deformsbook

Jake Grohs

2023-05-30

Table of contents

Preface

This is a Quarto book.

To learn more about Quarto books visit https://quarto.org/docs/books.

```
1 + 1
```

```
[1] 2
```

1 Strength of Materials Problem Workout

To scaffold your learning in this example, we have provided a free body diagram for you and a repeat of the problem statement.

A city planner is installing a new traffic light. Light A weighs 65 lb, while lights B and C weigh 50 lb each. The post at O has a hollow circular cross-section with an outer diameter of 5 inches and a wall thickness of 0.2 inches. It will be made from aluminum alloy with a tensile yield stress of 35 ksi and a compressive yield stress of 20 ksi. A factor of safety of 2 is required with respect to yield. You may ignore the weight of the post.

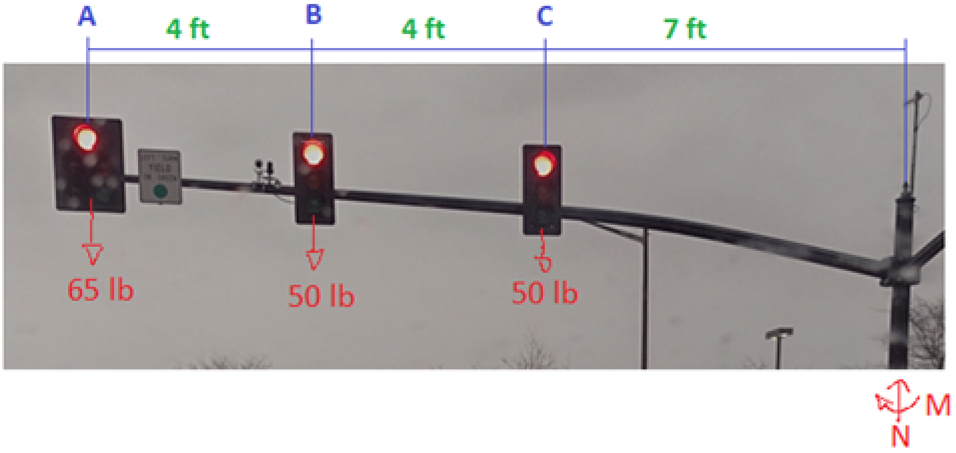

Figure 1.1: Figure 1: Three traffic light installation with loads

Please work through the problem step by step showing your math in the interactive interface here.

```
#| standalone: true
#| viewerHeight: 420
#| components: [viewer]
from shiny import App, render, ui, reactive, req
from sympy import solve, Eq, Symbol
from sympy.parsing.sympy_parser import parse_expr
from shiny.ui import h4
```

```
# load equations lists


class eqn:
    def __init__(self, name, inline_math, newline_math, working_sym, working_eqn_latex,wor
        self.name = name
        self.inline_math = inline_math
        self.newline_math = newline_math
        self.working_sym = working_sym
        self.working_eqn_latex = working_eqn_latex
        self.working_eqn_solver = working_eqn_solver

StaticsSumFx = eqn(
    "Equilibrium Forces in X",
    "\(\Sigma F_x=0\)",
    "$$\Sigma F_x=0$$",
    "SigmaFx",
    "$$F_x1+F_x2+F_x3+F_x4+F_x5=0$$",
    "F_x1+F_x2+F_x3+F_x4+F_x5=0"
)

StaticsSumFy = eqn(
    "Equilibrium Forces in Y",
    "\(\Sigma F_y=0\)",
    "$$\Sigma F_y=0$$",
    "SigmaFy",
    "$$F_y1+F_y2+F_y3+F_y4+F_y5=0$$",
    "F_y1+F_y2+F_y3+F_y4+F_y5=0"
)

StaticsSumM = eqn(
    "Equilibrium Moments about O",
    "\(\Sigma M_O=0\)",
    "$$\Sigma M_O=0$$",
    "SigmaM",
    "$$M_1+M_2+M_3+M_4+M_5=0$$",
    "M_1+M_2+M_3+M_4+M_5=0"
)

StressEqn = eqn(
    "Stress Equation",
    "\(\sigma=\\frac{F}{A}\)",
    "$$\sigma=\\frac{F}{A}$$",
    "sigma,F,A",
    "$$\sigma=\\frac{(F)}{(A)}$$",
    "Eq(sigma,(F)/(A))"
)

AxialDeform = eqn(
    "Axial Deformation by Force",
    "\(\delta_l=\\frac{P L}{AE}\)",
    "$$\delta_l=\\frac{P L}{AE}$$",
    "delta_l,P,L,A,E",
    "$$\delta_l=\\frac{(P)(L)}{(A)(E)}$$",
    "delta_l=(P)*(L)/(A)/(E)"
)

ThermalDeform = eqn(
    "Axial Deformation by Thermal",
    "\(\delta_t= \\alpha \Delta T L\)",
    "$$\delta_t= \\alpha \Delta T L$$",
    "delta_t,alpha,DeltaT,L",
    "$$\delta_t= \\alpha \Delta T L$$",
    "delta_t= alpha*(Delta_T)*L"
)

AreaTube = eqn(
    "Area of a Tube",
    "\(A_{tube}=\pi(r_o^2-r_i^2)\)",
    "$$A_{tube}=\pi(r_o^2-r_i^2)$$",
    "A_tube,r_o,r_i",
    "$$A_{tube}=\pi(r_o^2-r_i^2)$$",
    "Eq(A_tube,pi*((r_o)**2-(r_i)**2))"
)

ITube = eqn(
    "Moment of Inertia of a Tube",
    "\(I_{tube}=\\frac{\pi}{4}(r_o^4-r_i^4)\)",
    "$$I_{tube}=\\frac{\pi}{4}(r_o^4-r_i^4)$$",
    "I_tube,r_o,r_i",
    "$$I_{tube}=\\frac{\pi}{4}(r_o^4-r_i^4)$$",
    "Eq(I_tube,pi/4*((r_o)**4-(r_i)**4))"
```

```
)


statics_eqnbank_inline = {
    StaticsSumFx.name: StaticsSumFx.inline_math,
    StaticsSumFy.name: StaticsSumFy.inline_math,
    StaticsSumM.name: StaticsSumM.inline_math,
}
deforms_eqnbank_inline = {
    StressEqn.name: StressEqn.inline_math,
    AxialDeform.name: AxialDeform.inline_math,
    ThermalDeform.name: ThermalDeform.inline_math,
}

geom_eqnbank_inline = {
    AreaTube.name: AreaTube.inline_math,
    ITube.name: ITube.inline_math,
}

eqnbank_inline = {
    StaticsSumFx.name: StaticsSumFx.inline_math,
