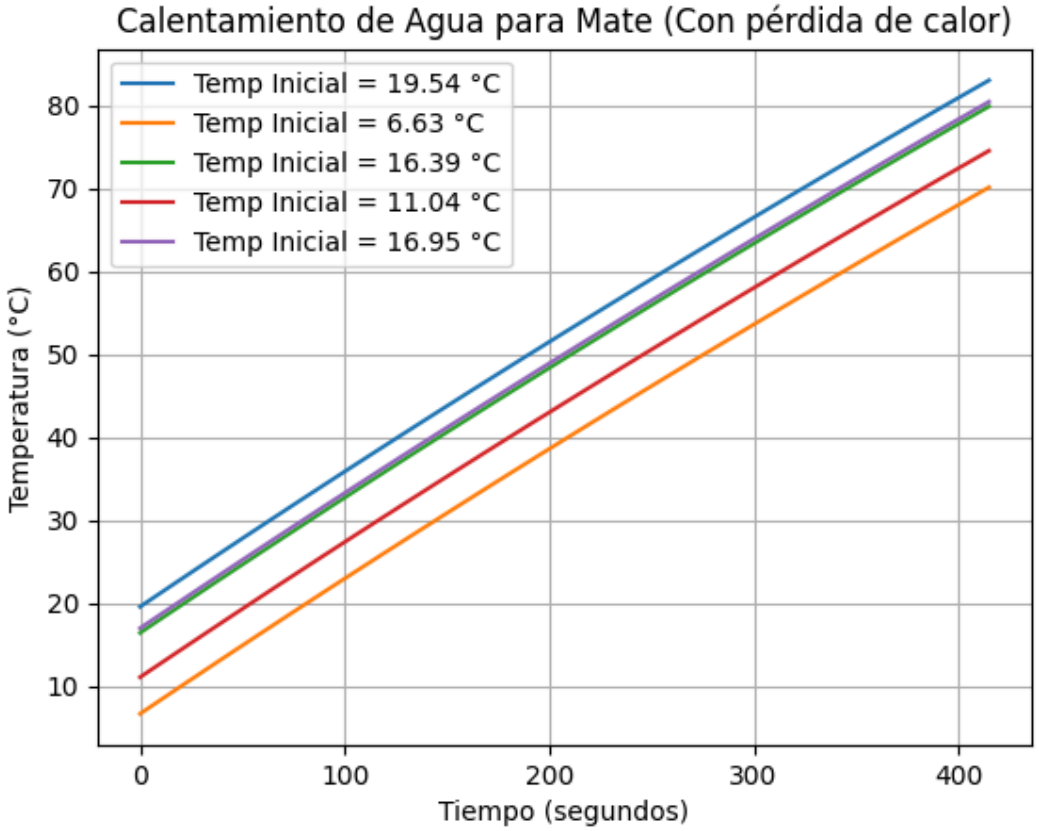



```
import numpy as np

import matplotlib.pyplot as plt



# Datos

t_discreto = np.arange(0, 420, 5)    # Tiempo en segundos (discreto,
intervalo de 5 segundos)

tasa_aumento = 0.1664  # Tasa de aumento de temperatura (°C por segundo)

temperatura_inicial = 15  # Temperatura inicial en °C

resistencia_inicial = 0.118  # Resistencia inicial en ohmios



# Valores de resistencia (con una diferencia de 0.010 ohms)
```

```
valores_resistencia = np.linspace(resistencia_inicial, resistencia_inicial
+ 0.010 * 4, 5)



# Crear una familia de curvas para diferentes resistencias

for resistencia in valores_resistencia:

    temperatura_con_perdida = []

    temperatura_anterior = temperatura_inicial

    for t in t_discreto:

        delta_temp = 1.3288 * t * (temperatura_anterior - 15) / (1.5 * 4180
* resistencia)

        temperatura_nueva = 15 + tasa_aumento * t - delta_temp

        temperatura_con_perdida.append(temperatura_nueva)

        temperatura_anterior = temperatura_nueva

            plt.plot(t_discreto,   temperatura_con_perdida,   label=f"R   =
{resistencia:.3f} ohms")



plt.xlabel("Tiempo (segundos)")

plt.ylabel("Temperatura (°C)")

plt.title("Calentamiento de Agua para Mate (Con pérdida de calor)")

plt.grid(True)

plt.legend()

plt.show()
```

b. Distribución normal de las temperatura iniciales del agua (con media 10 y desvío estándar de 5)

   En el siguiente gráfico veremos el comportamiento del calentador de agua en distintas temperaturas iniciales, teniendo en cuenta una distribución normal de la temperatura ambiente donde tenemos una media de 10 con una desviación estándar de 5.

