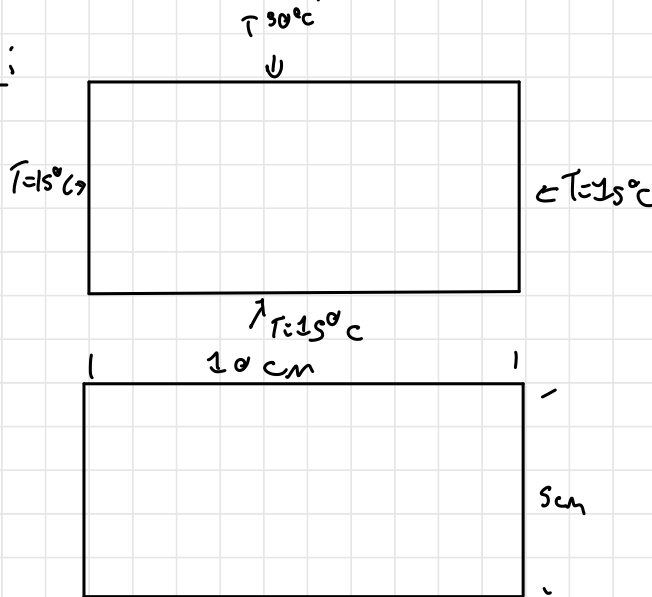
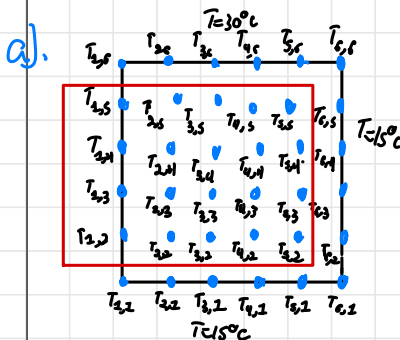


Given:



Read:

- Symmetry right hand side of the domain / adiabatic vert
- Identify all nodes w/ known temp in a list
- Derive all eqns for each of two groups
- Solve for all node  $T$ . Matrix  $A$   $A^{-1}$   $G$  and  $T$
- Create a contour plot
- Compare w/ exact solution (22), (33), (55), and (25)
- Avg and max error. Is this acceptable? What can you do to reduce?



b)

Nodes w/ known  $T$ 's =

$$\{ T_{1,5}, T_{2,5}, T_{3,5}, T_{4,5}, T_{5,5}, T_{1,1}, T_{2,1}, T_{3,1}, T_{4,1}, T_{5,1}, T_{1,2}, T_{1,3}, T_{1,4}, T_{5,2}, T_{5,3}, T_{5,4} \}$$

C. Internal graphs:



Row 2:

$$\begin{aligned} T_{2,2}: T_{2,2} + T_{2,3} + T_{3,2} + T_{2,1} - 4T_{2,2} &= 0 \quad -7.5 \\ T_{3,2}: T_{2,2} + T_{3,3} + T_{4,2} + T_{3,1} - 4T_{3,2} &= 0 \quad -2.5 \\ T_{4,2}: T_{3,2} + T_{4,3} + T_{5,2} + T_{4,1} - 4T_{4,2} &= 0 \quad -2.5 \\ T_{5,2}: T_{4,2} + T_{5,3} + T_{6,2} + T_{5,1} - 4T_{5,2} &= 0 \quad -3.0 \end{aligned}$$

Row 3:

$$\begin{aligned} T_{2,3}: T_{2,3} + T_{2,4} + T_{3,3} + T_{2,2} - 4T_{2,3} &= 0 \\ T_{3,3}: T_{2,3} + T_{3,4} + T_{4,3} + T_{3,2} - 4T_{3,3} &= 0 \\ T_{4,3}: T_{2,3} + T_{4,4} + T_{5,3} + T_{4,2} - 4T_{4,3} &= 0 \\ T_{5,3}: T_{4,3} + T_{5,4} + T_{6,3} + T_{5,2} - 4T_{5,3} &= 0 \quad -2.5 \end{aligned}$$

Row 4:

$$\begin{aligned} T_{2,4}: T_{2,4} + T_{2,5} + T_{3,4} + T_{2,3} - 4T_{2,4} &= 0 \\ T_{3,4}: T_{2,4} + T_{3,5} + T_{4,4} + T_{3,3} - 4T_{3,4} &= 0 \\ T_{4,4}: T_{2,4} + T_{4,5} + T_{5,4} + T_{4,3} - 4T_{4,4} &= 0 \\ T_{5,4}: T_{4,4} + T_{5,5} + T_{6,4} + T_{5,3} - 4T_{5,4} &= 0 \quad -2.5 \end{aligned}$$

