

Software Integrated Instruction of Solid Mechanics

Zubaer Hossain

*Department of Mechanical Engineering
Texas A&M University
College Station, USA
zubaer@tamu.edu*

Shadi Balawi

*Department of Mechanical Engineering
Texas A&M University
College Station, USA
shadi.balawi@tamu.edu*

Abstract—This paper presents the role of JupyterLab, a web-based, free Python coding platform, in solving and analyzing complicated problems of solid mechanics in real-time during classroom lectures. The objectives of using JupyterLab are five-fold: (a) to address the difficulty in explaining the interconnection of multiple variables that affect the solution to solid mechanics problems from a holistic standpoint, (b) to discuss the core concepts of solid mechanics using a set of generalized problems and their solutions with easily modifiable input parameters, (c) to help students attain the ability to critically think about the course concepts and apply them to analyze complex problems, (d) to test students' ability to apply the concepts covered in class to problems that are outside the scope of the course topics, and (e) to promote awareness of outlier situations wherein traditionally taught tools are inapplicable. The ultimate goal is to amplify the learning outcomes of the course and actively engage students in the learning process, helping them prepare for the growing AI-inclined job market in the engineering sector.

Index Terms—JupyterLab, ChatGPT, Solid Mechanics, Machine Design

I. INTRODUCTION

JupyterLab, an advanced web-based interactive development environment for notebooks, code, and data, has emerged as a powerful tool in the field of solid mechanics. Solid mechanics, which deals with the behavior of solid materials under various forces and conditions, traditionally relies on theoretical frameworks and numerical methods for analysis. The integration of JupyterLab into this domain marks a significant advancement, offering an intuitive and dynamic platform for solving and analyzing complex solid mechanics problems during class lectures. One of the primary advantages of using JupyterLab in solid mechanics is its ability to simplify the explanation of complex interdependencies among multiple variables. Traditional methods often struggle to convey the holistic nature of solid mechanics problems, where factors such as material properties, load conditions, and geometric configurations interplay in intricate ways. JupyterLab's interactive notebooks allow users to manipulate these variables dynamically and observe the resulting effects in real-time, fostering a more intuitive and comprehensive understanding during lectures.

Furthermore, in the era of artificial intelligence (AI) [1], [2], it is important to expose students to AI-related tools and approaches to develop AI literacy and essential skills to solve problems using available resources effectively [3]. This effort requires finding innovative ways to instruct and guide students

effectively by minimizing the time required to learn the material and teaching them how to learn. While there is a debate on how 'artificial intelligence' could be used effectively in preparing future workforce without compromising the essence of learning, it can be argued that, if used correctly, it can significantly impact learning complex subjects, such as solid mechanics, that heavily depend on interrelated equations and graphical visualization of their meaning.

For example, consider designing a stepped shaft for application in a transmission system, as depicted in Fig.1. Successful

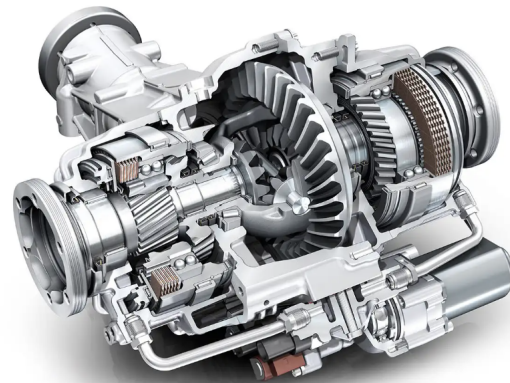


Fig. 1. An example application of a stepped shaft is in a transmission system consisting of many machine elements, including gears, a stepped shaft, and bearings. Developing holistic design skills to design different components of the system and analyze the performance of the whole system requires theoretical calculations and their integration to assess the role of individual components in the overall performance of the system. This task is formidable if done theoretically. The application of software is critical for this, as is routinely done in industry. The question is: Can we use software tools to teach students the fundamental solid mechanics theories and their applications at the component level so that students are familiar with the process and have the ability to apply advanced AI-based tools for engineering tasks?

