# ps1_2

February 12, 2023

[216]:
```
import math
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint
```

[602]:
```
temp_naut = 320  # degC
temp_ambient = 20  # degC
diameter = 0.02  # meters
thermal_conductivity = 50  # W/mC
heat_transfer_coeff = 100  # W/m^2C
length = 0.05  # meters (L)
distance_step = 0.00625  # meters (deltaX)
distance_range = np.arange(0,length + distance_step,distance_step)
```

$$C_1 = u_o = T_o - T_\infty$$

$$C_2 = -u_o[\frac{\sinh(mL) + (\frac{\beta}{mk})\cosh(mL)}{\cosh(mL) + (\frac{\beta}{mk})\sinh(mL)}]$$

$$m = \sqrt{\frac{\beta P}{kA}}$$

[603]:
```
radius = diameter / 2
perimeter = 2*math.pi*radius
area = math.pi * radius**2
```

[604]:
```
m = math.sqrt((heat_transfer_coeff * perimeter)/(thermal_conductivity * area))
C1 = temp_naut - temp_ambient
C2 = -C1 * (((math.sinh(m*length))+((heat_transfer_coeff/
 ↪(m*thermal_conductivity))*math.cosh(m*length))))/((math.
 ↪cosh(m*length))+((heat_transfer_coeff/(m*thermal_conductivity))*math.
 ↪sinh(m*length)))
```

$$general\ solution u(x) = C_1\cosh(mx) + C_2\sinh(mx)$$

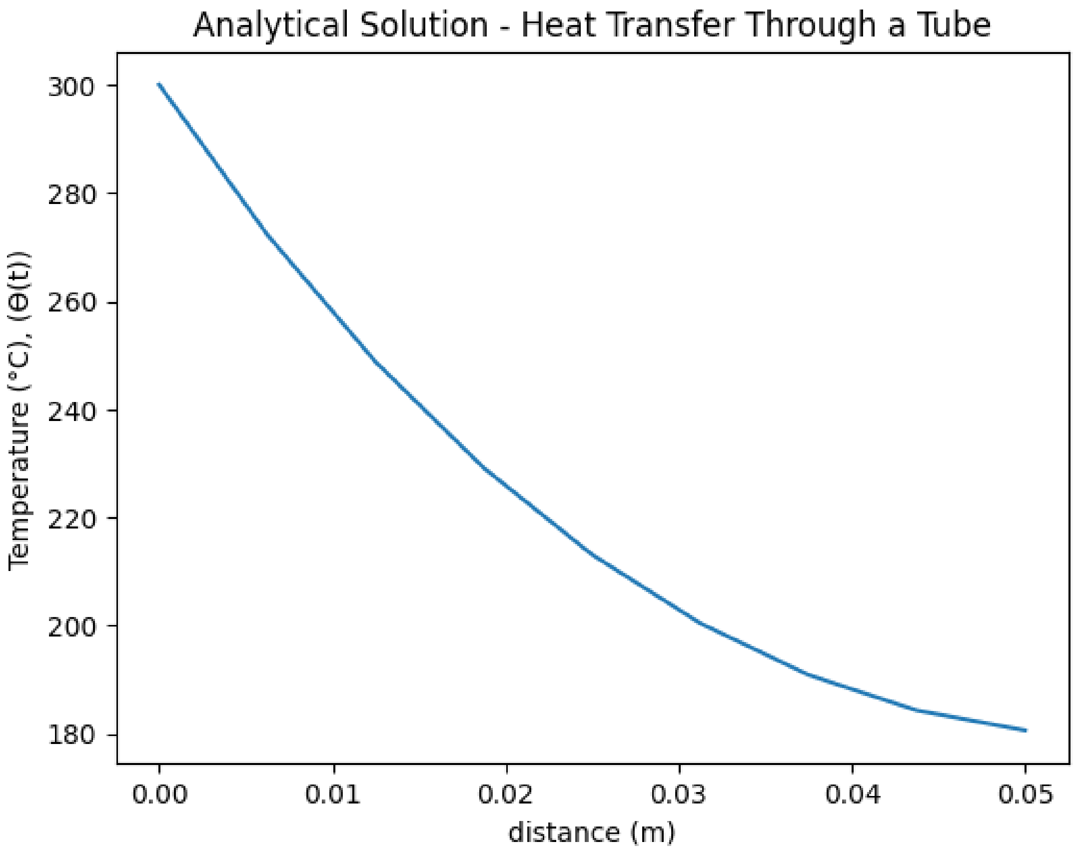
[605]:
```python
def thermal_analytical_solution():
    temp_at_distance = []
    for distance in distance_range:
        temp_at_distance.append((C1*math.cosh(m*distance)) + (C2*math.
 ↪sinh(m*distance)))  # eq 1.2.21
        # temp_at_distance.append(C1 * ((math.cosh(m*(length -␣
 ↪distance))+(heat_transfer_coeff/m*thermal_conductivity)*math.sinh(m*(length␣
 ↪- distance)))/(math.cosh(m*length)+(heat_transfer_coeff/
 ↪m*thermal_conductivity)*math.sinh(m*length)))) # eq 1.2.23
    return temp_at_distance

thermal_analytical_solution()
```