Row 5:

$$\begin{aligned} T_{2,5}: T_{2,5} + T_{3,5} + T_{4,5} + T_{2,4} - 4T_{2,5} &= 0 \quad -3.0 \\ T_{3,5}: T_{2,5} + T_{3,6} + T_{4,5} + T_{3,4} - 4T_{3,5} &= 0 \quad -3.0 \\ T_{4,5}: T_{2,5} + T_{4,6} + T_{5,5} + T_{4,4} - 4T_{4,5} &= 0 \quad -3.0 \\ T_{5,5}: T_{4,5} + T_{5,6} + T_{6,5} + T_{5,4} - 4T_{5,5} &= 0 \quad -4.5 \end{aligned}$$

Har Edge:

$$\begin{aligned} T_{2,2}: 0.5 T_{2,3} + T_{2,2} + 0.5 T_{2,1} - 2T_{2,2} &= 0 \quad -7.5 \\ T_{2,3}: 0.5 T_{2,4} + T_{2,3} + 0.5 T_{2,2} - 2T_{2,3} &= 0 \\ T_{2,4}: 0.5 T_{2,5} + T_{2,4} + 0.5 T_{2,3} - 2T_{2,4} &= 0 \\ T_{2,5}: 0.5 T_{2,6} + T_{2,5} + 0.5 T_{2,4} - 2T_{2,5} &= 0 \quad -7.5 \end{aligned}$$

Internal:

$$\begin{bmatrix} T_{2,2} & T_{3,2} & T_{4,2} & T_{5,2} \\ T_{2,3} & T_{3,3} & T_{4,3} & T_{5,3} \\ T_{2,4} & T_{3,4} & T_{4,4} & T_{5,4} \\ T_{2,5} & T_{3,5} & T_{4,5} & T_{5,5} \end{bmatrix}$$

Insulator Edge:

$$\begin{bmatrix} T_{2,2} & T_{2,3} & T_{2,4} & T_{2,5} \end{bmatrix}$$

d.)

82 STOPS different  
figure 3.2 resistance  
figure 3.10 Table 3.3 No post  
figure #  
figure (5.4) 5.3 APP C. 3.

# hw4

October 8, 2024

## 1 BENG 37303 HW 4

```
[433]: import numpy as np
import matplotlib.pyplot as plt
print("Transport HW4")

np.set_printoptions(precision=3, suppress=True, linewidth=200)
```

Transport HW4

## 2 Part D

### 2.1 Solve for all node temperatures including $A$ , $A^{-1}$ , $C$ , and $T$ .

Tiene la forma de matriz

$$AT = C$$

donde el  $T$  es el vector de las Temperatures y la  $C$  is el vector de columna de constantes en las equaciones. Luego, puedes invertir el matriz  $A$  a  $A^{-1}$  para que se encuentra las soluciones de la temperatura.

$$\begin{pmatrix} -2 & 0.5 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.5 & -2 & 0.5 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & -2 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -4 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -4 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 & -4 \end{pmatrix} \begin{pmatrix} T_{1,2} \\ T_{1,3} \\ T_{1,4} \\ T_{1,5} \\ T_{2,2} \\ T_{3,2} \\ T_{4,2} \\ T_{5,2} \\ T_{2,3} \\ T_{3,3} \\ T_{4,3} \\ T_{5,3} \\ T_{2,4} \\ T_{3,4} \\ T_{4,4} \\ T_{5,4} \\ T_{2,5} \\ T_{3,5} \\ T_{4,5} \\ T_{5,5} \end{pmatrix} = \begin{pmatrix} -7.5 \\ 0 \\ 0 \\ -7.5 \\ -15 \\ -15 \\ -15 \\ -30 \\ 0 \\ 0 \\ 0 \\ -15 \\ 0 \\ 0 \\ -15 \\ -30 \\ -30 \\ -30 \\ -30 \\ -45 \end{pmatrix}$$

