Strength of Materials Problem Exercises

Curated by James Lord, Jake Grohs, …

2024-01-05

Table of contents

Welcome to Demo Site **3**

I Problem 1 **4**

Static Problem Statement **5**
Problem Statement 5
Worked Out Solution 5

Dynamic Problem Statement **7**
Problem Image 7

Interactive Problem Interface **10**

II Problem 391 **43**

Dynamic Problem Statement **44**
Problem Image 44

1 Summary **47**

References **48**

Welcome to Demo Site

Welcome to this demonstration site of the Strength of Materials Open Problem Exercises companion to the Strength of Materials Open Textbook.

At this time, we are simply using this site as a demonstration and shell for our ongoing work. The intent is to demonstrate a more traditional static style problem exercise pack along with dynamic versions which allow students to quickly check answers and receive basic feedback, and/or to input their math in an interactive interface which will provide them with targeted feedback based on their atempted solution.

This work is still very much in progress and you may find bugs. We would welcome any input or feedback you have about this. Thanks!

Part I

Problem 1

Static Problem Statement

This is a static rendering of the problem with fixed variables that correspond to the variables used in the problem solution provided.

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. Please calculate the magnitude of the maximum combined stress in the post. You may ignore the weight of the post.

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

Worked Out Solution

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

Use equilibrium equations to find the internal loads:

$$\Sigma F_y = 0 : N - 65 - 50 - 50 = 0$$

$$N = 165 \; lbs$$

Figure 2: Figure 3: Three traffic light installation with loads

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

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: $\sigma_T = -54.7 - 6460 = -6520\ psi$

Dynamic Problem Statement

This is a dynamic rendering of the problem with dynamic variables based on the username entered. Please note at this time that the figure displays incorrect values. This will be corrected when drawn by the graphic artist.

Problem Image

Figure 3: Figure 1: Three traffic light installation with loads

```
#| standalone: true
#| viewerHeight: 600
#| components: [viewer]

from shiny import App, render, ui, reactive
import random
import asyncio
import io
import math
from datetime import datetime
from pathlib import Path

problem_ID="1"
light_a=reactive.Value("__")
```

```
lights_bc=reactive.Value("__")
attempts=["Timestamp,Attempt,Answer,Feedback\n"]

app_ui = ui.page_fluid(
    ui.markdown("**Please enter your ID number from your instructor and click to generate
    ui.input_text("ID","", placeholder="Enter ID Number Here"),
    ui.input_action_button("generate_problem", "Generate Problem", class_="btn-primary"),
    ui.markdown("**Problem Statement**"),
    ui.output_ui("ui_problem_statement"),
    ui.input_text("answer","Your Answer in units of psi", placeholder="Please enter your a
    ui.input_action_button("submit", "Submit Answer", class_="btn-primary"),
    ui.download_button("download", "Download File to Submit", class_="btn-success"),
)


def server(input, output, session):
    @output
    @render.ui
    def ui_problem_statement():
        return[ui.markdown(f"A city planner is installing a new traffic light. Light A wei

    @reactive.Effect
    @reactive.event(input.generate_problem)
    def randomize_vars():
        random.seed(input.ID())
        light_a.set(round(65+65*(.5-random.random())*.2))
        lights_bc.set(round(50+50*(.5-random.random())*.2))

    @reactive.Effect
    @reactive.event(input.submit)
    def _():
        instr= (light_a()+2*lights_bc()/math.pi*(2.5**2 - 2.3**2))+ ((-1*lights_bc()*7- li
        #check=math.isclose(float(input.answer()),instr,rel_tol=0.001)
        if math.isclose(float(input.answer()),instr,rel_tol=0.001):
           check="*Correct*"
        else:
           check="*Not Correct.*"

        if check=="*Not Correct.*" and math.isclose(abs(float(input.answer())),abs(instr),
           extra_check="An extra check says you may have a sign error."
        else:
```

```
            extra_check=""
        #extra_check = "An extra check says you may have a sign error." if math.isclose(ab
        feedback=ui.markdown(f"Your answer of {input.answer()} is {check} {extra_check} Fo
        attempts.append(f"{datetime.now()}, {input.submit()},{input.answer()},{check}\n")
        m=ui.modal(
          feedback,
          title="Feedback",
          easy_close=True
        )
        ui.modal_show(m)

    @session.download(
        filename=lambda: f"Problem_Log-{problem_ID}-{input.ID()}.csv"
    )
    async def download():
        # This version uses a function to generate the filename. It also yields data
        # multiple times.
        await asyncio.sleep(0.25)
        yield f"{problem_ID}_{input.submit()}_{input.ID()}\n"
        yield ''.join(attempts)


app = App(app_ui, server)
```

Interactive Problem Interface

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. Please calculate the magnitude of the maximum combined stress in the post. You may ignore the weight of the post.

Figure 4: 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: 600
#| components: [viewer]



import io
import numpy as np
import asyncio
from datetime import datetime
from pathlib import Path
import matplotlib.pyplot as plt
```

```
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",
    "\(\delta_l=\\frac{P L}{AE}\)",
    "$$\delta_l=\\frac{P\cdot L}{A \cdot E}$$",
    "delta_l,P,L,A,E",
    "$$\delta_l=\\frac{(P)(L)}{(A)(E)}$$",
    "Eq(delta_l,(P)*(L)/(A)/(E))"
)

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

BendingStress = eqn(
    "Bending Stress from a Moment",
    "\(\sigma_b=\\frac{M*y}{I}\)",
    "$$\sigma_b=\\frac{M*y}{I}$$",
    "sigma_b,M,y,I,",
    "$$\sigma_b=\\frac{M*y}{I}$$",
    "Eq(sigma_b,M*y/I))"
)

Compatability1 = eqn(
    "Compatability Equation 1",
    "\(a_1+\ldots=b_1+b_2+\ldots\)",
    "$$a_1+\ldots=b_1+b_2+\ldots$$",
    "",
    "$$a_1+a_n=b_1+b_n$$",
    "Eq(a_1+a_n=b_1+b_n)"
)

Compatability2 = eqn(
    "Compatability Equation 2",
    "\(c_1+\ldots=d_1+d_2+\ldots\)",
    "$$c_1+\ldots=d_1+d_2+\ldots$$",
    "",
    "$$c_1+c_n=d_1+d_n$$",
    "Eq(c_1+c_n=d_1+d_n)"
)


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,
    BendingStress.name: BendingStress.inline_math,
    AxialDeform.name: AxialDeform.inline_math,
    ThermalDeform.name: ThermalDeform.inline_math,
    AreaTube.name: AreaTube.inline_math,
    ITube.name: ITube.inline_math,
    Compatability1.name: Compatability1.inline_math,
    Compatability2.name: Compatability2.inline_math,
}

eqnbank_newline = {
    StaticsSumFx.name: StaticsSumFx.newline_math,
    StaticsSumFy.name: StaticsSumFy.newline_math,
    StaticsSumM.name: StaticsSumM.newline_math,
    StressEqn.name: StressEqn.newline_math,
    BendingStress.name: BendingStress.newline_math,
    AxialDeform.name: AxialDeform.newline_math,
    ThermalDeform.name: ThermalDeform.newline_math,
    AreaTube.name: AreaTube.newline_math,
    ITube.name: ITube.newline_math,
    Compatability1.name: Compatability1.newline_math,
    Compatability2.name: Compatability2.newline_math,
}