fatigue design of the shaft involves a clear examination of the load diagrams, stress concentration factors, stress fluctuations, endurance limit, materials strength, and so on. In traditional mechanics courses, the problems students solve primarily have one dependent variable, such as the factor of safety (FOS), for a number of given dependent and independent variables (such as the loading condition and geometric parameters). Students use a series of equations to arrive at the final answer. In the process, they often miss seeing the role of one parameter in the

overall design of the system from a holistic viewpoint. How can someone see what percentage change in one geometric parameter is needed to alter the design parameter by a certain percentage?

In situations where an independent variable is unknown (such as the diameter of a stepped shaft), students are taught to follow an iterative process, which is very important but tedious. Students often do not grasp the key variables that affect the overall design. Instead of offering a black-box platform (such as FEA software packages), where students may not understand the steps followed to find a solution from a given set of inputs, a coding platform with an easily understandable conceptual sequence can play a significant role in giving students a holistic design perspective. Modification of one independent parameter can show its impact on a series of interdependent variables that together form the basis for the overall design. The use and application of free yet powerful web-based platforms (such as JupyterLab [4]) can help students solve complex problems in a modular way and can give them insights into the underlying steps. Students get a broad and comprehensive view of the solution process and investigate how a small part of the solution sequence can bring a dramatic effect on the ultimate solution. In this paper, we show how the platform can be integrated (with ChatGPT and class lectures) to advance mechanics instruction.

II. SOFTWARE INTEGRATED INSTRUCTION (SII) APPROACH

The objectives of using SII is to make it easy for students to see the inner workings and interconnection of different interdependent concepts and steps in the design process. In this approach, there are four basic steps:

- Step 1: **Choose a concept** to teach, with an outcome-oriented mindset and a focus on what the students should be 'able to do' with the concept (e.g. how to design diameters of a stepped shaft?)
- Step 2: **Introduce the concept using a software package** and solve a simple example problem to discuss different parts of the solution process and the meaning of solutions (e.g. show the steps to determine the FOS of a shaft under fatigue loading and discuss what parameter affect the FOS by what percentage)
- Step 3: **Engage students in the learning process** by asking them find solutions to an inverse problem wherein an independent variable needs to be determined to achieve a desired value of the dependent variables (e.g. what diameter is needed to achieve a desired fatigue FOS)
- Step 4: **Assess learning outcome** by asking them to evaluate existing solutions, correcting mistakes in solutions, and enhance the scope of the solutions (e.g. asking them to evaluate correctness of ChatGPT solutions and to find necessary modifications needed to address any limitations).

The approach is focused on enhancing students' ability to analyze solutions and correct errors, both of which require a deeper level of understanding than just doing homework

problems. It is built on the idea that a student who is able to identify mistakes in a solution understands the concepts of a problem well. Exposing students to search for solutions, finding mistakes in solutions, correcting them, and verifying the solution forms the foundation of the SII approach. The software tool acts as a tool in accomplishing the objectives of the approach. In order to demonstrate the approach, in the following section, we focus on solving a problem that is tedious to solve by hand. We show an integration of ChatGPT and JupyterLab to initialize the solution to a problem, complete the solution, generalize the solution, and create a platform to comprehensively analyze the solutions.

III. APPLICATION OF THE SII APPROACH

Consider a stepped shaft that forms the foundation of many engineering applications (ranging from the wind turbine to the automobile transmission system to industrial machinery). We seek to design the diameters of the shaft for a given level of factor of safety (FOS) and material, and we want to understand how changing diameter affects different parts of the calculation/design process. A relevant problem is shown in Fig.2.