El vector de columna (temperatura) puede ser determinado por

$$T = A^{-1}C$$

```

[434]: A = np.array([
    [-2, 0.5, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0.5, -2, 0.5, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0.5, -2, 0.5, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
    [0, 0, 0.5, -2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
    [1, 0, 0, 0, -4, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 1, -4, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 1, -4, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 1, -4, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
    [0, 1, 0, 0, 1, 0, 0, 0, -4, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 1, 0, 0, 1, -4, 1, 0, 0, 1, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 1, 0, 0, 1, -4, 1, 0, 0, 1, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, -4, 0, 0, 0, 1, 0, 0, 0],
    [0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, -4, 1, 0, 0, 1, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, -4, 1, 0, 0, 1, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, -4, 1, 0, 0, 1],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, -4, 0, 0, 1],
    [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, -4, 1, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, -4, 1],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, -4]
])

print("Original Matrix A: ")
print(A)

# Vector de soluciones
C = np.array([-7.5, 0, 0, -7.5, -15, -15, -15, -30, 0, 0, 0, -15, 0, 0, 0, -15,
    ↪-30, -30, -30, -45])

print("\nOriginal Solution Vector C: ")
print(C)

Ainv = np.linalg.inv(A)

print("\nInverted matrix A: ")
print(Ainv)

T = np.matmul(Ainv, C)

print("\nSolution for Right handed Temperatures Vector T: ")
print(T)

name_list = [
    "T_{1,2}", "T_{1,3}", "T_{1,4}", "T_{1,5}",
    "T_{2,2}", "T_{3,2}", "T_{4,2}", "T_{5,2}",
    "T_{2,3}", "T_{3,3}", "T_{4,3}", "T_{5,3}",

```

```

    "T_{2,4}", "T_{3,4}", "T_{4,4}", "T_{5,4}",
    "T_{2,5}", "T_{3,5}", "T_{4,5}", "T_{5,5}"
]

temperature_dict = dict(zip(name_list, T))
print("\nTemperatures: ")
for name, temperature in zip(name_list, T):
    print(f"{name}: {temperature} °C ")

fig2 = plt.figure(figsize=(12,4))
plt.subplot(121)
plt.imshow(A,interpolation='none')
clb=plt.colorbar()
clb.set_label('Matrix elements values')
plt.title('Matrix A ',fontsize=24)
plt.subplot(122)
plt.imshow(Ainv,interpolation='none')
clb=plt.colorbar()
clb.set_label('Matrix elements values')
plt.title(r'Matrix  $A^{-1}$  ',fontsize=24)

fig2.tight_layout()
plt.show()

```

Orginal Matrix A:

```

[[-2.  0.5  0.  0.  1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
  0.  0.  0.  0. ]
 [ 0.5 -2.  0.5  0.  0.  0.  0.  0.  1.  0.  0.  0.  0.  0.  0.  0.
  0.  0.  0.  0. ]
 [ 0.  0.5 -2.  0.5  0.  0.  0.  0.  0.  0.  0.  0.  1.  0.  0.  0.
  0.  0.  0.  0. ]
 [ 0.  0.  0.5 -2.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
  1.  0.  0.  0. ]
 [ 1.  0.  0.  0. -4.  1.  0.  0.  1.  0.  0.  0.  0.  0.  0.  0.
  0.  0.  0.  0. ]
 [ 0.  0.  0.  0.  1. -4.  1.  0.  0.  1.  0.  0.  0.  0.  0.  0.
  0.  0.  0.  0. ]
 [ 0.  0.  0.  0.  0.  1. -4.  1.  0.  0.  1.  0.  0.  0.  0.  0.
  0.  0.  0.  0. ]
 [ 0.  0.  0.  0.  0.  0.  1. -4.  0.  0.  0.  1.  0.  0.  0.  0.
  0.  0.  0.  0. ]
 [ 0.  1.  0.  0.  1.  0.  0.  0. -4.  1.  0.  0.  1.  0.  0.  0.
  0.  0.  0.  0. ]
 [ 0.  0.  0.  0.  0.  1.  0.  0.  1. -4.  1.  0.  0.  1.  0.  0.

```

```

0.  0.  0.  0. ]
[ 0.  0.  0.  0.  0.  0.  1.  0.  0.  1. -4.  1.  0.  0.  1.  0.
0.  0.  0.  0. ]
[ 0.  0.  0.  0.  0.  0.  0.  1.  0.  0.  1. -4.  0.  0.  0.  1.
0.  0.  0.  0. ]
[ 0.  0.  1.  0.  0.  0.  0.  0.  1.  0.  0.  0. -4.  1.  0.  0.
1.  0.  0.  0. ]
[ 0.  0.  0.  0.  0.  0.  0.  0.  0.  1.  0.  0.  1. -4.  1.  0.
0.  1.  0.  0. ]
[ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  1.  0.  0.  1. -4.  1.
0.  0.  1.  0. ]
[ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  1.  0.  0.  1. -4.
0.  0.  0.  1. ]
[ 0.  0.  0.  1.  0.  0.  0.  0.  0.  0.  0.  0.  1.  0.  0.  0.
-4.  1.  0.  0. ]
[ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  1.  0.  0.
1. -4.  1.  0. ]
[ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  1.  0.
0.  1. -4.  1. ]
[ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  1.
0.  0.  1. -4. ]]