[605]:
```
[300.0,
 272.24532749565884,
 248.75002996442507,
 229.1465151202865,
 213.12807900148871,
 200.44410747797355,
 190.89615530418308,
 184.33484137261553,
 180.65751159327107]
```

[606]:
```python
analytical_solution_output = thermal_analytical_solution()
```

[607]:
```python
plt.xlabel('distance (m)')
plt.ylabel('Temperature (°C), ( (t))')
plt.title('Analytical Solution - Heat Transfer Through a Tube')
plot_time = np.linspace(0, length, len(analytical_solution_output))
plt.plot(plot_time,analytical_solution_output)
```

[607]:
```
[<matplotlib.lines.Line2D at 0x289898a2fd0>]
```




$$-\frac{d^2\theta}{dx^2} + m^2\theta = 0, \quad m = \sqrt{\frac{\beta P}{kA}}, \quad 0 < x < L$$

[608]:

```
# def thermal_numerical_heat_sol_backward_difference():
#     m_series = [m]
#     temp_at_distance = [C1]

#     for i in range(1, len(distance_range)):
#         m_series.append(m_series[i-1] - (distance_step * m**2 *
↪temp_at_distance[i-1]))
#         temp_at_distance.append(temp_at_distance[i-1] + (distance_step *
↪m_series[i]))

#     return temp_at_distance
```

[641]:

```
def thermal_numerical_heat_sol_backward_difference():
    D = 2 + (m * distance_step)**2
    array = np.array([[D,-1,0,0,0,0,0,0,temp_naut],
                      [-1,D,-1,0,0,0,0,0,0],
```

```
                        [0,-1,D,-1,0,0,0,0,0],
                        [0,0,-1,D,-1,0,0,0,0],
                        [0,0,0,-1,D,-1,0,0,0],
                        [0,0,0,0,-1,D,-1,0,0],
                        [0,0,0,0,0,-1,D,-1,0],
                        [0,0,0,0,0,0,-1,1.0875,0]])

    b = array[:, -1]  # Right-hand side of the system of equations
    A = array[:, :-1]  # Coefficient matrix

    # Perform Gaussian elimination
    n = len(b)
    for i in range(n):
        # Find pivot row and swap if necessary
        max_index = np.abs(A[i:, i]).argmax() + i
        if A[max_index, i] == 0:
            raise ValueError("Matrix is singular.")
        if max_index != i:
            A[[i, max_index]] = A[[max_index, i]]
            b[[i, max_index]] = b[[max_index, i]]
        # Reduce rows below pivot
        for j in range(i+1, n):
            ratio = A[j, i] / A[i, i]
            A[j, i:] -= ratio * A[i, i:]
            b[j] -= ratio * b[i]

    # Backsubstitution
    x = np.zeros(n)
    for i in range(n-1, -1, -1):
        x[i] = (b[i] - np.dot(A[i, i+1:], x[i+1:])) / A[i, i]

    x_list = x.tolist()
    x_list.insert(0, 300)
    print(x_list)
    return x_list
```

[642]:
```
thermal_numerical_heat_sol_backward_difference()
```

```
[300, 284.7659823650446, 253.981433204543, 227.16534393786242,
203.89871317021095, 183.81799979584403, 166.60944266828716, 152.00415808242226,
139.7739384665952]
```

[642]:
```
[300,
 284.7659823650446,
 253.981433204543,
 227.16534393786242,
 203.89871317021095,
 183.81799979584403,
```

```
 166.60944266828716,
 152.00415808242226,
 139.7739384665952]
```

[643]:
```python
numerical_solution_output = thermal_numerical_heat_sol_backward_difference()
```

```
[300, 284.7659823650446, 253.981433204543, 227.16534393786242,
203.89871317021095, 183.81799979584403, 166.60944266828716, 152.00415808242226,
139.7739384665952]
```