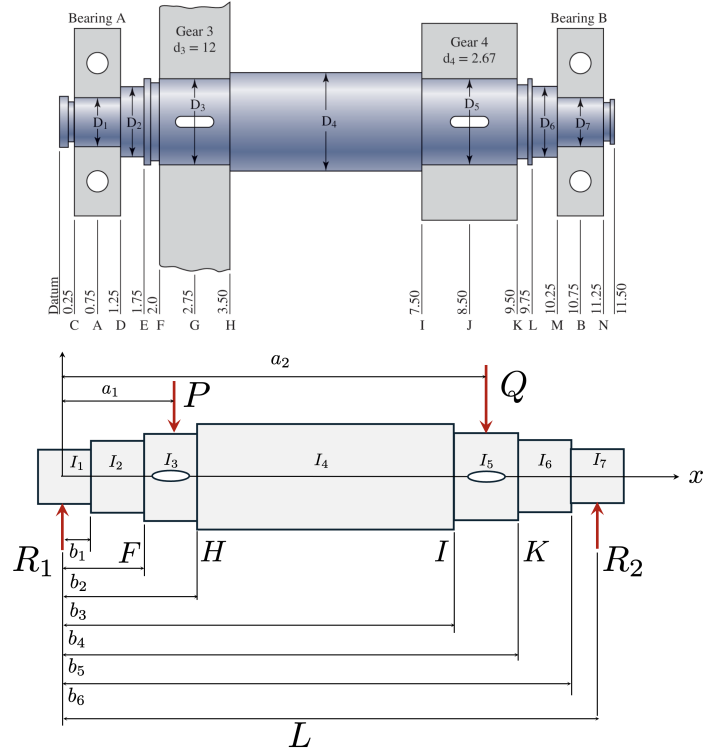


Fig. 2. (Top) A 1020 cold-drawn steel counter-shaft of ultimate strength 68 kpsi and yield strength 57 kpsi carrying two spur gears supported by two bearings at ends. The tangential and radial forces transmitted through the gears to the shaft are $W_{23}^t = 540$ lbf, $W_{23}^r = 197$ lbf, $W_{45}^t = 2431$ lbf, and $W_{45}^r = 885$ lbf. Determine the diameters of each section of the shaft for infinite life with minimum safety factors of 1.5. The problem is taken from Shigley's Mechanical Engineering Design book [5]. (Bottom) Free body diagram representing the generic loading situation in two different planes.

A tentative process for solving the problem involves the following steps:

- Calculate the axial force, bending moment, and torsional moment equations.
- Identify the critical section(s) where the combined effects of forces and moments are the maximum.
- Determine the characteristics of stress fluctuation, representing the mean stress and amplitude of dynamic stress.
- Determine the endurance strength of the solid member at critical section(s).
- Employ a failure criterion to determine the design parameter for a desired FOS.

These steps are explicitly dependent on the diameter. An initial value is thus approximated, and the steps are repeated until a reasonable value of the diameter is found to satisfy the design requirements. The mathematical equations involved in the solution process are as follows:

$$M = R_1 \langle x \rangle^1 - P \langle x - a_1 \rangle^1 - Q \langle x - a_2 \rangle^1 \quad (1)$$

$$R_1 = \frac{P(L - a_1)}{L} + \frac{Q(L - a_2)}{L} \quad (2)$$

$$\sigma'_a = \sqrt{\sigma_a^2 + 3\tau_a^2} = \sqrt{\left(K_{fb} \frac{32M}{\pi d^3} + 3(0)^2\right)} \quad (3)$$

$$\sigma'_m = \sqrt{\sigma_m^2 + 3\tau_m^2} = \sqrt{\left(0 + 3\left(K_{fs} \frac{16T}{\pi d^3}\right)^2\right)} \quad (4)$$

