Lab I

Benjamin Coveler

Northwestern University

Civil Engineering 216

13 April 2023

Contents

List of Figures

List of Tables

1 Problem Statement

A 4-kip force P forming an angle D with the vertical is applied as shown to member *ABC*, which is supported by a pin and bracket at *C* and by a cable *BD* forming an angle *E* with the horizontal.

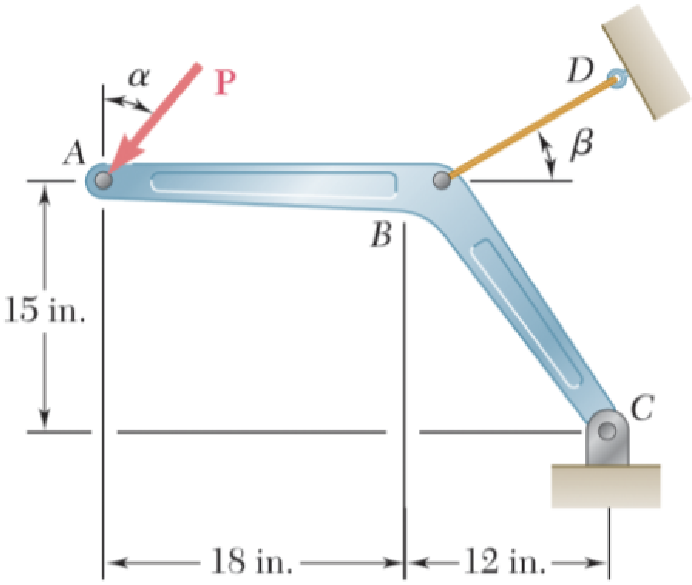

Figure 1: Given system

1.1 Part A Question

Knowing that the ultimate load of the cable is 25 kips, write a computer program to construct a table of the values of the factor of safety of the cable for values of α and β from 0 to 45 degrees, using increments in α and β corresponding to 0.1-degree increments in $\tan\alpha$ and $\tan\beta$.

1.2 Part B Question

Check that for any given value of α, the maximum value of the factor of safety is obtained for $\beta = 38.66°$ and explain why.

1.3 Part C Question

Determine the smallest possible value of the factor of safety for $\beta = 38.66°$, as well as the corresponding value of α, and explain the result obtained.

1.4 Unknowns

Table 1: Unknown Values to Solve For

Variables	**Symbols**	**Units**
Factor of Safety	FS	n/a
Tension in BD	T_{BD}	Newtons

1.5 Givens

Table 2: Known Values from Problems

Variables	**Symbols**	**Values**	**Units**
Applied Force	P	4	kips
Ultimate Load	F_{ult}	25	kips
Alpha	α	0->45	Degrees
Beta	β	0->45	Degrees
Length AB	L_{AB}	18	Inches
Horizontal Length BC	$L_{BC,h}$	12	Inches
Length BC	L_{BC}	19.2	Inches
Height of Overall System	H	15	Inches
Length of Overall System	L	30	Inches

1.6 Part A Process Outline/Task

In Part A, we will create an equation for the tension on the cable, using the definition of the moment about point C (see Figure 2 and the Theory Manual section). Next, we create an equation for the factor of safety, FS. Finally, we create a Python script to evaluate the effects of different choices of α and β on the FS.

1.7 Part B Process Outline/Task

In Part B, we use the output data from Part A (Figure 3) to verify that $\beta = 38.66°$ has the highest FS for all values of α.

1.8 Part C Process Outline/Task

Finally, in Part C, we use the table generated in Part A (Figure 3) to locate the *smallest* value of FS for $\beta = 38.66°$.

2 Theory Manual

2.1 Equation Derivation

The free body diagram for the system is shown in Figure 2 below:

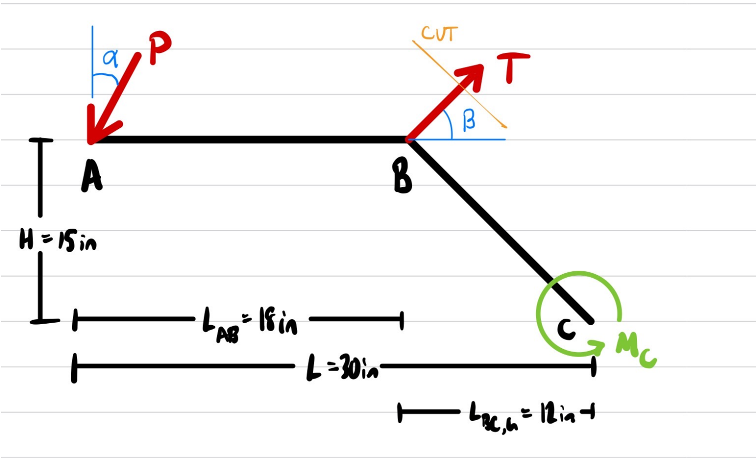

Figure 2: Free Body Diagram

In Figure 2, we draw a cut over element BD in order to isolate the internal forces – in this case, T_{BD}. This internal force T can be broken into its components, $T_{BD}\cos\beta$ and $T_{BD}\sin\beta$.