    StaticsSumFy.name: StaticsSumFy.inline_math,
    StaticsSumM.name: StaticsSumM.inline_math,
    StressEqn.name: StressEqn.inline_math,
    AxialDeform.name: AxialDeform.inline_math,
    ThermalDeform.name: ThermalDeform.inline_math,
    AreaTube.name: AreaTube.inline_math,
    ITube.name: ITube.inline_math,
}

eqnbank_newline = {
    StaticsSumFx.name: StaticsSumFx.newline_math,
    StaticsSumFy.name: StaticsSumFy.newline_math,
    StaticsSumM.name: StaticsSumM.newline_math,
    StressEqn.name: StressEqn.newline_math,
    AxialDeform.name: AxialDeform.newline_math,
    ThermalDeform.name: ThermalDeform.newline_math,
    AreaTube.name: AreaTube.newline_math,
    ITube.name: ITube.newline_math,
}
```

```
working_equations_solver=reactive.Value([])
working_symbols=reactive.Value([])

app_ui = ui.page_fluid(
    ui.head_content(
        ui.tags.script(
            src="https://mathjax.rstudio.com/latest/MathJax.js?config=TeX-AMS-MML_HTMLorMM
        ),
        ui.tags.script(
            "if (window.MathJax) MathJax.Hub.Queue(['Typeset', MathJax.Hub]);"
        ),
    ),
    ui.panel_title("Interactive Problem Solving Environment"),
        ui.row(
            ui.markdown("Your Equation Workspace"),
            ui.column(3,ui.output_ui("dyn_eqns"),style='margin-bottom:30 px;border-right:1
            ui.column(9,ui.output_ui("dyn_working_eqns"),ui.output_text("txt")),
        ),
        ui.row(ui.output_ui("ui_equation_bookkeeping")),
        #ui.row(ui.input_action_button(
        #            "solveEquations", "Solve Equations", class_="btn-success", width="240
        #      ),
        ui.output_ui("ui_solutions"),
        ui.row(
            ui.column(8,ui.output_ui("dyn_ui_nav")),
            ui.column(4,
                ui.navset_tab_card(
                    ui.nav_spacer(),
                    ui.nav("Equation Bank",
                        ui.input_checkbox_group("selected_eqns","Choose your equations:",e
                        ),
                    ),
                ),
            ),
        ),
)


def server(input, output, session):

    @output
    @render.ui
    def dyn_eqns():
        eqns_keys = input.selected_eqns()
        req(eqns_keys)
        lookup_eqns = [eqnbank_newline[key] for key in eqns_keys]
        mystring_eqns = "".join(lookup_eqns)

        return [
            ui.markdown(mystring_eqns),
            ui.tags.script(
                "if (window.MathJax) MathJax.Hub.Queue(['Typeset', MathJax.Hub]);"
            ),
        ]

    @output
    @render.ui
    def dyn_working_eqns():
        eqns_keys = input.selected_eqns()
        req(eqns_keys)
        lookup_eqns = [eqnbank_newline[key] for key in eqns_keys]


        # Dynamic Filling of Force equations
        if StaticsSumFy.newline_math in lookup_eqns:
            StaticsSumFy_list = ["F_y1","F_y2","F_y3","F_y4","F_y5"]
            StaticsSumFy_list = StaticsSumFy_list[:input.NumForcesY()]
            StaticsSumFy.working_sym = ",".join(StaticsSumFy_list)
            StaticsSumFy.working_eqn_latex = "$$" + "+".join(StaticsSumFy_list) + "=0$$"
            StaticsSumFy.working_eqn_solver = "+".join(StaticsSumFy_list)

            if str(input.F1y()) != "" :
                StaticsSumFy.working_eqn_latex = StaticsSumFy.working_eqn_latex.replace("F
                StaticsSumFy.working_sym = StaticsSumFy.working_sym.replace("F_y1",str(inp
                StaticsSumFy.working_eqn_solver = StaticsSumFy.working_eqn_solver.replace(
            else:
                StaticsSumFy.working_eqn_latex = StaticsSumFy.working_eqn_latex.replace("F

            if str(input.F2y()) != "" :
                StaticsSumFy.working_eqn_latex = StaticsSumFy.working_eqn_latex.replace("F
                StaticsSumFy.working_sym = StaticsSumFy.working_sym.replace("F_y2",str(inp