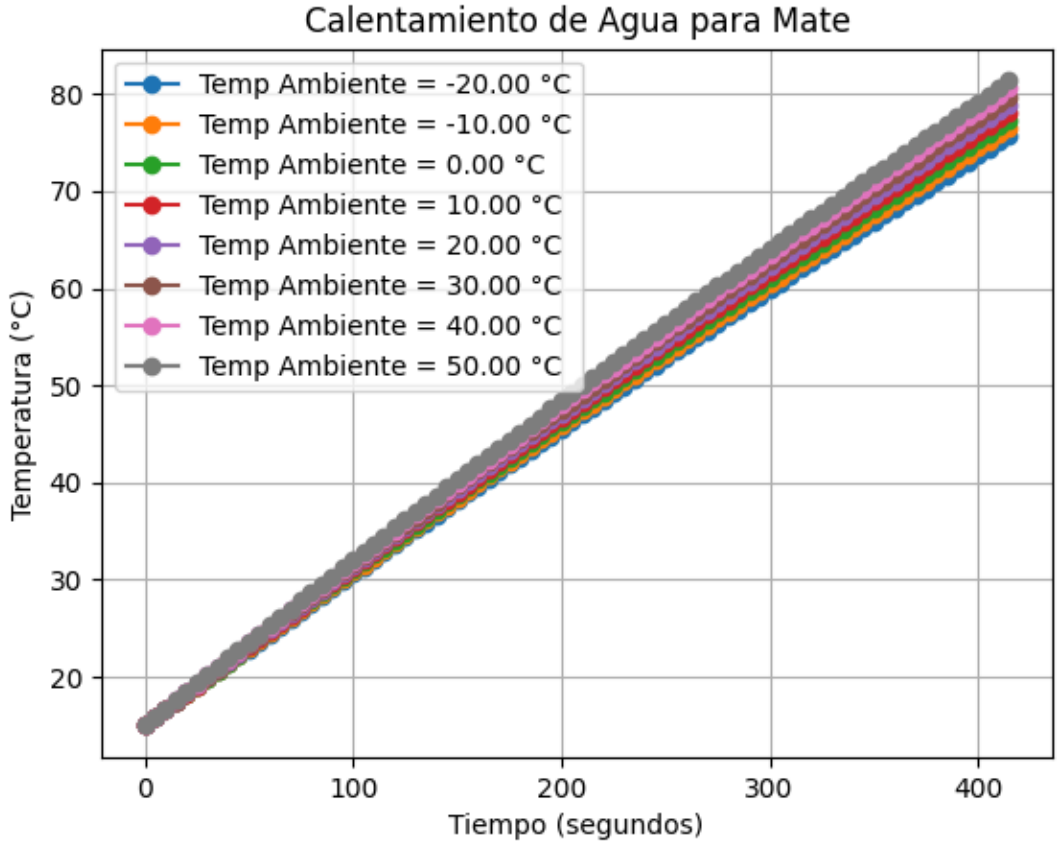



```python
import numpy as np

import matplotlib.pyplot as plt



# Datos

t_discreto = np.arange(0, 420, 5)    # Tiempo en segundos (discreto, intervalo de 5 segundos)

tasa_aumento = 0.1664  # Tasa de aumento de temperatura (°C por segundo)



# Generamos 5 valores de temperatura inicial siguiendo una distribución normal

media_temperatura_inicial = 10

desviacion_estandar_temperatura_inicial = 5

num_valores = 5
```

```
temperaturas_iniciales = np.random.normal(loc=media_temperatura_inicial, scale=desviacion_estandar_temperatura_inicial, size=num_valores)



# Calculamos la temperatura en función del tiempo para cada valor de temperatura inicial

for temperatura_inicial in temperaturas_iniciales:

    temperatura_con_perdida = []

    temperatura_anterior = temperatura_inicial

    for t in t_discreto:

                delta_temp = 1.3288 * t * (temperatura_anterior - temperatura_inicial) / (1.5 * 4180)

            temperatura_nueva = temperatura_inicial + tasa_aumento * t - delta_temp

        temperatura_con_perdida.append(temperatura_nueva)

        temperatura_anterior = temperatura_nueva

     plt.plot(t_discreto, temperatura_con_perdida, label=f"Temp Inicial = {temperatura_inicial:.2f} °C")



plt.xlabel("Tiempo (segundos)")

plt.ylabel("Temperatura (°C)")

plt.title("Calentamiento de Agua para Mate (Con pérdida de calor)")

plt.grid(True)

plt.legend()

plt.show()
```

c. Distribución uniforme de 8 temperaturas iniciales del ambiente (entre -20 y 50 grados)