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

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

feedback_equations=reactive.Value([])
feedback_solns=reactive.Value([])
feedback_syms=reactive.Value([])

working_SumFx_render=reactive.Value("")
working_SumFy_render=reactive.Value("")
working_SumM_render=reactive.Value("")
working_StressEqn_render=reactive.Value("")
working_BendingStress_render=reactive.Value("")
working_AxialDeform_render=reactive.Value("")
working_ThermalDeform_render=reactive.Value("")
working_AreaTube_render=reactive.Value("")
working_Itube_render=reactive.Value("")
working_Compatability1_render=reactive.Value("")
working_Compatability2_render=reactive.Value("")

working_SumFx_string=reactive.Value("")
working_SumFy_string=reactive.Value("")
working_SumM_string=reactive.Value("")
working_StressEqn_string=reactive.Value("")
working_BendingStress_string=reactive.Value("")
working_AxialDeform_string=reactive.Value("")
working_ThermalDeform_string=reactive.Value("")
working_AreaTube_string=reactive.Value("")
working_Itube_string=reactive.Value("")
working_Compatability1_string=reactive.Value("")
working_Compatability2_string=reactive.Value("")

NumForcesY=reactive.Value(2)
F1y=reactive.Value("")
F2y=reactive.Value("")
F3y=reactive.Value("")
F4y=reactive.Value("")
F5y=reactive.Value("")
Equil_latex=reactive.Value("")

NumForcesX=reactive.Value(2)
F1x=reactive.Value("")
```

```
F2x=reactive.Value("")
F3x=reactive.Value("")
F4x=reactive.Value("")
F5x=reactive.Value("")

NumMoments=reactive.Value(2)
M1=reactive.Value("")
M2=reactive.Value("")
M3=reactive.Value("")
M4=reactive.Value("")
M5=reactive.Value("")

axial_stress_sigma=reactive.Value("")
axial_stress_force=reactive.Value("")
axial_stress_area=reactive.Value("")

bending_stress_sigma=reactive.Value("")
bending_stress_M=reactive.Value("")
bending_stress_y=reactive.Value("")
bending_stress_I=reactive.Value("")

axial_delta_l=reactive.Value("")
axial_P=reactive.Value("")
axial_L=reactive.Value("")
axial_A=reactive.Value("")
axial_E=reactive.Value("")

thermal_delta_t=reactive.Value("")
thermal_alpha=reactive.Value("")
thermal_Delta_T=reactive.Value("")
thermal_L=reactive.Value("")

area_tube_A_tube=reactive.Value("")
area_tube_Ar_o=reactive.Value("")
area_tube_Ar_i=reactive.Value("")

I_tube_I_tube=reactive.Value("")
I_tube_Ir_o=reactive.Value("")
i_tube_Ir_i=reactive.Value("")

Compatability1_NumLHS=reactive.Value(1)
```

```
Compatability1_NumRHS=reactive.Value(2)
Compatability1_a_1=reactive.Value("")
Compatability1_a_2=reactive.Value("")
Compatability1_a_3=reactive.Value("")
Compatability1_a_4=reactive.Value("")
Compatability1_a_5=reactive.Value("")
Compatability1_b_1=reactive.Value("")
Compatability1_b_2=reactive.Value("")
Compatability1_b_3=reactive.Value("")
Compatability1_b_4=reactive.Value("")
Compatability1_b_5=reactive.Value("")

Compatability2_NumLHS=reactive.Value(1)
Compatability2_NumRHS=reactive.Value(2)
Compatability2_c_1=reactive.Value("")
Compatability2_c_2=reactive.Value("")
Compatability2_c_3=reactive.Value("")
Compatability2_c_4=reactive.Value("")
Compatability2_c_5=reactive.Value("")
Compatability2_d_1=reactive.Value("")
Compatability2_d_2=reactive.Value("")
Compatability2_d_3=reactive.Value("")
Compatability2_d_4=reactive.Value("")
Compatability2_d_5=reactive.Value("")

active_eqn_tab=reactive.Value("Instructions")

prob_statement="To scaffold your learning in this example, we have provided a free body di

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.column(6,
```

```
                ui.markdown("**Problem Statement**"),
                ui.markdown(prob_statement),
                ),
            ui.column(6,ui.output_ui("dyn_ui_nav")),
            ),
        #ui.row(
        #    ui.output_ui("dyn_ui_nav"),
        #    ),
        ui.row(
            ui.markdown("**Your Equation Workspace**"),
            #ui.column(6,ui.output_ui("dyn_ui_nav")),
            ui.column(4,
                ui.navset_tab_card(
                    ui.nav("Equation Bank",
                        ui.input_checkbox_group("selected_eqns","Choose your equations:",e
                        ),
                    ),
                ),
            ),
            ui.column(2,ui.output_ui("dyn_eqns"),style='border-right:1px solid;'),
            ui.column(4,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"),
)


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)
```

```
    feedback_equations.set(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 Force equations
    if StaticsSumFx.newline_math in lookup_eqns:
        StaticsSumFx_list = ["F_x1","F_x2","F_x3","F_x4","F_x5"]
        StaticsSumFx_list = StaticsSumFx_list[:input.NumForcesX()]
        StaticsSumFx.working_sym = ",".join(StaticsSumFx_list)
        StaticsSumFx.working_eqn_latex = "$$" + "+".join(StaticsSumFx_list) + "=0$$"
        StaticsSumFx.working_eqn_solver = "+".join(StaticsSumFx_list)

        if str(input.F1x()) != "" :
            StaticsSumFx.working_eqn_latex = StaticsSumFx.working_eqn_latex.replace("F
            StaticsSumFx.working_sym = StaticsSumFx.working_sym.replace("F_x1",str(inp
            StaticsSumFx.working_eqn_solver = StaticsSumFx.working_eqn_solver.replace(
        else:
            StaticsSumFx.working_eqn_latex = StaticsSumFx.working_eqn_latex.replace("F

        if str(input.F2x()) != "" :
            StaticsSumFx.working_eqn_latex = StaticsSumFx.working_eqn_latex.replace("F
            StaticsSumFx.working_sym = StaticsSumFx.working_sym.replace("F_x2",str(inp
            StaticsSumFx.working_eqn_solver = StaticsSumFx.working_eqn_solver.replace(
        else:
            StaticsSumFx.working_eqn_latex = StaticsSumFx.working_eqn_latex.replace("F
```