                StaticsSumFy.working_eqn_solver = StaticsSumFy.working_eqn_solver.replace(
            else:
                StaticsSumFy.working_eqn_latex = StaticsSumFy.working_eqn_latex.replace("F

            if str(input.F3y()) != "" :
                StaticsSumFy.working_eqn_latex = StaticsSumFy.working_eqn_latex.replace("F
                StaticsSumFy.working_sym = StaticsSumFy.working_sym.replace("F_y3",str(inp
                StaticsSumFy.working_eqn_solver = StaticsSumFy.working_eqn_solver.replace(
            else:
                StaticsSumFy.working_eqn_latex = StaticsSumFy.working_eqn_latex.replace("F

            if str(input.F4y()) != "" :
                StaticsSumFy.working_eqn_latex = StaticsSumFy.working_eqn_latex.replace("F
                StaticsSumFy.working_sym = StaticsSumFy.working_sym.replace("F_y4",str(inp
                StaticsSumFy.working_eqn_solver = StaticsSumFy.working_eqn_solver.replace(
            else:
                StaticsSumFy.working_eqn_latex = StaticsSumFy.working_eqn_latex.replace("F

            if str(input.F5y()) != "" :
                StaticsSumFy.working_eqn_latex = StaticsSumFy.working_eqn_latex.replace("F
                StaticsSumFy.working_sym = StaticsSumFy.working_sym.replace("F_y5",str(inp
                StaticsSumFy.working_eqn_solver = StaticsSumFy.working_eqn_solver.replace(
            else:
                StaticsSumFy.working_eqn_latex = StaticsSumFy.working_eqn_latex.replace("F

        # Dynamic Filling of Moment equations
        if StaticsSumM.newline_math in lookup_eqns:
            StaticsSumM_list = ["M_1","M_2","M_3","M_4","M_5"]
            StaticsSumM_list = StaticsSumM_list[:input.NumMoments()]
            StaticsSumM.working_sym = ",".join(StaticsSumM_list)
            StaticsSumM.working_eqn_latex = "$$" + "+".join(StaticsSumM_list) + "=0$$"
            StaticsSumM.working_eqn_solver = "+".join(StaticsSumM_list)

            if str(input.M1()) != "" :
                StaticsSumM.working_eqn_latex = StaticsSumM.working_eqn_latex.replace("M_1
                StaticsSumM.working_sym = StaticsSumM.working_sym.replace("M_1",str(input.
                StaticsSumM.working_eqn_solver = StaticsSumM.working_eqn_solver.replace("M
            else:
                StaticsSumM.working_eqn_latex = StaticsSumM.working_eqn_latex.replace("M_1

            if str(input.M2()) != "" :
                StaticsSumM.working_eqn_latex = StaticsSumM.working_eqn_latex.replace("M_2
                StaticsSumM.working_sym = StaticsSumM.working_sym.replace("M_2",str(input.
                StaticsSumM.working_eqn_solver = StaticsSumM.working_eqn_solver.replace("M
            else:
                StaticsSumM.working_eqn_latex = StaticsSumM.working_eqn_latex.replace("M_2

            if str(input.M3()) != "" :
                StaticsSumM.working_eqn_latex = StaticsSumM.working_eqn_latex.replace("M_3
                StaticsSumM.working_sym = StaticsSumM.working_sym.replace("M_3",str(input.
                StaticsSumM.working_eqn_solver = StaticsSumM.working_eqn_solver.replace("M
            else:
                StaticsSumM.working_eqn_latex = StaticsSumM.working_eqn_latex.replace("M_3

            if str(input.M4()) != "" :
                StaticsSumM.working_eqn_latex = StaticsSumM.working_eqn_latex.replace("M_4
                StaticsSumM.working_sym = StaticsSumM.working_sym.replace("M_4",str(input.
                StaticsSumM.working_eqn_solver = StaticsSumM.working_eqn_solver.replace("M
            else:
                StaticsSumM.working_eqn_latex = StaticsSumM.working_eqn_latex.replace("M_4

            if str(input.M5()) != "" :
                StaticsSumM.working_eqn_latex = StaticsSumM.working_eqn_latex.replace("M_5
                StaticsSumM.working_sym = StaticsSumM.working_sym.replace("M_5",str(input.