```

Original Solution Vector C:

```

[ -7.5  0.  0. -7.5 -15. -15. -15. -30.  0.  0.  0. -15.  0.
0.  0. -15. -30. -30. -30. -45. ]

```

Inverted matrix A:

```

[[-0.704 -0.315 -0.158 -0.069 -0.251 -0.101 -0.043 -0.016 -0.2 -0.109 -0.055
-0.023 -0.124 -0.079 -0.044 -0.019 -0.058 -0.04 -0.024 -0.011]
[-0.315 -0.862 -0.384 -0.158 -0.2 -0.109 -0.055 -0.023 -0.375 -0.18 -0.087
-0.036 -0.258 -0.149 -0.078 -0.033 -0.124 -0.079 -0.044 -0.019]
[-0.158 -0.384 -0.862 -0.315 -0.124 -0.079 -0.044 -0.019 -0.258 -0.149 -0.078
-0.033 -0.375 -0.18 -0.087 -0.036 -0.2 -0.109 -0.055 -0.023]
[-0.069 -0.158 -0.315 -0.704 -0.058 -0.04 -0.024 -0.011 -0.124 -0.079 -0.044
-0.019 -0.2 -0.109 -0.055 -0.023 -0.251 -0.101 -0.043 -0.016]
[-0.251 -0.2 -0.124 -0.058 -0.403 -0.147 -0.059 -0.021 -0.212 -0.127 -0.066
-0.027 -0.119 -0.084 -0.049 -0.022 -0.054 -0.041 -0.026 -0.012]
[-0.101 -0.109 -0.079 -0.04 -0.147 -0.36 -0.126 -0.042 -0.127 -0.169 -0.1
-0.043 -0.084 -0.089 -0.062 -0.03 -0.041 -0.04 -0.029 -0.015]
[-0.043 -0.055 -0.044 -0.024 -0.059 -0.126 -0.344 -0.104 -0.066 -0.1 -0.146
-0.073 -0.049 -0.062 -0.069 -0.04 -0.026 -0.029 -0.029 -0.017]
[-0.016 -0.023 -0.019 -0.011 -0.021 -0.042 -0.104 -0.302 -0.027 -0.043 -0.073
-0.103 -0.022 -0.03 -0.04 -0.039 -0.012 -0.015 -0.017 -0.014]
[-0.2 -0.375 -0.258 -0.124 -0.212 -0.127 -0.066 -0.027 -0.521 -0.231 -0.108
-0.044 -0.266 -0.168 -0.091 -0.039 -0.119 -0.084 -0.049 -0.022]
[-0.109 -0.18 -0.149 -0.079 -0.127 -0.169 -0.1 -0.043 -0.231 -0.449 -0.187
-0.072 -0.168 -0.209 -0.129 -0.058 -0.084 -0.089 -0.062 -0.03 ]
[-0.055 -0.087 -0.078 -0.044 -0.066 -0.1 -0.146 -0.073 -0.108 -0.187 -0.413

```

```

-0.144 -0.091 -0.129 -0.175 -0.09  -0.049 -0.062 -0.069 -0.04 ]
[-0.023 -0.036 -0.033 -0.019 -0.027 -0.043 -0.073 -0.103 -0.044 -0.072 -0.144
-0.341 -0.039 -0.058 -0.09  -0.118 -0.022 -0.03  -0.04  -0.039]
[-0.124 -0.258 -0.375 -0.2   -0.119 -0.084 -0.049 -0.022 -0.266 -0.168 -0.091
-0.039 -0.521 -0.231 -0.108 -0.044 -0.212 -0.127 -0.066 -0.027]
[-0.079 -0.149 -0.18  -0.109 -0.084 -0.089 -0.062 -0.03  -0.168 -0.209 -0.129
-0.058 -0.231 -0.449 -0.187 -0.072 -0.127 -0.169 -0.1  -0.043]
[-0.044 -0.078 -0.087 -0.055 -0.049 -0.062 -0.069 -0.04  -0.091 -0.129 -0.175
-0.09  -0.108 -0.187 -0.413 -0.144 -0.066 -0.1  -0.146 -0.073]
[-0.019 -0.033 -0.036 -0.023 -0.022 -0.03  -0.04  -0.039 -0.039 -0.058 -0.09
-0.118 -0.044 -0.072 -0.144 -0.341 -0.027 -0.043 -0.073 -0.103]
[-0.058 -0.124 -0.2   -0.251 -0.054 -0.041 -0.026 -0.012 -0.119 -0.084 -0.049
-0.022 -0.212 -0.127 -0.066 -0.027 -0.403 -0.147 -0.059 -0.021]
[-0.04  -0.079 -0.109 -0.101 -0.041 -0.04  -0.029 -0.015 -0.084 -0.089 -0.062
-0.03  -0.127 -0.169 -0.1  -0.043 -0.147 -0.36  -0.126 -0.042]
[-0.024 -0.044 -0.055 -0.043 -0.026 -0.029 -0.029 -0.017 -0.049 -0.062 -0.069
-0.04  -0.066 -0.1  -0.146 -0.073 -0.059 -0.126 -0.344 -0.104]
[-0.011 -0.019 -0.023 -0.016 -0.012 -0.015 -0.017 -0.014 -0.022 -0.03  -0.04
-0.039 -0.027 -0.043 -0.073 -0.103 -0.021 -0.042 -0.104 -0.302]]