```
    if str(input.F3x()) != "" :
        StaticsSumFx.working_eqn_latex = StaticsSumFx.working_eqn_latex.replace("F
        StaticsSumFx.working_sym = StaticsSumFx.working_sym.replace("F_x3",str(inp
        StaticsSumFx.working_eqn_solver = StaticsSumFx.working_eqn_solver.replace(
    else:
        StaticsSumFx.working_eqn_latex = StaticsSumFx.working_eqn_latex.replace("F

    if str(input.F4x()) != "" :
        StaticsSumFx.working_eqn_latex = StaticsSumFx.working_eqn_latex.replace("F
        StaticsSumFx.working_sym = StaticsSumFx.working_sym.replace("F_x4",str(inp
        StaticsSumFx.working_eqn_solver = StaticsSumFx.working_eqn_solver.replace(
    else:
        StaticsSumFx.working_eqn_latex = StaticsSumFx.working_eqn_latex.replace("F

    if str(input.F5x()) != "" :
        StaticsSumFx.working_eqn_latex = StaticsSumFx.working_eqn_latex.replace("F
        StaticsSumFx.working_sym = StaticsSumFx.working_sym.replace("F_x5",str(inp
        StaticsSumFx.working_eqn_solver = StaticsSumFx.working_eqn_solver.replace(
    else:
        StaticsSumFx.working_eqn_latex = StaticsSumFx.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"
            StressEqn.working_eqn_solver = StressEqn.working_eqn_solver.replace("sigma
            StressEqn.working_sym = StressEqn.working_sym.replace("sigma",str(input.si
```

```
        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(
            StressEqn.working_eqn_solver = StressEqn.working_eqn_solver.replace("F",st
            StressEqn.working_sym = StressEqn.working_sym.replace("F",str(input.force(
        else:
            StressEqn.working_eqn_latex = StressEqn.working_eqn_latex.replace("F","\\b
        if str(input.area()) != "" :
            StressEqn.working_eqn_latex = StressEqn.working_eqn_latex.replace("A",str(
            StressEqn.working_eqn_solver = StressEqn.working_eqn_solver.replace("A",st
            StressEqn.working_sym = StressEqn.working_sym.replace("A",str(input.area()
        else:
            StressEqn.working_eqn_latex = StressEqn.working_eqn_latex.replace("A","\\b


    # Dynamic Filling of Bending Stress equation
    if BendingStress.newline_math in lookup_eqns:
        BendingStress.working_eqn_latex = BendingStress.newline_math
        BendingStress.working_eqn_solver = "Eq(sigma_b,M*y/I)"
        BendingStress.working_sym = "sigma_b,M,y,I"
        if str(input.bendingstress_sigma_b()) != "" :
            BendingStress.working_eqn_latex = BendingStress.working_eqn_latex.replace(
            BendingStress.working_eqn_solver = BendingStress.working_eqn_solver.replac
            BendingStress.working_sym = BendingStress.working_sym.replace("sigma_b",st
        else:
            BendingStress.working_eqn_latex = BendingStress.working_eqn_latex.replace(
        if str(input.bendingstress_M()) != "" :
            BendingStress.working_eqn_latex = BendingStress.working_eqn_latex.replace(
            BendingStress.working_eqn_solver = BendingStress.working_eqn_solver.replac
            BendingStress.working_sym = BendingStress.working_sym.replace("M",str(inpu
        else:
            BendingStress.working_eqn_latex = BendingStress.working_eqn_latex.replace(
        if str(input.bendingstress_y()) != "" :
            BendingStress.working_eqn_latex = BendingStress.working_eqn_latex.replace(
            BendingStress.working_eqn_solver = BendingStress.working_eqn_solver.replac
            BendingStress.working_sym = BendingStress.working_sym.replace("y",str(inpu
        else:
            BendingStress.working_eqn_latex = BendingStress.working_eqn_latex.replace(
        if str(input.bendingstress_I()) != "" :
            BendingStress.working_eqn_latex = BendingStress.working_eqn_latex.replace(
```

```
            BendingStress.working_eqn_solver = BendingStress.working_eqn_solver.replac
            BendingStress.working_sym = BendingStress.working_sym.replace("I",str(inpu
        else:
            BendingStress.working_eqn_latex = BendingStress.working_eqn_latex.replace(


    # Dynamic Filling of Axial Deform equation
    if AxialDeform.newline_math in lookup_eqns:
        AxialDeform.working_eqn_latex = AxialDeform.newline_math
        AxialDeform.working_eqn_solver = "Eq(delta_l,P*L/A/E)"
        AxialDeform.working_sym = "delta_l,P,L,A,E"
        if str(input.axial_delta_l()) != "" :
            AxialDeform.working_eqn_latex = AxialDeform.working_eqn_latex.replace("\de
            AxialDeform.working_eqn_solver = AxialDeform.working_eqn_solver.replace("d
            AxialDeform.working_sym = AxialDeform.working_sym.replace("delta_l",str(in
        else:
            AxialDeform.working_eqn_latex = AxialDeform.working_eqn_latex.replace("\de
        if str(input.axial_P()) != "" :
            AxialDeform.working_eqn_latex = AxialDeform.working_eqn_latex.replace("P",
            AxialDeform.working_eqn_solver = AxialDeform.working_eqn_solver.replace("P
            AxialDeform.working_sym = AxialDeform.working_sym.replace("P",str(input.ax
        else:
            AxialDeform.working_eqn_latex = AxialDeform.working_eqn_latex.replace("P",
        if str(input.axial_L()) != "" :
            AxialDeform.working_eqn_latex = AxialDeform.working_eqn_latex.replace("L",
            AxialDeform.working_eqn_solver = AxialDeform.working_eqn_solver.replace("L
            AxialDeform.working_sym = AxialDeform.working_sym.replace("L",str(input.ax
        else:
            AxialDeform.working_eqn_latex = AxialDeform.working_eqn_latex.replace("L",
        if str(input.axial_A()) != "" :
            AxialDeform.working_eqn_latex = AxialDeform.working_eqn_latex.replace("A",
            AxialDeform.working_eqn_solver = AxialDeform.working_eqn_solver.replace("A
            AxialDeform.working_sym = AxialDeform.working_sym.replace("A",str(input.ax
        else:
            AxialDeform.working_eqn_latex = AxialDeform.working_eqn_latex.replace("A",
        if str(input.axial_E()) != "" :
            AxialDeform.working_eqn_latex = AxialDeform.working_eqn_latex.replace("E",
            AxialDeform.working_eqn_solver = AxialDeform.working_eqn_solver.replace("E
            AxialDeform.working_sym = AxialDeform.working_sym.replace("E",str(input.ax
        else:
            AxialDeform.working_eqn_latex = AxialDeform.working_eqn_latex.replace("E",
```