                StaticsSumM.working_eqn_solver = StaticsSumM.working_eqn_solver.replace("M
            else:
                StaticsSumM.working_eqn_latex = StaticsSumM.working_eqn_latex.replace("M_5


        # Dynamic Filling of A equations
        if AreaTube.newline_math in lookup_eqns:
            AreaTube.working_eqn_latex = AreaTube.newline_math
            AreaTube.working_eqn_solver = "Eq(A_tube,pi*((r_o)**2-(r_i)**2))"
            AreaTube.working_sym = "A_tube,r_o,r_i"

            if str(input.A_tube()) != "" :
                AreaTube.working_eqn_latex = AreaTube.working_eqn_latex.replace("A_{tube}"
                AreaTube.working_sym = AreaTube.working_sym.replace("A_tube",str(input.A_t
                AreaTube.working_eqn_solver = AreaTube.working_eqn_solver.replace("A_tube"
            else:
                AreaTube.working_eqn_latex = AreaTube.working_eqn_latex.replace("A_{tube}"
            if str(input.Ar_o()) != "" :
                AreaTube.working_eqn_latex = AreaTube.working_eqn_latex.replace("r_o",str(
                AreaTube.working_sym = AreaTube.working_sym.replace("r_o",str(input.Ar_o()
                AreaTube.working_eqn_solver = AreaTube.working_eqn_solver.replace("r_o",st
            else:
                AreaTube.working_eqn_latex = AreaTube.working_eqn_latex.replace("r_o","\\b
            if str(input.Ar_i()) != "" :
                AreaTube.working_eqn_latex = AreaTube.working_eqn_latex.replace("r_i",str(
                AreaTube.working_sym = AreaTube.working_sym.replace("r_i",str(input.Ar_i()
                AreaTube.working_eqn_solver = AreaTube.working_eqn_solver.replace("r_i",st
            else:
                AreaTube.working_eqn_latex = AreaTube.working_eqn_latex.replace("r_i","\\b

        # Dynamic Filling of I equations
        if ITube.newline_math in lookup_eqns:
            ITube.working_eqn_latex = ITube.newline_math
            ITube.working_eqn_solver = "Eq(I_tube,pi/4*((r_o)**4-(r_i)**4))"
            ITube.working_sym = "I_tube,r_o,r_i"
            if str(input.I_tube()) != "" :
                ITube.working_eqn_latex = ITube.working_eqn_latex.replace("I_{tube}",str(i
                ITube.working_sym = ITube.working_sym.replace("I_tube",str(input.I_tube())
                ITube.working_eqn_solver = ITube.working_eqn_solver.replace("I_tube",str(i
            else:
                ITube.working_eqn_latex = ITube.working_eqn_latex.replace("I_{tube}","\\bo
            if str(input.Ir_o()) != "" :
                ITube.working_eqn_latex = ITube.working_eqn_latex.replace("r_o",str(input.
                ITube.working_sym = ITube.working_sym.replace("r_o",str(input.Ir_o()))
                ITube.working_eqn_solver = ITube.working_eqn_solver.replace("r_o",str(inpu
            else:
                ITube.working_eqn_latex = ITube.working_eqn_latex.replace("r_o","\\boxed{r
            if str(input.Ir_i()) != "" :
                ITube.working_eqn_latex = ITube.working_eqn_latex.replace("r_i",str(input.
                ITube.working_sym = ITube.working_sym.replace("r_i",str(input.Ir_i()))
                ITube.working_eqn_solver = ITube.working_eqn_solver.replace("r_i",str(inpu
            else:
                ITube.working_eqn_latex = ITube.working_eqn_latex.replace("r_i","\\boxed{r

        # Dynamic Filling of Stress equation
        if StressEqn.newline_math in lookup_eqns:
            StressEqn.working_eqn_latex = StressEqn.newline_math
            StressEqn.working_eqn_solver = "Eq(sigma,(F)/(A))"
            StressEqn.working_sym = "sigma,F,A"
        if str(input.sigma()) != "" :
            StressEqn.working_eqn_latex = StressEqn.working_eqn_latex.replace("\sigma",str
            StressEqn.working_eqn_solver = StressEqn.working_eqn_solver.replace("sigma",st
            StressEqn.working_sym = StressEqn.working_sym.replace("sigma",str(input.sigma(