Next, we turn our attention to the force P. Like the tension force, it can be deconstructed into $P\cos\alpha$ and $P\sin\alpha$. Now, we may construct the equation for the moment about point C, since all relevant forces are found. We ignore the reaction forces at C, because the moment about C does not include forces that are on the point concerned.

$$\sum M_C = \sum(L \times F) = 0 \tag{1}$$

$$\sum M_C = LP\cos\alpha + HP\sin\alpha - HT\cos\beta - L_{BC,h}T\sin beta = 0 \tag{2}$$

$$30P\cos\alpha + 15P\sin\alpha = 15T\cos\beta + 12T\sin\beta \tag{3}$$

Now that we have found an equilibrium equation, we isolate T.

$$P(30\cos\alpha + 15\sin\alpha) = T(15\cos\beta + 12\sin\beta) \tag{4}$$

$$T = \frac{P(30\cos\alpha + 15\sin\alpha)}{(15\cos\beta + 12\sin\beta)} \tag{5}$$

Finally, we use our equation for the safety factor to identify the impact of α and β on the FS.

$$FS = \frac{\sigma_{ultimate}}{\sigma working} \tag{6}$$

$$FS = \frac{F_{ult}}{T} \tag{7}$$

$$FS = \frac{F_{ult}(15\cos\beta + 12\sin\beta)}{P(30\cos\alpha + 15\sin\alpha)} \tag{8}$$

$$\mathbf{FS} = \frac{\mathbf{25}}{\mathbf{4}} * \frac{(\mathbf{15}\cos\beta + \mathbf{12}\sin\beta)}{(\mathbf{30}\cos\alpha + \mathbf{15}\sin\alpha)} \tag{9}$$

Equation (10) represents the equation we will use in Python, with the given ranges for α and β. Parts B and C will rely on the figure generated by this equation.

3 Programmer Manual

3.1 Python Packages

Table 3: Packages Used

Package	Description
numpy	Numerical methods/matrix analysis
pandas	Database sorting
openpyxl	Microsoft Excel .xlsx format exporting

3.2 Variables and Constants

Table 4: Variables and Constants Used

Variable	Description
`tanAlpha`	Array storing values of $tan\alpha$
`alphaR`	Array storing values of α in radians
`alpha`	Array storing values of α in degrees
`tanBeta`	Array storing values of $tan\beta$
`betaR`	Array storing values of β in radians
`beta`	Array storing values of β in degrees
`P`	Variable storing integer value of force
`fUlt`	Variable storing integer value of ultimate load
`numerator`	Variable storing the top half of the symbolic fraction for `tension`
`denominator`	Variable storing the bottom half of the `tension` fraction
`tension`	Variable storing the calculated (numerical) solutions based on values of `alpha` and `beta`.
`factor`	Variable storing the full numerical FS solution
`resultMatrix`	Array storing results of FS calculations

3.3 Functions and Operations

Table 5: Functions and Operations used in Scripts, in Order of Appearance

Function or Operator	Description
`np.sin/np.cos/np.(arc)tan`	Operators that take in an int
`np.arange`	Function that creates an array based on numerical parameters
`np.array`	Identifier for a `numpy` array structure
`np.rad2deg/np.deg2rad`	Functions that convert radians to degrees (and back)
`fos`	Function that takes in two arrays. Uses the FS equation from Part 1
`np.empty`	Function that creates an empty matrix with given dimensions
`for`	Loop statement that runs until a parameter is met
`np.column-stack`	Function that vertically stacks matrices
`np.row-stack`	Function that horizontally stacks matrices
`np.insert`	Function that adds a value into a matrix
`pd.DataFrame`	Function that converts a matrix into a data format called a dataframe
`pd.to-excel`	Function that converts a dataframe into an Excel spreadsheet
$+$	Addition operator
-	Subtraction operator
*	Multiplication operator
/	Division operator
print	Function that prints a specified statement

3.4 Code Outline

- Create specified ranges for `tanAlpha` and `tanBeta`
- Convert these arrays into radians and degrees
- Define function based on Equation (10)
- Initialise empty result matrix, with dimensions given by lengths of `tanAlpha` and `tanBeta`
- Use `for` loop to go through each value of α and β and add resulting FS value to the result matrix
- Append the α and β intervals to the top and side to serve as axis labels
- Export the final result matrix to an Excel spreadsheet for easier viewing