```
# Dynamic Filling of Thermal Deform equation
if ThermalDeform.newline_math in lookup_eqns:
    ThermalDeform.working_eqn_latex = ThermalDeform.newline_math
    ThermalDeform.working_eqn_solver = "Eq(delta_t,alpha*Delta_T*L)"
    ThermalDeform.working_sym = "delta_t,Delta_T,alpha,L"
    if str(input.thermal_delta_t()) != "" :
        ThermalDeform.working_eqn_latex = ThermalDeform.working_eqn_latex.replace(
        ThermalDeform.working_eqn_solver = ThermalDeform.working_eqn_solver.replac
        ThermalDeform.working_sym = ThermalDeform.working_sym.replace("delta_t",st
    else:
        ThermalDeform.working_eqn_latex = ThermalDeform.working_eqn_latex.replace(
    if str(input.thermal_alpha()) != "" :
        ThermalDeform.working_eqn_latex = ThermalDeform.working_eqn_latex.replace(
        ThermalDeform.working_eqn_solver = ThermalDeform.working_eqn_solver.replac
        ThermalDeform.working_sym = ThermalDeform.working_sym.replace("alpha",str(
    else:
        ThermalDeform.working_eqn_latex = ThermalDeform.working_eqn_latex.replace(
    if str(input.thermal_Delta_T()) != "" :
        ThermalDeform.working_eqn_latex = ThermalDeform.working_eqn_latex.replace(
        ThermalDeform.working_eqn_solver = ThermalDeform.working_eqn_solver.replac
        ThermalDeform.working_sym = ThermalDeform.working_sym.replace("Delta_T",st
    else:
        ThermalDeform.working_eqn_latex = ThermalDeform.working_eqn_latex.replace(
    if str(input.thermal_L()) != "" :
        ThermalDeform.working_eqn_latex = ThermalDeform.working_eqn_latex.replace(
        ThermalDeform.working_eqn_solver = ThermalDeform.working_eqn_solver.replac
        ThermalDeform.working_sym = ThermalDeform.working_sym.replace("L",str(inpu
    else:
        ThermalDeform.working_eqn_latex = ThermalDeform.working_eqn_latex.replace(


# Dynamic Filling of Compatability equation 1
if Compatability1.newline_math in lookup_eqns:
    Compatability1_list__LHS = ["a_1","a_2","a_3","a_4","a_5"]
    Compatability1_list__RHS = ["b_1","b_2","b_3","b_4","b_5"]
    Compatability1_list_LHS = Compatability1_list__LHS[:input.Compatability1_NumLH
    Compatability1_list_RHS = Compatability1_list__RHS[:input.Compatability1_NumRH
    Compatability1_list = Compatability1_list_LHS + Compatability1_list_RHS
    Compatability1.working_sym = ",".join(Compatability1_list)
    Compatability1.working_eqn_latex = "$$" + "+".join(Compatability1_list_LHS) +
    Compatability1.working_eqn_solver = "Eq("+ "+".join(Compatability1_list_LHS) +
```

```
if str(input.a_1()) != "" :
    Compatability1.working_eqn_latex = Compatability1.working_eqn_latex.replac
    Compatability1.working_sym = Compatability1.working_sym.replace("a_1",str(
    Compatability1.working_eqn_solver = Compatability1.working_eqn_solver.repl
else:
    Compatability1.working_eqn_latex = Compatability1.working_eqn_latex.replac

if str(input.a_2()) != "" :
    Compatability1.working_eqn_latex = Compatability1.working_eqn_latex.replac
    Compatability1.working_sym = Compatability1.working_sym.replace("a_2",str(
    Compatability1.working_eqn_solver = Compatability1.working_eqn_solver.repl
else:
    Compatability1.working_eqn_latex = Compatability1.working_eqn_latex.replac

if str(input.a_3()) != "" :
    Compatability1.working_eqn_latex = Compatability1.working_eqn_latex.replac
    Compatability1.working_sym = Compatability1.working_sym.replace("a_3",str(
    Compatability1.working_eqn_solver = Compatability1.working_eqn_solver.repl
else:
    Compatability1.working_eqn_latex = Compatability1.working_eqn_latex.replac

if str(input.a_4()) != "" :
    Compatability1.working_eqn_latex = Compatability1.working_eqn_latex.replac
    Compatability1.working_sym = Compatability1.working_sym.replace("a_4",str(
    Compatability1.working_eqn_solver = Compatability1.working_eqn_solver.repl
else:
    Compatability1.working_eqn_latex = Compatability1.working_eqn_latex.replac

if str(input.a_5()) != "" :
    Compatability1.working_eqn_latex = Compatability1.working_eqn_latex.replac
    Compatability1.working_sym = Compatability1.working_sym.replace("a_5",str(
    Compatability1.working_eqn_solver = Compatability1.working_eqn_solver.repl
else:
    Compatability1.working_eqn_latex = Compatability1.working_eqn_latex.replac

if str(input.b_1()) != "" :
    Compatability1.working_eqn_latex = Compatability1.working_eqn_latex.replac
    Compatability1.working_sym = Compatability1.working_sym.replace("b_1",str(
    Compatability1.working_eqn_solver = Compatability1.working_eqn_solver.repl
else:
    Compatability1.working_eqn_latex = Compatability1.working_eqn_latex.replac
```

```
            if str(input.b_2()) != "" :
                Compatability1.working_eqn_latex = Compatability1.working_eqn_latex.replac
                Compatability1.working_sym = Compatability1.working_sym.replace("b_2",str(
                Compatability1.working_eqn_solver = Compatability1.working_eqn_solver.repl
            else:
                Compatability1.working_eqn_latex = Compatability1.working_eqn_latex.replac

            if str(input.b_3()) != "" :
                Compatability1.working_eqn_latex = Compatability1.working_eqn_latex.replac
                Compatability1.working_sym = Compatability1.working_sym.replace("b_3",str(
                Compatability1.working_eqn_solver = Compatability1.working_eqn_solver.repl
            else:
                Compatability1.working_eqn_latex = Compatability1.working_eqn_latex.replac

            if str(input.b_4()) != "" :
                Compatability1.working_eqn_latex = Compatability1.working_eqn_latex.replac
                Compatability1.working_sym = Compatability1.working_sym.replace("b_4",str(
                Compatability1.working_eqn_solver = Compatability1.working_eqn_solver.repl
            else:
                Compatability1.working_eqn_latex = Compatability1.working_eqn_latex.replac

            if str(input.b_5()) != "" :
                Compatability1.working_eqn_latex = Compatability1.working_eqn_latex.replac
                Compatability1.working_sym = Compatability1.working_sym.replace("b_5",str(
                Compatability1.working_eqn_solver = Compatability1.working_eqn_solver.repl
            else:
                Compatability1.working_eqn_latex = Compatability1.working_eqn_latex.replac

    # Dynamic Filling of Compatability equation 2
            if Compatability2.newline_math in lookup_eqns:
                Compatability2_list__LHS = ["c_1","c_2","c_3","c_4","c_5"]
                Compatability2_list__RHS = ["d_1","d_2","d_3","d_4","d_5"]
                Compatability2_list_LHS = Compatability2_list__LHS[:input.Compatability2_NumLH
                Compatability2_list_RHS = Compatability2_list__RHS[:input.Compatability2_NumRH
                Compatability2_list = Compatability2_list_LHS + Compatability2_list_RHS
                Compatability2.working_sym = ",".join(Compatability2_list)
                Compatability2.working_eqn_latex = "$$" + "+".join(Compatability2_list_LHS) +
                Compatability2.working_eqn_solver = "Eq("+ "+".join(Compatability2_list_LHS) +

                if str(input.c_1()) != "" :
                    Compatability2.working_eqn_latex = Compatability2.working_eqn_latex.replac
```