        else:
            StressEqn.working_eqn_latex = StressEqn.working_eqn_latex.replace("\sigma","\\
        if str(input.force()) != "" :
            StressEqn.working_eqn_latex = StressEqn.working_eqn_latex.replace("F",str(inpu
            StressEqn.working_eqn_solver = StressEqn.working_eqn_solver.replace("F",str(in
            StressEqn.working_sym = StressEqn.working_sym.replace("F",str(input.force()))
        else:
            StressEqn.working_eqn_latex = StressEqn.working_eqn_latex.replace("F","\\boxed
        if str(input.area()) != "" :
            StressEqn.working_eqn_latex = StressEqn.working_eqn_latex.replace("A",str(inpu
            StressEqn.working_eqn_solver = StressEqn.working_eqn_solver.replace("A",str(in
            StressEqn.working_sym = StressEqn.working_sym.replace("A",str(input.area()))
        else:
            StressEqn.working_eqn_latex = StressEqn.working_eqn_latex.replace("A","\\boxed

        eqnbank_working_latex = {
        StaticsSumFx.name: StaticsSumFx.working_eqn_latex,
        StaticsSumFy.name: StaticsSumFy.working_eqn_latex,
        StaticsSumM.name: StaticsSumM.working_eqn_latex,
        StressEqn.name: StressEqn.working_eqn_latex,
        AxialDeform.name: AxialDeform.working_eqn_latex,
        ThermalDeform.name: ThermalDeform.working_eqn_latex,
        AreaTube.name: AreaTube.working_eqn_latex,
        ITube.name: ITube.working_eqn_latex,
        }

        eqnbank_working_solver = {
        StaticsSumFx.name: StaticsSumFx.working_eqn_solver,
        StaticsSumFy.name: StaticsSumFy.working_eqn_solver,
        StaticsSumM.name: StaticsSumM.working_eqn_solver,
        StressEqn.name: StressEqn.working_eqn_solver,
        AxialDeform.name: AxialDeform.working_eqn_solver,
        ThermalDeform.name: ThermalDeform.working_eqn_solver,
        AreaTube.name: AreaTube.working_eqn_solver,
        ITube.name: ITube.working_eqn_solver,
        }

        symbank_working = {
        StaticsSumFx.name: StaticsSumFx.working_sym,
        StaticsSumFy.name: StaticsSumFy.working_sym,
        StaticsSumM.name: StaticsSumM.working_sym,
        StressEqn.name: StressEqn.working_sym,
        AxialDeform.name: AxialDeform.working_sym,
        ThermalDeform.name: ThermalDeform.working_sym,
        AreaTube.name: AreaTube.working_sym,
        ITube.name: ITube.working_sym,
        }

        working_eqns_latex = [eqnbank_working_latex[key] for key in eqns_keys]
        working_eqns_solver = [eqnbank_working_solver[key] for key in eqns_keys]
        working_syms = [symbank_working[key] for key in eqns_keys]
        mystring_working_eqns = "".join(working_eqns_latex)
        working_equations_solver.set(working_eqns_solver)

        working_syms_only=[]
        for j in working_syms:
            temp=j.split(",")
            for k in temp:
                try:
                    float(k)
                except:
                    working_syms_only.append(k)
        working_syms_only=list(dict.fromkeys(working_syms_only))
        working_symbols.set(working_syms_only)

        return [
            ui.markdown(mystring_working_eqns),
            ui.tags.script(
                "if (window.MathJax) MathJax.Hub.Queue(['Typeset', MathJax.Hub]);"
            )
        ]

    @output
    @render.ui
    def dyn_ui_nav():

        equations = ui.navset_tab_card(
            ui.nav(
                str(StaticsSumFy.inline_math),
                ui.input_numeric("NumForcesY","How many terms do you want?",2,min=2,max=5)
                ui.input_text("F1y","\(F_{y_1}=\)", placeholder="Please type in variable o
                ui.input_text("F2y","\(F_{y_2}=\)", placeholder="Please type in variable o
                ui.panel_conditional("input.NumForcesY>=3", ui.input_text("F3y","\(F_{y_3}
                ui.panel_conditional("input.NumForcesY>=4", ui.input_text("F4y","\(F_{y_4}