```

Solution for Right handed Temperatures Vector T:

```

[16.997 19.    20.782 21.168 16.994 16.867 16.51  15.869 19.11  18.966 18.302
16.967 21.48  21.586 20.767 18.695 24.444 25.129 24.485 22.045]

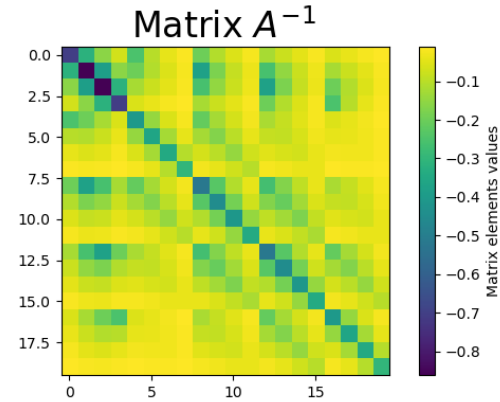
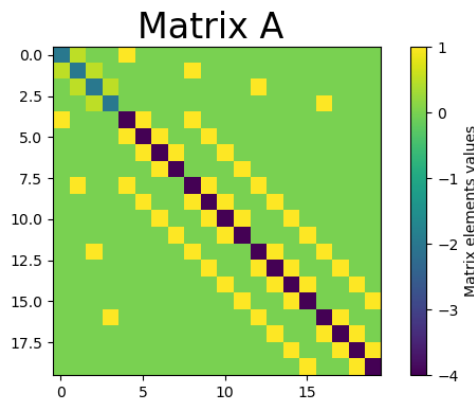
```

Temperatures:

```

T_{1,2}: 16.996687041101918 °C
T_{1,3}: 18.999698103939426 °C
T_{1,4}: 20.78207339798611 °C
T_{1,5}: 21.16762618097767 °C
T_{2,2}: 16.993525030234125 °C
T_{3,2}: 16.867397091499747 °C
T_{4,2}: 16.50970717011274 °C
T_{5,2}: 15.869053570948786 °C
T_{2,3}: 19.110015988334837 °C
T_{3,3}: 18.966356165652122 °C
T_{4,3}: 18.302378018002422 °C
T_{5,3}: 16.96650711368241 °C
T_{2,4}: 21.480484653513678 °C
T_{3,4}: 21.5856335647715 °C
T_{4,4}: 20.766941622562413 °C
T_{5,4}: 18.694596865778436 °C
T_{2,5}: 24.44421566296228 °C
T_{3,5}: 25.128751817357767 °C
T_{4,5}: 24.48515804169728 °C
T_{5,5}: 22.044938726868928 °C

```



ya nos hace falta reparar la matriz en total

### 3 Part E

#### 3.1 Create a contour plot of the Temperature Distribution.

```
[435]: T_right = np.zeros((6, 6))

# Set the bottom row to 15
T_right[5, :] = 15

# Set the right column to 15
T_right[:, 5] = 15

# Set the top row to 30
T_right[0, :] = 30

#print(T_right)

left_values = T[:4]
left_values = np.flip(left_values)

interior_values = T[4:]
row1 = interior_values[:4]
row2 = interior_values[4:8]
row3 = interior_values[8:12]
row4 = interior_values[12:16]

row1 = np.flip(row1)
row2 = np.flip(row2)
row3 = np.flip(row3)
row4 = np.flip(row4)
```



```

interior_values = np.concatenate((row1, row2, row3, row4))

#print("row 2" + str(row1))
interior_values = np.flipud(interior_values)

interior_values = interior_values.reshape((4,4))
#$print(interior_values)

# Fill in the interior values from T_vector
T_right[1:5, 0] = left_values

T_right[1:5, 1:5] = interior_values
#print(interior_values.reshape(4,4))

# Print the result
print("T matrix (right side 6x6): ")
print(T_right)

T_full= np.zeros((6,11))
#print(T_full)

T_full[:, 5:11] = T_right
T_full[:, 0:5] = np.fliplr(T_right[:, 1:6])
print("\n T matrix (full): ")
print(T_full)

joe = plt.figure(figsize=(12, 4))

# First subplot (Matrix A)
plt.subplot(121)
plt.imshow(T_right, interpolation='none', extent=[0, 5, 0, 5]) # Adjusting the
↪ extent
clb = plt.colorbar()
clb.set_label('Temperature (°C)')
plt.title('Plot Right Side Exchanger', fontsize=18)
plt.xlabel('Distance (cm)')
plt.ylabel('Distance (cm)')
plt.xticks(np.arange(0, 6, 1)) # Set ticks for x-axis
plt.yticks(np.arange(0, 6, 1)) # Set ticks for y-axis

# Second subplot (Matrix A-1)
plt.subplot(122)