4 Results and Analysis

		Beta										
		0.00	5.71	11.31	16.70	21.80	26.57	30.96	34.99	38.66	41.99	45.00
Alpha	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	5.71	0.00	3.21	3.40	3.55	3.67	3.75	3.80	3.82	3.83	3.82	3.81
	11.31	0.00	3.11	3.30	3.44	3.55	3.63	3.68	3.70	3.71	3.70	3.69
	16.70	0.00	3.05	3.23	3.37	3.48	3.55	3.60	3.63	3.63	3.63	3.61
	21.80	0.00	3.01	3.19	3.33	3.44	3.51	3.56	3.58	3.59	3.59	3.57
	26.57	0.00	3.00	3.18	3.32	3.43	3.50	3.55	3.57	3.58	3.57	3.56
	30.96	0.00	3.01	3.19	3.33	3.44	3.51	3.56	3.58	3.59	3.58	3.57
	34.99	0.00	3.04	3.21	3.36	3.46	3.54	3.59	3.61	3.62	3.61	3.60
	38.66	0.00	3.07	3.25	3.40	3.50	3.58	3.63	3.65	3.66	3.65	3.64
	41.99	0.00	3.12	3.30	3.44	3.55	3.63	3.68	3.71	3.71	3.71	3.69
	45.00	0.00	3.17	3.35	3.50	3.61	3.69	3.74	3.77	3.77	3.77	3.75

Figure 3: FS in terms of α and β

4.1 Part B

For each row of Figure 3 - each value of α - the column $\beta = 38.66°$ contains the maximum FS value as predicted. We can see general trends from the data: as α increases to $26.57°$, the FS decreases until it reaches a local minimum at that point. Then, it rises until $45°$. Similarly, β has a local maximum at $38.66°$.

4.2 Part C

The minimum FS value at $\beta = 38.66°$ is 3.58, at $\alpha = 26.57°$. Intuitively, this makes sense - the load on the cable is largest at this choice of α, because the loads are roughly orthogonal to the load-bearing device.

4.2.1 Result Validation

From Equation (10), we have an expression for FS. We can substitute in the values used for Part C to validate Figure 3 against our mathematical model.

$$FS = \frac{25}{4} * \frac{(15\cos\beta + 12\sin\beta)}{(30\cos\alpha + 15\sin\alpha)}$$
$$FS = \frac{25}{4} * \frac{(15\cos 38.66 + 12\sin 38.66)}{(30\cos 26.57 + 15\sin 26.57)} \approx 3.58$$

This confirms the table entry, and supports the conclusion that the table data matches the equation.

5 Appendices and Attributions

5.1 Python Script (.py)

```python

import numpy as np
import pandas as pd
import openpyxl

# Create a range from 0 to 1 for tan(alpha)
tanAlpha = np.arange(0, np.tan(np.deg2rad(45)) + 0.1, 0.1)
# Convert the range to radians using arctan (inverse tangent)
alphaR = np.arctan(np.array(tanAlpha))
# Create another range, converted to 0->45 degrees.
# The numpy tangent function only takes/makes radian arguments without additional conversion.
alpha = np.rad2deg(np.array(alphaR))

# Perform the same operations for beta
tanBeta = np.arange(0, np.tan(np.deg2rad(45)) + 0.1, 0.1)
betaR = np.arctan(np.array(tanBeta))
beta = np.rad2deg(np.array(betaR))

# Define parameters from the problem statement
# P is the force exerted on the system, 4 kips
P = 4
# fUlt is the ultimate load (needed for FoS), 25 kips
fUlt = 25


# Define function to compute factor of safety for given alpha and beta arrays
def fos(a, b):
    numerator = 30 * np.cos(a) + 15 * np.sin(a)
    denominator = 15 * np.cos(b) + 12 * np.sin(b)
    tension = P * (numerator / denominator)
    factor = fUlt / tension
    return factor


resultMatrix = np.empty([tanAlpha.size, tanBeta.size])

for i in range(1, tanAlpha.size):
    for j in range(1,  tanBeta.size):
        resultMatrix[i, j] = fos(alphaR[i], betaR[j])

# Concatenate the matrices to get one unified matrix with axes
# Setting alpha as horizontal axis and beta as horizontal axis
resultMatrixBeta = np.column_stack((beta, resultMatrix))
# Append a 0 to the beginning of beta to match horizontal dimensions with alpha+results matrix
```

```
alpha0 = np.insert(alpha, 0, 0)
resultMatrixAlphaBeta = np.row_stack((alpha0, resultMatrixBeta))
print(resultMatrixAlphaBeta)
print('The maximum FoS of this system is', resultMatrix.max())

# Convert the result matrix into a dataframe
df = pd.DataFrame(resultMatrixAlphaBeta)

# Save to Excel spreadsheet in this .py file's project directory
filepath = 'FoSresults.xlsx'
df.to_excel(filepath, index=False)
```

5.2 Attributions

The author would like to thank Armaan Jain for his valuable help in proofing the Python code for this assignment.