                ui.panel_conditional("input.NumForcesY>=5", ui.input_text("F5y","\(F_{y_5}
                ),
            ui.nav(
                str(StaticsSumM.inline_math),
                ui.input_numeric("NumMoments","How many terms do you want?",2,min=2,max=5)
                ui.input_text("M1","\(M_1=\)", placeholder="Please type in variables or va
                ui.input_text("M2","\(M_2=\)", placeholder="Please type in variables or va
                ui.panel_conditional("input.NumMoments>=3", ui.input_text("M3","\(M_3=\)",
                ui.panel_conditional("input.NumMoments>=4", ui.input_text("M4","\(M_4=\)",
                ui.panel_conditional("input.NumMoments>=5", ui.input_text("M5","\(M_5=\)",
                ),
            ui.nav(
                str(AreaTube.inline_math),
                ui.input_text("A_tube","\(A_{tube}=\)", placeholder="Please type in variab
                ui.input_text("Ar_o","\(r_o=\)", placeholder="Please type in variables or
                ui.input_text("Ar_i","\(r_i\)", placeholder="Please type in variables or v
                ),
            ui.nav(
                str(ITube.inline_math),
                ui.input_text("I_tube","\(I_{tube}=\)", placeholder="Please type in variab
                ui.input_text("Ir_o","\(r_o=\)", placeholder="Please type in variables or
                ui.input_text("Ir_i","\(r_i\)", placeholder="Please type in variables or v
                ),
            ui.nav(
                str(StressEqn.inline_math),
                ui.input_text("sigma","\(\sigma\)", placeholder="Please type in variables
                ui.input_text("force","\(F\)", placeholder="Please type in variables or va
                ui.input_text("area","\(A\)", placeholder="Please type in variables or val
                )
            )

        return [equations,
               ui.tags.script(
                "if (window.MathJax) MathJax.Hub.Queue(['Typeset', MathJax.Hub]);"
        ),]

    @output
    @render.ui
    def ui_equation_bookkeeping():
        req(working_equations_solver())
        num_working_equations=len(working_equations_solver())
        num_working_symbols=len(working_symbols())
        string_working_symbols= ",".join(working_symbols())
        return [ui.markdown(f"Your equation-solver set up currently has **{num_working_equ
                ui.input_action_button(
                    "solveEquations", "Solve Equations", class_="btn-success", width="240p
                ui.tags.script(
                "if (window.MathJax) MathJax.Hub.Queue(['Typeset', MathJax.Hub]);"
                )]

    @output
    @render.ui
    @reactive.event(input.solveEquations)
    def ui_solutions():
        for j in working_symbols():
            j=Symbol(j)
        print(working_equations_solver())
        print(working_symbols())
        my_solver_equations=[]
        for m in working_equations_solver():
            m=parse_expr(m)
        solve_eqns = solve(working_equations_solver(),working_symbols(),dict=True)
        answers=[]
        for k in working_symbols():
            try:
                temp=solve_eqns[0][parse_expr(k)]
                temp2="$$"+k+"="+str(temp)+"$$"
                answers.append(temp2)
            except:
                pass
        mystring_answers="".join(answers)
        mystring_answers=mystring_answers.replace("pi","\pi")
        mystring_answers=mystring_answers.replace("delta","\delta")
        mystring_answers=mystring_answers.replace("delta","\Delta")
        mystring_answers=mystring_answers.replace("sigma","\sigma")
        return [ui.markdown(f"Your solution is {mystring_answers}"),
                ui.tags.script(
                "if (window.MathJax) MathJax.Hub.Queue(['Typeset', MathJax.Hub]);"
                )]


app = App(app_ui, server)
```