```

```

plt.imshow(T_full, interpolation='none', extent=[0, 10, 0, 5]) # Adjusting the
↳ extent
clb = plt.colorbar()
clb.set_label('Temperature (°C)')
plt.title('Plot Entire Exchanger', fontsize=18)
plt.xlabel('Distance (cm)')
plt.ylabel('Distance (cm)')
plt.xticks(np.arange(0, 11, 1)) # Set ticks for x-axis
plt.yticks(np.arange(0, 6, 1)) # Set ticks for y-axis

# Adjust layout
joe.tight_layout()
plt.show()

fig = plt.figure(figsize=(12,4))
# Left subplot for the right-hand matrix
plt.subplot(121)
# Create a filled contour plot
contour_filled = plt.contourf(T_right, cmap='viridis', extent=[0,5,0,5])
# Add contour lines
contour_lines = plt.contour(T_right, colors='black', linewidths=0.5)
# Add labels to the contour lines
plt.clabel(contour_lines, inline=True, fontsize=8)
clb = plt.colorbar(contour_filled)
clb.set_label('Temperature (°C)')
plt.title('Contour Plot Right Side Exchanger', fontsize=18)
plt.xlabel('Distance (cm)')
plt.ylabel('Distance (cm)')
plt.gca().invert_yaxis()
plt.xticks(np.arange(0, 6, 1)) # Set ticks for x-axis
plt.yticks(np.arange(0, 6, 1)) # Set ticks for y-axis

# Right subplot for the full matrix
plt.subplot(122)
# Create a filled contour plot
contour_filled = plt.contourf(T_full, cmap='viridis', extent=[0,10,0,5])
# Add contour lines
contour_lines = plt.contour(T_full, colors='black', linewidths=0.5)
# Add labels to the contour lines
plt.clabel(contour_lines, inline=True, fontsize=8)
clb = plt.colorbar(contour_filled)
clb.set_label('Temperature (°C)')
plt.title('Contour Plot Entire Exchanger', fontsize=18)
plt.xlabel('Distance (cm)')
plt.ylabel('Distance (cm)')
plt.gca().invert_yaxis()

```

```
plt.xticks(np.arange(0, 11, 1)) # Set ticks for x-axis
plt.yticks(np.arange(0, 6, 1)) # Set ticks for y-axis

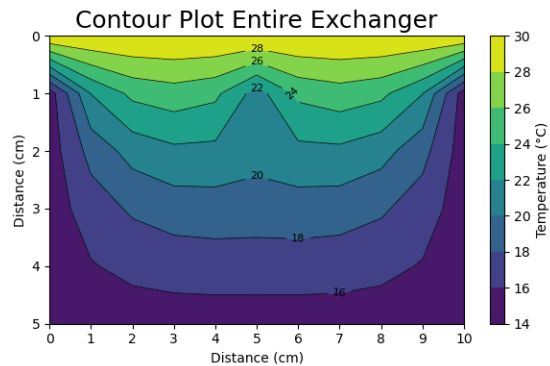
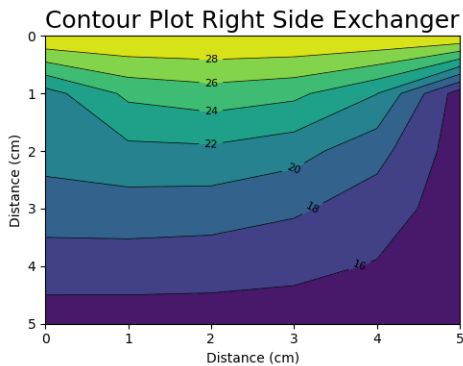
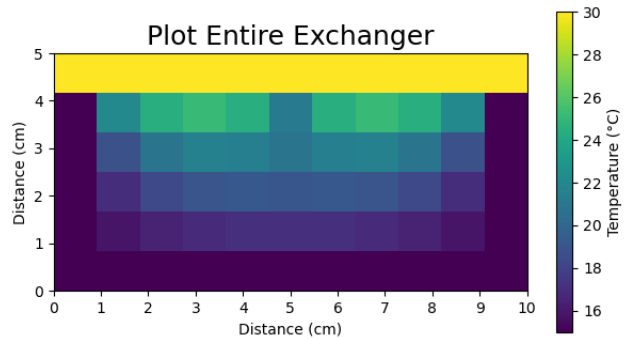
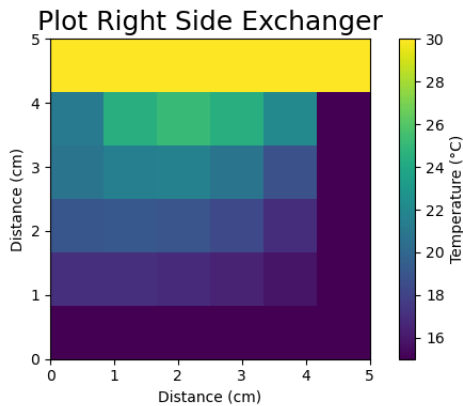
fig.tight_layout()
plt.show()
```