```
        Compatability2.working_sym = Compatability2.working_sym.replace("c_1",str(
        Compatability2.working_eqn_solver = Compatability2.working_eqn_solver.repl
    else:
        Compatability2.working_eqn_latex = Compatability2.working_eqn_latex.replac

    if str(input.c_2()) != "" :
        Compatability2.working_eqn_latex = Compatability2.working_eqn_latex.replac
        Compatability2.working_sym = Compatability2.working_sym.replace("c_2",str(
        Compatability2.working_eqn_solver = Compatability2.working_eqn_solver.repl
    else:
        Compatability2.working_eqn_latex = Compatability2.working_eqn_latex.replac

    if str(input.c_3()) != "" :
        Compatability2.working_eqn_latex = Compatability2.working_eqn_latex.replac
        Compatability2.working_sym = Compatability2.working_sym.replace("c_3",str(
        Compatability2.working_eqn_solver = Compatability2.working_eqn_solver.repl
    else:
        Compatability2.working_eqn_latex = Compatability2.working_eqn_latex.replac

    if str(input.c_4()) != "" :
        Compatability2.working_eqn_latex = Compatability2.working_eqn_latex.replac
        Compatability2.working_sym = Compatability2.working_sym.replace("c_4",str(
        Compatability2.working_eqn_solver = Compatability2.working_eqn_solver.repl
    else:
        Compatability2.working_eqn_latex = Compatability2.working_eqn_latex.replac

    if str(input.c_5()) != "" :
        Compatability2.working_eqn_latex = Compatability2.working_eqn_latex.replac
        Compatability2.working_sym = Compatability2.working_sym.replace("c_5",str(
        Compatability2.working_eqn_solver = Compatability2.working_eqn_solver.repl
    else:
        Compatability2.working_eqn_latex = Compatability2.working_eqn_latex.replac

    if str(input.d_1()) != "" :
        Compatability2.working_eqn_latex = Compatability2.working_eqn_latex.replac
        Compatability2.working_sym = Compatability2.working_sym.replace("d_1",str(
        Compatability2.working_eqn_solver = Compatability2.working_eqn_solver.repl
    else:
        Compatability2.working_eqn_latex = Compatability2.working_eqn_latex.replac

    if str(input.d_2()) != "" :
```

```
            Compatability2.working_eqn_latex = Compatability2.working_eqn_latex.replac
            Compatability2.working_sym = Compatability2.working_sym.replace("d_2",str(
            Compatability2.working_eqn_solver = Compatability2.working_eqn_solver.repl
        else:
            Compatability2.working_eqn_latex = Compatability2.working_eqn_latex.replac

        if str(input.d_3()) != "" :
            Compatability2.working_eqn_latex = Compatability2.working_eqn_latex.replac
            Compatability2.working_sym = Compatability2.working_sym.replace("d_3",str(
            Compatability2.working_eqn_solver = Compatability2.working_eqn_solver.repl
        else:
            Compatability2.working_eqn_latex = Compatability2.working_eqn_latex.replac

        if str(input.d_4()) != "" :
            Compatability2.working_eqn_latex = Compatability2.working_eqn_latex.replac
            Compatability2.working_sym = Compatability2.working_sym.replace("d_4",str(
            Compatability2.working_eqn_solver = Compatability2.working_eqn_solver.repl
        else:
            Compatability2.working_eqn_latex = Compatability2.working_eqn_latex.replac

        if str(input.d_5()) != "" :
            Compatability2.working_eqn_latex = Compatability2.working_eqn_latex.replac
            Compatability2.working_sym = Compatability2.working_sym.replace("d_5",str(
            Compatability2.working_eqn_solver = Compatability2.working_eqn_solver.repl
        else:
            Compatability2.working_eqn_latex = Compatability2.working_eqn_latex.replac


    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,
    BendingStress.name: BendingStress.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,
    Compatability1.name: Compatability1.working_eqn_latex,
    Compatability2.name: Compatability2.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,
BendingStress.name: BendingStress.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,
Compatability1.name: Compatability1.working_eqn_solver,
Compatability2.name: Compatability2.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,
BendingStress.name: BendingStress.working_sym,
AxialDeform.name: AxialDeform.working_sym,
ThermalDeform.name: ThermalDeform.working_sym,
AreaTube.name: AreaTube.working_sym,
ITube.name: ITube.working_sym,
Compatability1.name: Compatability1.working_sym,
Compatability2.name: Compatability2.working_sym
}

working_eqns_latex = [eqnbank_working_latex[key] for key in eqns_keys]
working_SumFx_render.set(eqnbank_working_latex["Equilibrium Forces in X"])
working_SumFy_render.set(eqnbank_working_latex["Equilibrium Forces in Y"])
working_SumM_render.set(eqnbank_working_latex["Equilibrium Moments about O"])
working_StressEqn_render.set(eqnbank_working_latex["Stress Equation"])
working_BendingStress_render.set(eqnbank_working_latex["Bending Stress from a Mome
working_AxialDeform_render.set(eqnbank_working_latex["Axial Deformation by Force"]
working_ThermalDeform_render.set(eqnbank_working_latex["Axial Deformation by Therm
working_AreaTube_render.set(eqnbank_working_latex["Area of a Tube"])
working_Itube_render.set(eqnbank_working_latex["Moment of Inertia of a Tube"])
working_Compatability1_render.set(eqnbank_working_latex["Compatability Equation 1"
working_Compatability2_render.set(eqnbank_working_latex["Compatability Equation 2"
working_eqns_solver = [eqnbank_working_solver[key] for key in eqns_keys]
```

```
temp_working_equations_solver = "#".join(working_eqns_solver)
temp_working_equations_solver = temp_working_equations_solver.replace("Eq","Wrap_c
temp_working_equations_solver = temp_working_equations_solver.replace("E","E_clash
temp_working_equations_solver = temp_working_equations_solver.replace("I","I_clash
temp_working_equations_solver = temp_working_equations_solver.replace("N","N_clash
temp_working_equations_solver = temp_working_equations_solver.replace("Wrap_clash"
working_eqns_solver = temp_working_equations_solver.split("#")

#working_eqns_solver=[]
#for j in working_eqns_solver_pre:
#    temp=j.split(",")
#    temp2=temp.replace("I","I_clash")
#    temp3=temp2.replace("E","E_clash")
#    working_eqns_solver.append(temp3)

working_syms = [symbank_working[key] for key in eqns_keys]
mystring_working_eqns = "".join(working_eqns_latex)
mystring_working_eqns = mystring_working_eqns.replace("*","\\times")
feedback_syms.set(working_syms)
working_equations_solver.set(working_eqns_solver)

working_syms_only=[]
for j in working_syms:
    temp=j.split(",")
    for k in temp:
        try:
            float(eval(k))
        except:
            temp2=k.replace("I","I_clash")
            temp3=temp2.replace("E","E_clash")
            temp4=temp3.replace("N","N_clash")
            working_syms_only.append(temp4)
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():

    tab_Instructions = ui.nav(
        "Instructions",ui.markdown("Please choose the equations you would like to use
    )