En el gráfico podemos observar como la temperatura ambiente nos va afectar el calentamiento del agua, el gráfico nos muestra una serie de 8 temperaturas ambientales menores y mayores a la óptima decidida con anterioridad.

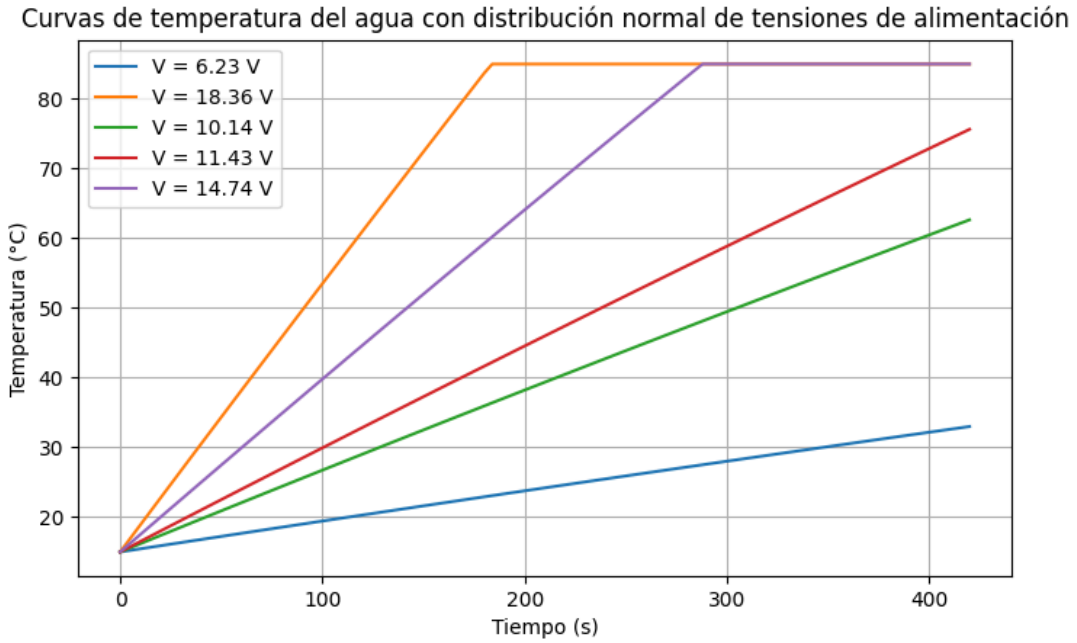



```
import numpy as np

import matplotlib.pyplot as plt



# Datos

t_discreto = np.arange(0, 420, 5)    # Tiempo en segundos (discreto,
intervalo de 5 segundos)

tasa_aumento = 0.1664  # Tasa de aumento de temperatura (°C por segundo)

temperatura_inicial_agua = 15  # Temperatura inicial del agua en °C

temperaturas_ambiente = np.linspace(-20, 50, 8)  # Valores de temperatura
ambiente



# Crear una familia de curvas para diferentes temperaturas ambiente
```

```
for temperatura_ambiente in temperaturas_ambiente:

    temperatura_con_perdida = []

    temperatura_anterior = temperatura_inicial_agua

    for t in t_discreto:

                delta_temp  =  1.3288  *  t  *  (temperatura_anterior  -
temperatura_ambiente) / (1.5 * 4180)

         temperatura_nueva = temperatura_inicial_agua + tasa_aumento * t -
delta_temp

        temperatura_con_perdida.append(temperatura_nueva)

        temperatura_anterior = temperatura_nueva

            plt.plot(t_discreto,   temperatura_con_perdida,   marker="o",
linestyle="-", label=f"Temp Ambiente = {temperatura_ambiente:.2f} °C")



plt.xlabel("Tiempo (segundos)")

plt.ylabel("Temperatura (°C)")

plt.title("Calentamiento de Agua para Mate")

plt.grid(True)

plt.legend()

plt.show()
```

d. Distribución normal de valores de tensión (media de 12V y desvío estándar de 4)

   Este gráfico nos muestra como va a variar el tiempo de calentamiento ante posibles inestabilidades de la tensión, para poder hacer esto calculamos nuevamente la potencia que tendría para cada tensión en particular y con eso el descenso de temperatura que va a tener para cada nueva variación de temperatura.