T matrix (right side 6x6):

```
[[30.    30.    30.    30.    30.    30.   ]
 [21.168 24.444 25.129 24.485 22.045 15.   ]
 [20.782 21.48  21.586 20.767 18.695 15.   ]
 [19.    19.11  18.966 18.302 16.967 15.   ]
 [16.997 16.994 16.867 16.51  15.869 15.   ]
 [15.    15.    15.    15.    15.    15.   ]]
```

T matrix (full):

```
[[30.    30.    30.    30.    30.    30.    30.    30.    30.    30.    30.   ]
 [15.    22.045 24.485 25.129 24.444 21.168 24.444 25.129 24.485 22.045 15.   ]
 [15.    18.695 20.767 21.586 21.48  20.782 21.48  21.586 20.767 18.695 15.   ]
 [15.    16.967 18.302 18.966 19.11  19.    19.11  18.966 18.302 16.967 15.   ]
 [15.    15.869 16.51  16.867 16.994 16.997 16.994 16.867 16.51  15.869 15.   ]
 [15.    15.    15.    15.    15.    15.    15.    15.    15.    15.    15.   ]]
```



## 4 Part F: Comparison

Part F: Compare your numerical solution to the exact solution (equation 4.19) for 4 selected calculated location within the dough (nodes 2,2; 3,3; 5,5; and 2,5)

$$\theta_{x,y} = \frac{T - T_1}{T_2 - T_1}$$
$$\theta_{x,y} = \frac{T - 15}{30 - 15}$$

where

$$\theta_{x,y} = \frac{2}{\pi} \sum_{n=1}^{\infty} \frac{(-1)^{n+1} + 1}{n} \sin \frac{n\pi x}{L} \frac{\sinh(n\pi y/L)}{\sinh(n\pi W/L)}$$

where  $W$  is the width and  $L$  is the length. After  $\theta$  is determined, then it can be subbed back into the transformation function to find the true value of  $T$ .

```
[436]: #PART F
t1 = 15
t2 = 30

def theta_to_temperature(theta):
    #T = Theta * t2-t1 + T1
    temperature = ((t2-t1) * theta) + t1
    return temperature

def truetemperature(x, y, L, W, N = 350):
    theta_sum = 0
    for n in range (1, N+1):
        term1= (-1)**(n+1) + 1
        term2= np.sin(n * np.pi * x / L)
        sinh_term = (np.sinh(n * np.pi * y / L )) / (np.sinh(n * np.pi * W / L))

        if np.sinh(n * np.pi * W/L) == 0:
            continue
        theta_sum += (term1/n) * term2 * sinh_term

    theta_val = theta_sum * (2/np.pi)
    temptrue = theta_to_temperature(theta_val)

    return temptrue

#num = truetemperature(5, 5, 10, 5)
#print(num)
```

```

# trying part F
infsum22 = truetemperature(6, 1, 10,5)
finite22 = temperature_dict["T_{2,2}"]
print(f"Infinte Sum Solution for Node (2,2): {infsum22}")
print(f"Finite Difference Solution for Node (2,2): {finite22}")
print()

infsum33 = truetemperature(7, 2, 10, 5)
finite33 = temperature_dict["T_{3,3}"]
print(f"Infinte Sum Solution for Node (3,3): {infsum33}")
print(f"Finite Difference Solution for Node (3,3): {finite33}")
print()

infsum55 = truetemperature(9, 4, 10, 5)
finite55 = temperature_dict["T_{5,5}"]
print(f"Infinte Sum Solution for Node (5,5): {infsum55}")
print(f"Finite Difference Solution for Node (5,5): {finite55}")
print()

infsum25 = truetemperature(6, 4, 10, 5)
finite25 = temperature_dict["T_{2,5}"]
print(f"Infinte Sum Solution for Node (2,5): {infsum25}")
print(f"Finite Difference Solution for Node (2,5): {finite25}")
print()

print("These numbers are roughly in the same ballpark.")

```

Infinte Sum Solution for Node (2,2): 17.447648097867173  
 Finite Difference Solution for Node (2,2): 16.993525030234125

Infinte Sum Solution for Node (3,3): 19.582606176069465  
 Finite Difference Solution for Node (3,3): 18.966356165652122

Infinte Sum Solution for Node (5,5): 22.171785767274407  
 Finite Difference Solution for Node (5,5): 22.044938726868928

Infinte Sum Solution for Node (2,5): 26.383109973779252  
 Finite Difference Solution for Node (2,5): 24.44421566296228

These numbers are roughly in the same ballpark.

## 5 Part G: Error Analysis

[437]: *## loops through the entire matrix and finds maintains the highest amount of*  
*↪error (squared)*

```

## find a sum of the squared errors, take the root, then divide by the number
↳ of nodes (36) take the square root

print("Temperature Finite Difference Method:")
print(T_right)

Trightinf = np.zeros((6, 6))

# Set the bottom row to 15
Trightinf[0, :] = 15

# Set the right column to 15
Trightinf[:, 5] = 15

# Set the top row to 30
Trightinf[5, :] = 30
#Trightinf = np.fliplr(Trightinf)
#print(truetemperature(6, 4, 10, 5))
#print(Trightinf)

num_x = 5
num_y = 5

for i in range(num_x):
    for j in range(num_y):

        if j == 0: continue
        x = i + 5
        y = j

        posx = x+1-5
        posy = y+1

        #print(f"Calculating temp for x= {x} and y= {y}, position {posx},
↳ {posy}")
        truetemp = truetemperature(x, y, 10, 5)
        #print(truetemp)

        #print(f"inputting true temp {truetemp} at matrix pos {posx-1}, {posy}")
        Trightinf[posy-1, posx-1] = truetemp

Trightinf = np.flipud(Trightinf)
print("\nTemperatures Fourier Fine Series:")
print(Trightinf)
maxError = 0
sumError = 0

```

```

sumPercentError = 0
maxPercentError = 0

for i in range(6):
    for j in range(6):

        inf = Trightinf[i,j]
        finite = T_right[i,j]

        error = abs(inf - finite)
        percentError = error/finite * 100

        sumPercentError += percentError
        sumError += error
        if error > maxError: maxError = error
        if percentError > maxPercentError: maxPercentError = percentError

avg_error = sumError / 36
ave_percenterror = sumPercentError / 36

print()
print(f"Average Error: {avg_error}")
print(f"Average Percent Error: {ave_percenterror}")
print(f"Maximum Error: {maxError}")
print(f"Maximum Percent Error: {maxPercentError}")

```

Temperature Finite Difference Method:

```

[[30.    30.    30.    30.    30.    30.   ]
 [21.168 24.444 25.129 24.485 22.045 15.   ]
 [20.782 21.48  21.586 20.767 18.695 15.   ]
 [19.    19.11  18.966 18.302 16.967 15.   ]
 [16.997 16.994 16.867 16.51  15.869 15.   ]
 [15.    15.    15.    15.    15.    15.  ]]

```

Temperatures Fourier Fine Series:

```

[[30.    30.    30.    30.    30.    30.   ]
 [26.498 26.383 25.964 24.922 22.172 15.   ]
 [23.206 23.038 22.455 21.19  18.764 15.   ]
 [20.227 20.078 19.583 18.617 17.055 15.   ]
 [17.532 17.448 17.176 16.676 15.927 15.   ]
 [15.    15.    15.    15.    15.    15.  ]]

```

Average Error: 0.5207871530159269

Average Percent Error: 2.505129356511474

Maximum Error: 5.330705877148141

Maximum Percent Error: 25.183295621209478

## **6 Part G Continued**

### **6.1 Is this error acceptable?**

This error is acceptable, the maximum error however is significant on the half node wall at the middle. This is a result of our methodology of using half-nodes and quarter-nodes as a part of our methodology. Splitting the system in half has caused a significant amount of error however the values outside of the border are very acceptable.

### **6.2 What could you do to reduce this error?**

Either to add more nodes, or to solve the system as a whole without doing half or quarter nodes just as a evenly spaced grid with all four boundary conditions known.