    tab_StaticsSumFy = ui.nav(
            str(StaticsSumFy.inline_math),
            #ui.markdown(working_SumFy_string()),
            ui.input_numeric("NumForcesY","How many terms do you want?",value=NumForce
            ui.input_text("F1y","\(F_{y_1}=\)", value=F1y(),placeholder="Please type i
            ui.input_text("F2y","\(F_{y_2}=\)", value=F2y(),placeholder="Please type i
            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}
            )

    tab_StaticsSumFx = ui.nav(
            str(StaticsSumFx.inline_math),
            #ui.markdown(working_SumFx_string()),
            ui.input_numeric("NumForcesX","How many terms do you want?",value=NumForce
            ui.input_text("F1x","\(F_{x_1}=\)", value=F1x(),placeholder="Please type i
            ui.input_text("F2x","\(F_{x_2}=\)", value=F2x(),placeholder="Please type i
            ui.panel_conditional("input.NumForcesX>=3", ui.input_text("F3x","\(F_{x_3}
            ui.panel_conditional("input.NumForcesX>=4", ui.input_text("F4x","\(F_{x_4}
            ui.panel_conditional("input.NumForcesX>=5", ui.input_text("F5x","\(F_{x_5}
            )

    tab_StaticsSumM = ui.nav(
            str(StaticsSumM.inline_math),
            #ui.markdown(working_SumM_string()),
            ui.input_numeric("NumMoments","How many terms do you want?",value=NumMomen
            ui.input_text("M1","\(M_1=\)", value=M1(),placeholder="Please type in vari
            ui.input_text("M2","\(M_2=\)", value=M2(),placeholder="Please type in vari
            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","\(M_5=\)",
            )
```

```
tab_StressEqn = ui.nav(
        str(StressEqn.inline_math),
        #ui.markdown(working_StressEqn_string()),
        ui.input_text("sigma","\(\sigma\)", value=axial_stress_sigma(),placeholder
        ui.input_text("force","\(F\)", value=axial_stress_force(),placeholder="Ple
        ui.input_text("area","\(A\)", value=axial_stress_area(),placeholder="Pleas
        )

tab_BendingStress = ui.nav(
        str(BendingStress.inline_math),
        #ui.markdown(working_BendingStress_string()),
        ui.input_text("bendingstress_sigma_b","\(\sigma_b\)", value=bending_stress
        ui.input_text("bendingstress_M","\(M\)", value=bending_stress_M(), placeho
        ui.input_text("bendingstress_y","\(y\)", value=bending_stress_y(), placeho
        ui.input_text("bendingstress_I","\(I\)", value= bending_stress_I(), placeh
        )

tab_AxialDeform = ui.nav(
        str(AxialDeform.inline_math),
        #ui.markdown(working_AxialDeform_string()),
        ui.input_text("axial_delta_l","\(\delta_l\)", value=axial_delta_l(), place
        ui.input_text("axial_P","\(P\)", value=axial_P(), placeholder="Please type
        ui.input_text("axial_L","\(L\)", value=axial_L(), placeholder="Please type
        ui.input_text("axial_A","\(A\)", value=axial_A(), placeholder="Please type
        ui.input_text("axial_E","\(E\)", value=axial_E(), placeholder="Please type
        )

tab_ThermalDeform = ui.nav(
        str(ThermalDeform.inline_math),
        #ui.markdown(working_ThermalDeform_string()),
        ui.input_text("thermal_delta_t","\(\delta_t\)", value=thermal_delta_t(), p
        ui.input_text("thermal_alpha","\(\\alpha\)", value=thermal_alpha(), placeh
        ui.input_text("thermal_Delta_T","\(\Delta T\)", value=thermal_Delta_T(), p
        ui.input_text("thermal_L","\(L\)", value=thermal_L(), placeholder="Please
        )

tab_AreaTube = ui.nav(
        str(AreaTube.inline_math),
        #ui.markdown(working_AreaTube_string()),
        ui.input_text("A_tube","\(A_{tube}=\)", value=area_tube_A_tube(), placehol
        ui.input_text("Ar_o","\(r_o=\)", value=area_tube_Ar_o(), placeholder="Plea
```

```
        ui.input_text("Ar_i","\(r_i\)", value=area_tube_Ar_i(), placeholder="Pleas
        )

    tab_ITube = ui.nav(
        str(ITube.inline_math),
        #ui.markdown(working_Itube_string()),
        ui.input_text("I_tube","\(I_{tube}=\)", value=I_tube_I_tube(), placeholder
        ui.input_text("Ir_o","\(r_o=\)", value=I_tube_Ir_o(), placeholder="Please
        ui.input_text("Ir_i","\(r_i\)", value=i_tube_Ir_i(), placeholder="Please t
        )

    tab_Compatability1 = ui.nav(
        str(Compatability1.inline_math),
        #ui.markdown(working_Compatability1_string()),
        ui.input_numeric("Compatability1_NumLHS","How many 'a' terms do you want?"
        ui.input_numeric("Compatability1_NumRHS","How many 'b' terms do you want?"
        ui.input_text("a_1","\(a_1=\)", value=Compatability1_a_1(),placeholder="Pl
        ui.panel_conditional("input.Compatability1_NumLHS>=2",ui.input_text("a_2",
        ui.panel_conditional("input.Compatability1_NumLHS>=3", ui.input_text("a_3"
        ui.panel_conditional("input.Compatability1_NumLHS>=4", ui.input_text("a_4"
        ui.panel_conditional("input.Compatability1_NumLHS>=5", ui.input_text("a_5"
        ui.input_text("b_1","\(b_1=\)", value=Compatability1_b_1(),placeholder="Pl
        ui.panel_conditional("input.Compatability1_NumRHS>=2",ui.input_text("b_2",
        ui.panel_conditional("input.Compatability1_NumRHS>=3",ui.input_text("b_3",
        ui.panel_conditional("input.Compatability1_NumRHS>=4",ui.input_text("b_4",
        ui.panel_conditional("input.Compatability1_NumRHS>=5",ui.input_text("b_5",
        )

    tab_Compatability2 = ui.nav(
        str(Compatability2.inline_math),
        #ui.markdown(working_Compatability2_string()),
        ui.input_numeric("Compatability2_NumLHS","How many 'c' terms do you want?"
        ui.input_numeric("Compatability2_NumRHS","How many 'd' terms do you want?"
        ui.input_text("c_1","\(c_1=\)", value=Compatability2_c_1(),placeholder="Pl
        ui.panel_conditional("input.Compatability2_NumLHS>=2",ui.input_text("c_2",
        ui.panel_conditional("input.Compatability2_NumLHS>=3", ui.input_text("c_3"
        ui.panel_conditional("input.Compatability2_NumLHS>=4", ui.input_text("c_4"
        ui.panel_conditional("input.Compatability2_NumLHS>=5", ui.input_text("c_5"
        ui.input_text("d_1","\(d_1=\)", value=Compatability2_d_1(),placeholder="Pl
        ui.panel_conditional("input.Compatability2_NumRHS>=2",ui.input_text("d_2",
        ui.panel_conditional("input.Compatability2_NumRHS>=3",ui.input_text("d_3",
```