2 Workout Example Solution

2.1 Worked Out Solution

This demonstrates a worked out solution to the problem. The best way to begin is by drawing a free body diagram.

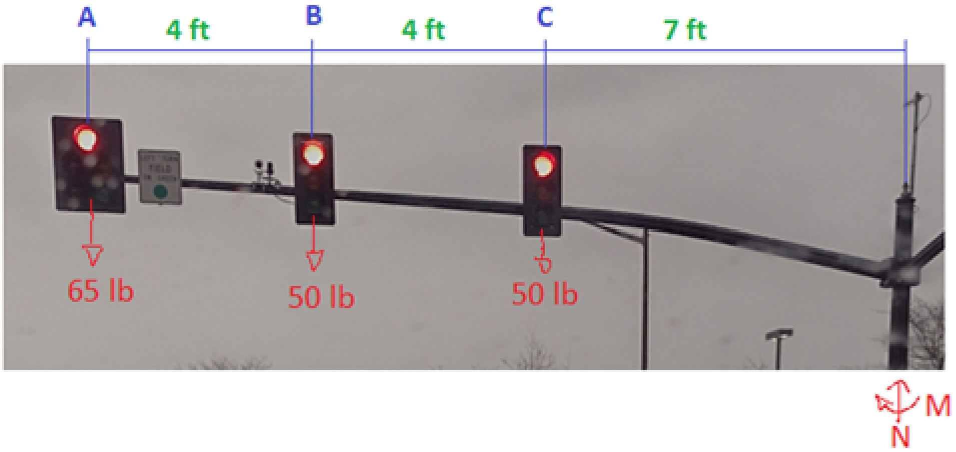

Figure 2.1: Figure 3: Three traffic light installation with loads

Use equilibrium equations to find the internal loads:

$$\Sigma F_y = 0 : N - 65 - 50 - 50 = 0$$
$$N = 165 \; lbs$$

$$\Sigma M_O = 0 : -M + (50 \times 7) + (50 \times 11) + (65 \times 15) = 0$$
$$M = 1875 \; lb \cdot ft = 22500 \; lb \cdot in$$

Now, determine the cross-sectional properties:

$$A = \pi(r_0^2 - r_i^2) = \pi(2.5^2 - 2.3^2) = 3.02 \; in^2 I = \frac{\pi}{4}(r_0^4 - r_i^4) = \frac{\pi}{4}(2.5^4 - 2.3^4) = 8.70 \; in^4$$

Calculate stress due to normal force:

$$\sigma_n = \frac{F}{A} = \frac{-165\ lbs}{3.02\ in^2} = -54.7\ psi$$

Calculate maximum stress due to bending moment (will have same magnitude in both tension and compression):

$$\sigma_m = \pm\frac{M_c}{I} = \pm\frac{22500 \times 2.5}{8.70} = \pm 6460\ psi$$

Determine combined tensile stress: $\sigma_T = -54.7 + 6460 = 6410\ psi$

Determine combined compressive stress: $\sigma_T = -54.7 - 6460 = -6520\ psi$

3 Summary

In summary, this book has no content whatsoever.

```
1 + 1
```

```
[1] 2
```

References