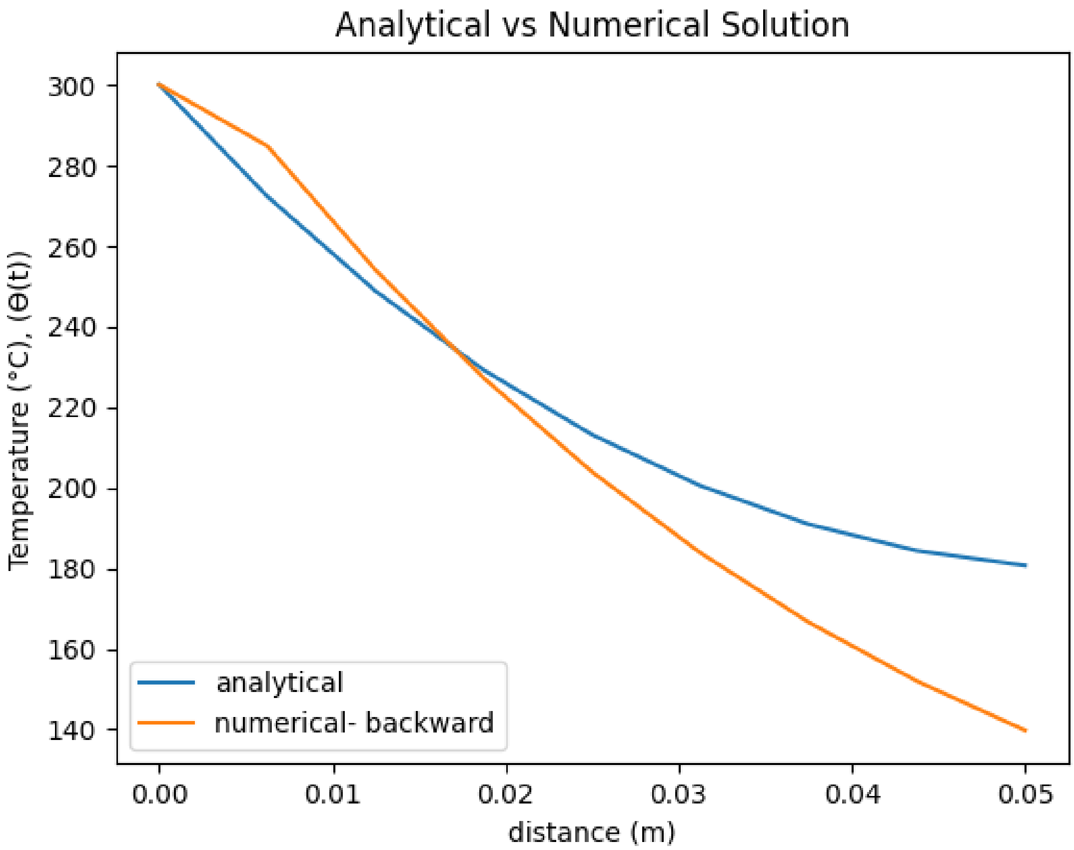
[644]:
```python
central_error_plot = []
def calculate_error_backward(numerical_solution, analytical_solution):
    for i in range(len(distance_range)):
        central_error_plot.append(abs(numerical_solution[i] -␣
 ↪analytical_solution[i]))
        print( f'error at distance {i*distance_step:.2f} =␣
 ↪{abs(numerical_solution[i] - analytical_solution[i]):.5f}')
```

[645]:
```python
calculate_error_backward(numerical_solution_output, analytical_solution_output)
```

```
error at distance 0.00 = 0.00000
error at distance 0.01 = 12.52065
error at distance 0.01 = 5.23140
error at distance 0.02 = 1.98117
error at distance 0.03 = 9.22937
error at distance 0.03 = 16.62611
error at distance 0.04 = 24.28671
error at distance 0.04 = 32.33068
error at distance 0.05 = 40.88357
```

[646]:
```python
plt.xlabel('distance (m)')
plt.ylabel('Temperature (°C), ( (t))')
plt.title('Analytical vs Numerical Solution')
plot_time = np.linspace(0, length, len(analytical_solution_output))
plt.plot(plot_time,analytical_solution_output,"-" , label='analytical',␣
 ↪markersize=3)
plt.plot(plot_time,numerical_solution_output,"-" , label='numerical- backward',␣
 ↪markersize=3)
plt.legend(loc="lower left")
```

[646]:
```
<matplotlib.legend.Legend at 0x2898844cee0>
```


Analytical vs Numerical Solution
Temperature (°C), (Θ(t))
distance (m)
analytical
numerical- backward
300
280
260
240
220
200
180
160
140
0.00
0.01
0.02
0.03
0.04
0.05

[ ]:

[ ]: