# EE374 Term Project Phase 2

*# Created main.py on 11.06.2023*

*#*

*# Author: Deniz Karakay (2443307)*

*# Copyright (c) 2023 Deniz Karakay*

*# EE374 - Term Project Phase 2*

*import* csv

*import* math

def termproject(text\_path: str, library\_path: str):

*"""*

*This function reads input data from the given text file path and library file path,*

*calculates the total resistance, inductance and shunt susceptance of the line in per unit.*

*Args:*

*text\_path (str): The file path of the input text file.*

*library\_path (str): The file path of the library file.*

*Returns:*

*List: A list containing the student ID and the calculated values.*

*"""*

text = {}

*with* open(text\_path, "r") *as* file:

*# Read the file and remove the \n*

lines = [line.strip() *for* line *in* file]

*# Get the values from the input file*

s\_base = float(lines[1])

v\_base = float(lines[3])

number\_of\_circuits = int(lines[5])

number\_of\_bundles = int(lines[7])

bundle\_distance = float(lines[9])

length\_of\_line = float(lines[11])

acsr\_name = lines[13]

*# Get the values from the input file and convert them to int*

c1\_phase\_c = [float(line) *for* line *in* lines[15:17]]

c1\_phase\_a = [float(line) *for* line *in* lines[18:20]]

c1\_phase\_b = [float(line) *for* line *in* lines[21:23]]

*# Create a dictionary with the values from text*

text = {

"s\_base": s\_base,

"v\_base": v\_base,

"number\_of\_circuits": number\_of\_circuits,

"number\_of\_bundles": number\_of\_bundles,

"bundle\_distance": bundle\_distance,

"length\_of\_line": length\_of\_line,

"acsr\_name": acsr\_name,

"c1\_phase\_c": c1\_phase\_c,

"c1\_phase\_a": c1\_phase\_a,

"c1\_phase\_b": c1\_phase\_b,

}

library = {}

*# Read library csv file*

*with* open(library\_path, "r") *as* file:

reader = csv.reader(file)

*# Exclude title row*

title\_row = next(reader)

*# Create a dictionary with the values from library,*

*# Set key as ACSR name and value as the rest of the row*

*for* row *in* reader:

*if* row[0]:

library[row[0]] = row[1:]

*# Get the ACSR name from the text*

acsr\_name = text["acsr\_name"]

*# Get the corresponding data from the library*

acsr\_data = library[acsr\_name]

*# Get only the data that we need*

*# Outside\_diameter, ac\_resistance, gmr*

acsr\_outside\_diameter\_in = acsr\_data[3]

ac\_resistance\_ohm\_over\_mi = acsr\_data[5]

acsr\_gmr\_ft = acsr\_data[6]

*# Convert the data to SI*

*# 1 in = 0.0254 m*

one\_inch\_in\_m = 0.0254

acsr\_outside\_diameter\_si = float(acsr\_outside\_diameter\_in) \* one\_inch\_in\_m

*# 1 mi = 1609.34 m*

one\_mile\_in\_m = 1609.34

acsr\_ac\_resistance\_ohm\_over\_m = float(ac\_resistance\_ohm\_over\_mi) \* 1 / one\_mile\_in\_m

ac\_resistance = acsr\_ac\_resistance\_ohm\_over\_m

*# 1 ft = 0.3048 m*

one\_foot\_in\_m = 0.3048

acsr\_gmr\_si = float(acsr\_gmr\_ft) \* one\_foot\_in\_m

*# 1 km = 1000 m*

length\_of\_line\_m = text["length\_of\_line"] \* 1000

t\_gmr = acsr\_gmr\_si

t\_r\_eq\_bundle = acsr\_outside\_diameter\_si / 2

*# Calculate the equivalent radius and GMR for the bundle based on the number of bundles*

*if* number\_of\_bundles == 1:

gmr\_bundle = t\_gmr

r\_eq\_bundle = t\_r\_eq\_bundle

*elif* number\_of\_bundles == 2:

gmr\_bundle = math.sqrt(t\_gmr \* bundle\_distance)

r\_eq\_bundle = math.sqrt(t\_r\_eq\_bundle \* bundle\_distance)

*elif* number\_of\_bundles == 3:

gmr\_bundle = (t\_gmr \* (bundle\_distance\*\*2)) \*\* (1 / 3)

r\_eq\_bundle = (t\_r\_eq\_bundle \* (bundle\_distance\*\*2)) \*\* (1 / 3)

*elif* number\_of\_bundles == 4:

gmr\_bundle = 1.09 \* ((t\_gmr \* (bundle\_distance\*\*3)) \*\* (1 / 4))

r\_eq\_bundle = 1.09 \* ((t\_r\_eq\_bundle \* (bundle\_distance\*\*3)) \*\* (1 / 4))

*elif* number\_of\_bundles == 5:

b\_dis\_sqr = bundle\_distance\*\*2

*# Use formula for diagonals of pentagon*

diagonal = bundle\_distance / 2 \* (math.sqrt(5) + 1)

gmr\_bundle = (t\_gmr \* (diagonal\*\*2) \* b\_dis\_sqr) \*\* (1 / 5)

r\_eq\_bundle = (t\_r\_eq\_bundle \* (diagonal\*\*2) \* b\_dis\_sqr) \*\* (1 / 5)