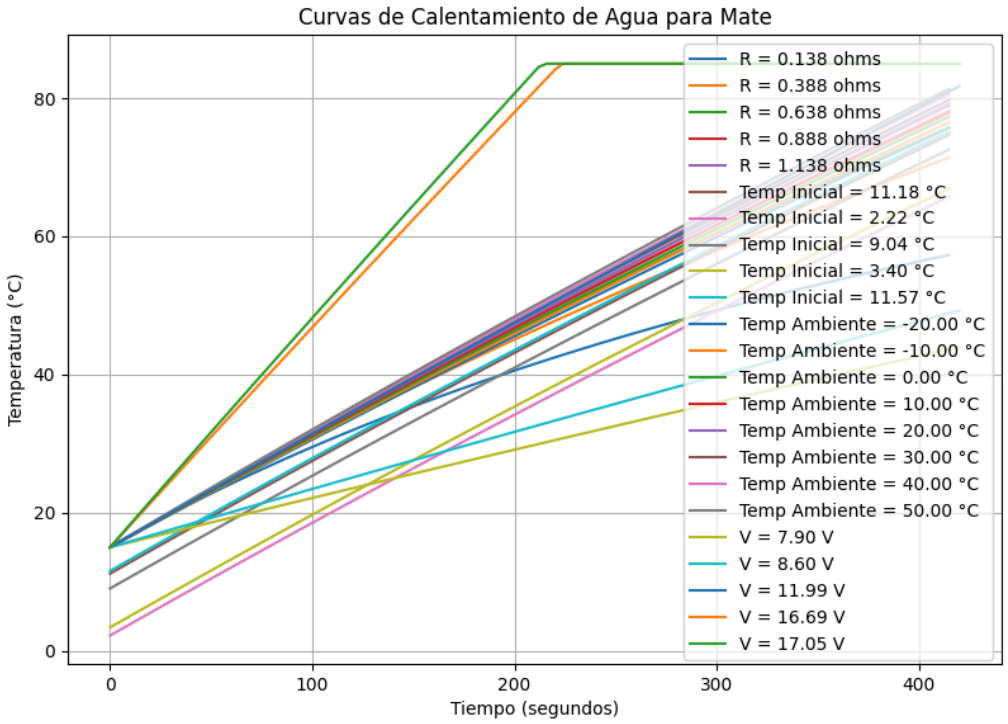



```
import numpy as np

import matplotlib.pyplot as plt

from scipy.integrate import odeint



# Datos

t_discreto = np.arange(0, 420, 5)   # Tiempo en segundos (discreto, intervalo de 5 segundos)

tasa_aumento = 0.1664   # Tasa de aumento de temperatura (°C por segundo)

Ta = 15  # Temperatura inicial en °C

k = 1.3288   # K/ °C

R = 0.138    # ohm

c = 4.186  # capacidad calorifica del agua

m = 1500    # en g

tiempo = 420
```

```
# Generamos 5 valores de temperatura inicial siguiendo una distribución normal

media_tension= 12

desviacion_estandar_tension= 4

num_valores = 5

tensiones = np.random.normal(loc=media_tension, scale=desviacion_estandar_tension, size=num_valores)




# Definir la ecuación diferencial para el cambio de temperatura del agua

def dTdt(T, t, V):

    P = V**2 / R  # Potencia en función de la tensión y la resistencia

    dTdt = P/(m*c)- (k*(T - Ta) ) / (m*c)

    if T >= 85:

        dTdt = 0  # Detener el calentamiento si se alcanza la temperatura máxima

    return dTdt



# Resolver la ecuación diferencial numéricamente y graficar para cada tensión de alimentación

for V in tensiones:

    T_values = odeint(dTdt, Ta, np.arange(0, tiempo + 4, 4), args=(V,))

    plt.plot(np.arange(0, tiempo + 4, 4), T_values, label=f"V = {V:.2f} V")
```

```
# Personalización de la gráfica

plt.title("Curvas de temperatura del agua con distribución normal de
tensiones de alimentación")

plt.xlabel("Tiempo (s)")

plt.ylabel("Temperatura (°C)")

plt.legend()

plt.grid(True)

plt.show()
```