```
            ui.panel_conditional("input.Compatability2_NumRHS>=4",ui.input_text("d_4",
            ui.panel_conditional("input.Compatability2_NumRHS>=5",ui.input_text("d_5",
            )

    tab_bank = {
            StaticsSumFx.name: tab_StaticsSumFx,
            StaticsSumFy.name: tab_StaticsSumFy,
            StaticsSumM.name: tab_StaticsSumM,
            StressEqn.name: tab_StressEqn,
            BendingStress.name: tab_BendingStress,
            AxialDeform.name: tab_AxialDeform,
            ThermalDeform.name: tab_ThermalDeform,
            AreaTube.name: tab_AreaTube,
            ITube.name: tab_ITube,
            Compatability1.name: tab_Compatability1,
            Compatability2.name: tab_Compatability2,
            }
    eqns_keys = input.selected_eqns()
    tabs = [tab_bank[key] for key in eqns_keys]
    tabs.insert(0,tab_Instructions)
    equations = ui.navset_tab_card(*tabs,id="mytab",selected=active_eqn_tab())

    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())+"\\\\)"
    string_working_symbols=string_working_symbols.replace("N_clash","N")
    string_working_symbols=string_working_symbols.replace("I_clash","I")
    string_working_symbols=string_working_symbols.replace("E_clash","E")
    string_working_symbols=string_working_symbols.replace("delta","\delta")
    string_working_symbols=string_working_symbols.replace("Delta","\Delta")
    string_working_symbols=string_working_symbols.replace("sigma","\sigma")
    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="$$"+k+"="+f'{temp:.2f}'+"$$"
            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")
    mystring_answers=mystring_answers.replace("E_clash","E")
    mystring_answers=mystring_answers.replace("I_clash","I")
    mystring_answers=mystring_answers.replace("N_clash","N")
    #feedback_solns.set(mystring_answers)
    return [ui.markdown(f"Your solution is {mystring_answers}"),
            ui.input_text("answer","Answer:",placeholder="Please type in your answer")
            ui.input_action_button("feedback", "Check answer and show feedback"),
            ui.download_button("download", "Download File to Submit", class_="btn-succ
            ui.tags.script(
            "if (window.MathJax) MathJax.Hub.Queue(['Typeset', MathJax.Hub]);"
```

```
            )]

#@reactive.Effect
#def _():

 #   active_eqn_tab.set(input.mytab())


@reactive.Effect
def _():

    input.selected_eqns()
    active_eqn_tab.set(input.mytab())


    #working_SumFx_render():
    #working_SumFy_render()
    #working_SumM_render()
    #working_StressEqn_render()
    #working_BendingStress_render()
    #working_AxialDeform_render()
    #working_ThermalDeform_render()
    #working_AreaTube_render()
    #working_Itube_render()
    #working_Compatability1_render()
    #working_Compatability2_render()

    with reactive.isolate():
        if "Equilibrium Forces in Y" in input.selected_eqns():
            NumForcesY.set(input.NumForcesY())
            F1y.set(input.F1y())
            F2y.set(input.F2y())
            F3y.set(input.F3y())
            F4y.set(input.F4y())
            F5y.set(input.F5y())
            working_SumFy_string.set(str(working_SumFy_render()))
        else:
             pass

        if "Equilibrium Forces in X" in input.selected_eqns():
            NumForcesX.set(input.NumForcesX())
```

```
            F1x.set(input.F1x())
            F2x.set(input.F2x())
            F3x.set(input.F3x())
            F4x.set(input.F4x())
            F5x.set(input.F5x())
            working_SumFx_string.set(str(working_SumFx_render()))
        else:
             pass

        if "Equilibrium Moments about O" in input.selected_eqns():
            NumMoments.set(input.NumMoments())
            M1.set(input.M1())
            M2.set(input.M2())
            M3.set(input.M3())
            M4.set(input.M4())
            M5.set(input.M5())
            working_SumM_string.set(str(working_SumM_render()))
        else:
             pass

        if "Stress Equation" in input.selected_eqns():
            axial_stress_sigma.set(input.sigma())
            axial_stress_force.set(input.force())
            axial_stress_area.set(input.area())
            working_StressEqn_string.set(str(working_StressEqn_render()))
        else:
             pass

        if "Axial Deformation by Force" in input.selected_eqns():
            axial_delta_l.set(input.axial_delta_l())
            axial_P.set(input.axial_P())
            axial_L.set(input.axial_L())
            axial_A.set(input.axial_A())
            axial_E.set(input.axial_E())
            working_AxialDeform_string.set(str(working_AxialDeform_render()))
        else:
             pass

        if "Axial Deformation by Thermal" in input.selected_eqns():
            thermal_delta_t.set(input.thermal_delta_t())
            thermal_alpha.set(input.thermal_alpha())
```

```
        thermal_Delta_T.set(input.thermal_Delta_T())
        thermal_L.set(input.thermal_L())
        working_ThermalDeform_string.set(str(working_ThermalDeform_render()))
    else:
         pass

    if "Area of a Tube" in input.selected_eqns():
        area_tube_A_tube.set(input.A_tube())
        area_tube_Ar_o.set(input.Ar_o())
        area_tube_Ar_i.set(input.Ar_i())
        working_AreaTube_string.set(str(working_AreaTube_render()))
    else:
         pass

    if "Moment of Inertia of a Tube" in input.selected_eqns():
        I_tube_I_tube.set(input.I_tube())
        I_tube_Ir_o.set(input.Ir_o())
        i_tube_Ir_i.set(input.Ir_i())
        working_Itube_string.set(str(working_Itube_render()))
    else:
         pass

    if "Bending Stress from a Moment" in input.selected_eqns():
        bending_stress_sigma.set(input.bendingstress_sigma_b())
        bending_stress_M.set(input.bendingstress_M())
        bending_stress_y.set(input.bendingstress_y())
        bending_stress_I.set(input.bendingstress_I())
        working_BendingStress_string.set(str(working_BendingStress_render()))
    else:
         pass

    if "Compatability Equation 1" in input.selected_eqns():
        Compatability1_NumLHS.set(input.Compatability1_NumLHS())
        Compatability1_NumRHS.set(input.Compatability1_NumRHS())
        Compatability1_a_1.set(input.a_1())
        Compatability1_a_2.set(input.a_2())
        Compatability1_a_3.set(input.a_3())
        Compatability1_a_4.set(input.a_4())
        Compatability1_a_5.set(input.a_5())
        Compatability1_b_1.set(input.b_1())
        Compatability1_b_2.set(input.b_2())
```

```
            Compatability1_b_3.set(input.b_3())
            Compatability1_b_4.set(input.b_4())
            Compatability1_b_5.set(input.b_5())
            working_Compatability1_string.set(str(working_Compatability1_render()))
        else:
             pass

        if "Compatability Equation 2" in input.selected_eqns():
            Compatability2_NumLHS.set(input.Compatability2_NumLHS())
            Compatability2_NumRHS.set(input.Compatability2_NumRHS())
            Compatability2_c_1.set(input.c_1())
            Compatability2_c_2.set(input.c_2())
            Compatability2_c_3.set(input.c_3())
            Compatability2_c_4.set(input.c_4())
            Compatability2_c_5.set(input.c_5())
            Compatability2_d_1.set(input.d_1())
            Compatability2_d_2.set(input.d_2())
            Compatability2_d_3.set(input.d_3())
            Compatability2_d_4.set(input.d_4())
            Compatability2_d_5.set(input.d_5())
            working_Compatability2_string.set(str(working_Compatability2_render()))
        else:
             pass