*elif* number\_of\_bundles == 6:

b\_dis\_sqr = bundle\_distance\*\*2

*# Use formula for diagonals of hexagon*

small\_d = math.sqrt(3) \* bundle\_distance

large\_d = 2 \* bundle\_distance

gmr\_bundle = (t\_gmr \* (small\_d\*\*2) \* large\_d \* b\_dis\_sqr) \*\* (1 / 6)

r\_eq\_bundle = (t\_r\_eq\_bundle \* (small\_d\*\*2) \* large\_d \* b\_dis\_sqr) \*\* (1 / 6)

*elif* number\_of\_bundles == 7:

b\_dis\_sqr = bundle\_distance\*\*2

*# Use formula for diagonals of heptagon*

large\_d = bundle\_distance / (2 \* math.sin(math.radians(90 / 7)))

small\_d = 2 \* bundle\_distance \* math.cos(math.radians(180 / 7))

gmr\_bundle = (t\_gmr \* (small\_d\*\*2) \* (large\_d\*\*2) \* b\_dis\_sqr) \*\* (1 / 7)

r\_eq\_bundle = (t\_r\_eq\_bundle \* (small\_d\*\*2) \* (large\_d\*\*2) \* b\_dis\_sqr) \*\* (

1 / 7

)

*elif* number\_of\_bundles == 8:

b\_dis\_sqr = bundle\_distance\*\*2

*# Use formula for diagonals of octagon*

large\_diagonal = bundle\_distance \* math.sqrt(4 + 2 \* math.sqrt(2))

medium\_diagonal = bundle\_distance \* (1 + math.sqrt(2))

small\_diagonal = bundle\_distance \* math.sqrt(2 + math.sqrt(2))

gmr\_bundle = (

t\_gmr

\* (small\_diagonal\*\*2)

\* (medium\_diagonal\*\*2)

\* (large\_diagonal)

\* b\_dis\_sqr

) \*\* (1 / 8)

*# Calculate the distance between the phases for ab*

phase\_ab\_x = c1\_phase\_a[0] - c1\_phase\_b[0]

phase\_ab\_y = c1\_phase\_a[1] - c1\_phase\_b[1]

distance\_ab = math.sqrt(phase\_ab\_x\*\*2 + phase\_ab\_y\*\*2)

*# Calculate the distance between the phases for bc*

phase\_bc\_x = c1\_phase\_b[0] - c1\_phase\_c[0]

phase\_bc\_y = c1\_phase\_b[1] - c1\_phase\_c[1]

distance\_bc = math.sqrt(phase\_bc\_x\*\*2 + phase\_bc\_y\*\*2)

*# Calculate the distance between the phases for ca*

phase\_ca\_x = c1\_phase\_c[0] - c1\_phase\_a[0]

phase\_ca\_y = c1\_phase\_c[1] - c1\_phase\_a[1]

distance\_ca = math.sqrt(phase\_ca\_x\*\*2 + phase\_ca\_y\*\*2)

*# Calculate GMD take third root of the product of the distances*

gmd = (distance\_ab \* distance\_bc \* distance\_ca) \*\* (1 / 3)

*# For Earth effect*

h\_a = 2 \* c1\_phase\_a[1]

h\_b = 2 \* c1\_phase\_b[1]

h\_c = 2 \* c1\_phase\_c[1]

*# Same as distance between phases*

h\_ab = math.sqrt((phase\_ab\_x\*\*2) + (c1\_phase\_a[1] + c1\_phase\_b[1]) \*\* 2)

h\_bc = math.sqrt((phase\_bc\_x\*\*2) + (c1\_phase\_b[1] + c1\_phase\_c[1]) \*\* 2)

h\_ca = math.sqrt((phase\_ca\_x\*\*2) + (c1\_phase\_c[1] + c1\_phase\_a[1]) \*\* 2)

*# Calculate the resistance of the conductor*

tot\_resistance = ac\_resistance \* length\_of\_line\_m / number\_of\_bundles

*# Calculate the inductance of the conductor*

inductance\_m = 2 \* 10\*\*-7 \* math.log(gmd / gmr\_bundle)

tot\_inductance = 2 \* 50 \* math.pi \* inductance\_m \* length\_of\_line\_m

*# Calculate earth effect*

h\_root = (h\_ab \* h\_bc \* h\_ca) \*\* (1 / 3)

h\_root\_2 = (h\_a \* h\_b \* h\_c) \*\* (1 / 3)

*# Calculate the capacitance of the conductor*

capacitance\_num = 2 \* math.pi \* 8.854 \* 10\*\*-12

capacitance\_den\_first = math.log(gmd / r\_eq\_bundle)

capacitance\_den\_second = math.log(h\_root / h\_root\_2)

capacitance\_m = capacitance\_num / (capacitance\_den\_first - capacitance\_den\_second)

*# Calculate the susceptance of the conductor*

susceptance = 2 \* math.pi \* 50 \* capacitance\_m \* length\_of\_line\_m

*# Calculate the base values*

z\_base = v\_base\*\*2 / s\_base

*# Calculate the per unit values*

resistance\_pu = float(tot\_resistance / (z\_base))

inductance\_pu = float(tot\_inductance / (z\_base))

susceptance\_pu = float(susceptance / (1 / z\_base))

student\_id = float(2443307)

*# Create a list of the results*

result = [student\_id, resistance\_pu, inductance\_pu, susceptance\_pu]

*return* result

*if* \_\_name\_\_ == "\_\_main\_\_":

*# Run the function*

output = termproject("Input\_file\_example.txt", "library.csv")

*# Print the output*

print(output)