e. Simulaciones que contengan todas las familias de curvas previas




```
import numpy as np

import matplotlib.pyplot as plt

from scipy.integrate import odeint
```

```
# Datos comunes

t_discreto = np.arange(0, 420, 5)  # Tiempo en segundos (discreto, intervalo de 5 segundos)

tasa_aumento = 0.1664  # Tasa de aumento de temperatura (°C por segundo)


# Código 1: Variación de resistencia

temperatura_inicial = 15  # Temperatura inicial en °C

resistencia_inicial = 0.138  # Resistencia inicial en ohmios

valores_resistencia = np.linspace(resistencia_inicial, resistencia_inicial + 0.25 * 4, 5)


for resistencia in valores_resistencia:

    temperatura_con_perdida = []

    temperatura_anterior = temperatura_inicial

    for t in t_discreto:

        delta_temp = 1.3288 * t * (temperatura_anterior - 15) / (1.5 * 4180 * resistencia)

        temperatura_nueva = 15 + tasa_aumento * t - delta_temp

        temperatura_con_perdida.append(temperatura_nueva)

        temperatura_anterior = temperatura_nueva

    plt.plot(t_discreto, temperatura_con_perdida, label=f"R = {resistencia:.3f} ohms")


# Código 2: Variación de temperatura inicial

media_temperatura_inicial = 10

desviacion_estandar_temperatura_inicial = 5

num_valores = 5

temperaturas_iniciales = np.random.normal(loc=media_temperatura_inicial, scale=desviacion_estandar_temperatura_inicial, size=num_valores)
```

```
for temperatura_inicial in temperaturas_iniciales:
    temperatura_con_perdida = []
    temperatura_anterior = temperatura_inicial
    for t in t_discreto:
        delta_temp = 1.3288 * t * (temperatura_anterior - temperatura_inicial) / (1.5 * 4180)
        temperatura_nueva = temperatura_inicial + tasa_aumento * t - delta_temp
        temperatura_con_perdida.append(temperatura_nueva)
        temperatura_anterior = temperatura_nueva
                plt.plot(t_discreto, temperatura_con_perdida, label=f"Temp Inicial = {temperatura_inicial:.2f} °C")

# Código 3: Variación de temperatura ambiente
temperatura_inicial_agua = 15 # Temperatura inicial del agua en °C
temperaturas_ambiente = np.linspace(-20, 50, 8)  # Valores de temperatura ambiente

for temperatura_ambiente in temperaturas_ambiente:
    temperatura_con_perdida = []
    temperatura_anterior = temperatura_inicial_agua
    for t in t_discreto:
          delta_temp = 1.3288 * t * (temperatura_anterior - temperatura_ambiente) / (1.5 * 4180)
        temperatura_nueva = temperatura_inicial_agua + tasa_aumento * t - delta_temp
        temperatura_con_perdida.append(temperatura_nueva)
        temperatura_anterior = temperatura_nueva
      plt.plot(t_discreto, temperatura_con_perdida, linestyle="-", label=f"Temp Ambiente = {temperatura_ambiente:.2f} °C")
```

```
# Código 4: Variación de tensión

Ta = 15  # Temperatura inicial en °C

k = 1.3288  # K/ °C

R = 0.138  # ohm

c = 4.186  # capacidad calorífica del agua

m = 1500  # en g

tiempo = 420

media_tension = 12

desviacion_estandar_tension = 4

tensiones  =  np.random.normal(loc=media_tension,  scale=desviacion_estandar_tension,
size=num_valores)


def dTdt(T, t, V):

    P = V**2 / R  # Potencia en función de la tensión y la resistencia

    dTdt = P / (m * c) - (k * (T - Ta)) / (m * c)

    if T >= 85:

        dTdt = 0  # Detener el calentamiento si se alcanza la temperatura máxima

    return dTdt


for V in tensiones:

    T_values = odeint(dTdt, Ta, np.arange(0, tiempo + 4, 4), args=(V,))

    plt.plot(np.arange(0, tiempo + 4, 4), T_values, label=f"V = {V:.2f} V")


# Personalización de la gráfica

plt.xlabel("Tiempo (segundos)")

plt.ylabel("Temperatura (°C)")

plt.title("Curvas de Calentamiento de Agua para Mate")
```

```
plt.grid(True)

plt.legend()

plt.show()
```