@reactive.Effect
@reactive.event(input.feedback)
def _():
    inst_eqns=[eqnbank_newline[key] for key in ["Equilibrium Forces in Y", "Equilibriu
    inst_soln="6520"
    inst_unknowns=["N","M_o","\sigma_b", "\sigma_l","A_{tube}","I_{tube}","\sigma_{max
    attempt_equations=feedback_equations()
    attempt_soln=input.answer()
    attempt_unknowns=feedback_syms()
    missing_inst_eqns=set(inst_eqns).difference(attempt_equations)
    extra_student_eqns=set(attempt_equations).difference(inst_eqns)

    if inst_soln==attempt_soln:
      feedback_message=ui.markdown("Congratulations! You are correct, great work.")
    else:
      feedback_message=ui.markdown(f"This feedback is the list method -- checking your
```

```
        m = ui.modal(
            feedback_message, ui.tags.script(
                "if (window.MathJax) MathJax.Hub.Queue(['Typeset', MathJax.Hub]);"
                ),
            title="Feedback on your solution",
            easy_close=True,
            footer=None,
        )
        ui.modal_show(m)


    @session.download(
    filename=lambda: f"Problem_Log.csv"
    )
    async def download():
        # This version uses a function to generate the filename. It also yields data
        # multiple times.
          await asyncio.sleep(0.25)
          yield f"{working_equations_solver()}\n"
          yield f"{working_equations_latex_render()}\n"

app = App(app_ui, server)
```

Part II

Problem 391

Dynamic Problem Statement

This is a dynamic rendering of the problem with dynamic variables based on the username entered.

Problem Image

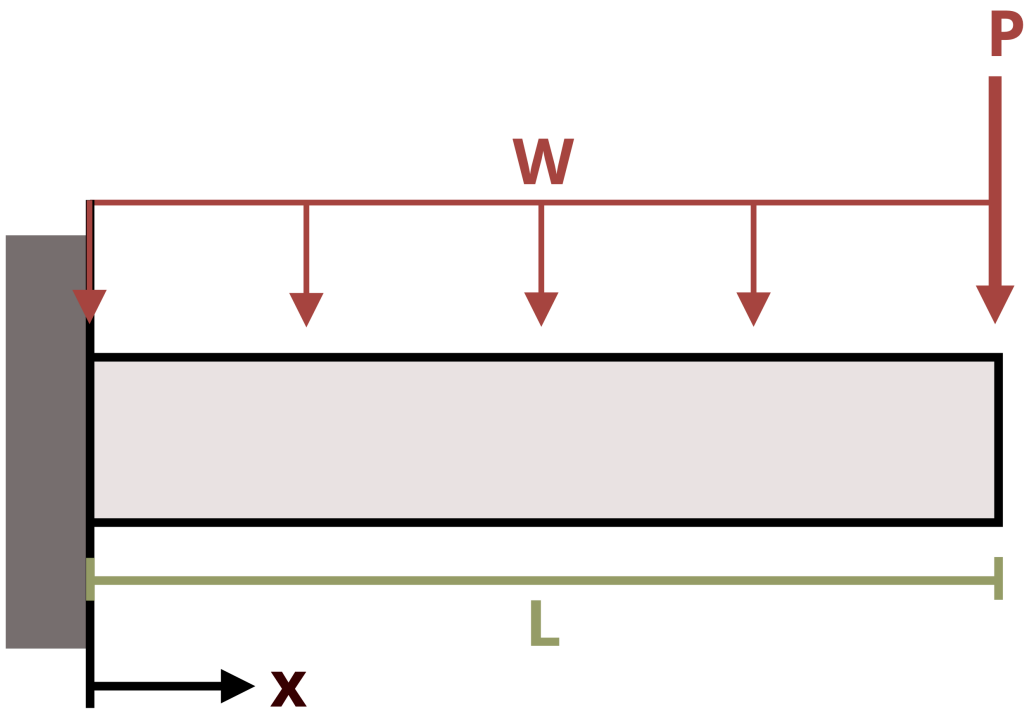

Figure 5: Figure 1: Cantilever beam of length L with distributed load w and point load P

```
#| standalone: true
#| viewerHeight: 600
#| components: [viewer]

from shiny import App, render, ui, reactive
import random
import asyncio
import io
import math
from datetime import datetime
from pathlib import Path

problem_ID="391"
L=reactive.Value("__")
w=reactive.Value("__")
P=reactive.Value("__")
x=reactive.Value("__")
E=1950000
I=21.33
attempts=["Timestamp,Attempt,Answer,Feedback\n"]

app_ui = ui.page_fluid(
    ui.markdown("**Please enter your ID number from your instructor and click to generate
    ui.input_text("ID","", placeholder="Enter ID Number Here"),
    ui.input_action_button("generate_problem", "Generate Problem", class_="btn-primary"),
    ui.markdown("**Problem Statement**"),
    ui.output_ui("ui_problem_statement"),
    ui.input_text("answer","Your Answer in units of inches", placeholder="Please enter you
    ui.input_action_button("submit", "Submit Answer", class_="btn-primary"),
    ui.download_button("download", "Download File to Submit", class_="btn-success"),
)


def server(input, output, session):
    @output
    @render.ui
    def ui_problem_statement():
        return[ui.markdown(f"A cantilever beam of length {L()} ft is loaded with a distrib

    @reactive.Effect
    @reactive.event(input.generate_problem)
```

```
    def randomize_vars():
        random.seed(input.ID())
        L.set(round(8+8*.5*(.5-random.random())))
        w.set(round(2+2*.5*(.5-random.random())))
        P.set(round(10+10*.5*(.5-random.random())))
        x.set(.9*L())

    @reactive.Effect
    @reactive.event(input.submit)
    def _():
        instr= -w()/12*x()*12*x()*12*(6*L()*12*L()*12-4*L()*12*x()*12+x()*12*x()*12)/24/E/
        #check=math.isclose(float(input.answer()),instr,rel_tol=0.001)
        if math.isclose(float(input.answer()),instr,rel_tol=0.001):
           check="*Correct*"
        else:
           check="*Not Correct.*"

        feedback=ui.markdown(f"Your answer of {input.answer()} is {check}.  For reference
        attempts.append(f"{datetime.now()}, {input.submit()},{input.answer()},{check}\n")
        m=ui.modal(
          feedback,
          title="Feedback",
          easy_close=True
        )
        ui.modal_show(m)

    @session.download(
        filename=lambda: f"Problem_Log-{problem_ID}-{input.ID()}.csv"
    )
    async def download():
        # This version uses a function to generate the filename. It also yields data
        # multiple times.
        await asyncio.sleep(0.25)
        yield f"{problem_ID}_{input.submit()}_{input.ID()}\n"
        yield ''.join(attempts)


app = App(app_ui, server)
```

1 Summary

In summary, this book has no content whatsoever.

References