$$K_{fb} = 1 + \frac{K_t - 1}{1 + \sqrt{a/r}} \quad (5)$$

$$K_{fs} = 1 + \frac{K_{ts} - 1}{1 + \sqrt{a/r}} \quad (6)$$

$$S_e = k_a k_b k_c k_d k_e S'_e \quad (7)$$

$$= (aS'_{ut}) (\alpha d^{-\beta}) (1)(1)(1)(0.5S_{ut}) \quad (8)$$

$$n_f = \left(\frac{\sigma'_a}{S_e} + \frac{\sigma'_m}{S_{ut}} \right)^{-1} \quad (9)$$

where, M is the equation of the bending moment (using singularity function approach), σ'_a is the amplitude of the effective alternating stress, σ'_m is the mean value of the effective alternating stress, R_1 is support reaction at left bearing, P and Q are applied concentrated loads at $x = a_1$ and $x = a_2$ respectively, K_{fb} and K_{fs} are the fatigue bending stress and torsional stress concentration factors respectively, a and b depend on surface type, α and β depends on d , and the static stress concentration factors K_t and K_{ts} depend on d and radius of curvature r at critical locations. Factors K_t and K_{ts} are obtained from available experimental data [6].

Note that in solving d for a desired n_f , an iterative process can be followed in which an initial value of d is assumed, n_f is calculated and compared to the design FOS, and d is updated until a value of n_f is achieved with an acceptable tolerance of the design FOS. The diameter of the neighboring shaft segments is determined from conventional diameter ratios (representing the conditions for K_t and K_{ts} consistently). If not, adjust the values and redetermine d following the process elaborated above.

Generate an initial code using ChatGPT. We first ask ChatGPT to ‘Write a JupyterLab notebook to design the diameters of a stepped shaft for fatigue.’ This step gives us an idea of the basic code structure for solving the problem. ChatGPT generates the following:

```
# Import necessary libraries
import numpy as np
import matplotlib.pyplot as plt

# Define the parameters for the stepped shaft
# For simplicity, we will use arbitrary units and simple geometry

# Define the lengths of the sections of the shaft
L1 = 100 # Length of the first section
L2 = 150 # Length of the second section
L3 = 100 # Length of the third section

# Define the applied loads
M_max = 500 # Maximum bending moment (N·m)
T_max = 200 # Maximum torque (N·m)

# Material properties
sigma_e = 250 # Endurance limit (MPa)
S_ut = 600 # Ultimate tensile strength (MPa)
S_y = 450 # Yield strength (MPa)

# Fatigue stress concentration factors (approximate values)
Kt_bending = 1.5; Kt_torsion = 1.2

# Safety factors
n_bending = 1.5; n_torsion = 1.5

# Define the function to calculate the diameter
def calculate_diameter(M, T, Kt_b, Kt_t, S_e, S_ut, S_y, n_b, n_t):
    # Equivalent alternating stress (Modified Goodman criterion)
    sigma_a = Kt_b * M / (np.pi / 32 * d**3)
    tau_a = Kt_t * T / (np.pi / 16 * d**3)
    sigma_eq = np.sqrt(sigma_a**2 + 3 * tau_a**2)

    # Endurance limit (corrected for alternating stress)
    sigma_a_corrected = S_e / n_b

    # Mean stress (assume zero mean stress for simplicity)
    sigma_m = 0

    # Modified Goodman criterion
    sigma_criterion = sigma_eq / (S_e / n_b) + sigma_m / S_ut
    return sigma_criterion

# Find the diameter that satisfies the fatigue criterion
d_initial = 10 # Initial guess for the diameter
tolerance = 0.01 # Convergence tolerance
d = d_initial

while True:
    criterion_value = calculate_diameter(M_max, T_max, Kt_bending, \
                                         Kt_torsion, sigma_e, S_ut, S_y, n_bending, n_torsion)
    if criterion_value < 1:
        break
    d += tolerance

print(f"Calculated diameter for the stepped shaft: {d:.2f} mm")
```

Calculated diameter for the stepped shaft: 10.00 mm

ChatGPT provides a solution for the diameter of one section of the stepped shaft and shows a route to solving the overall problem. It assumes values for important calculations (such as fatigue concentration factors), ignoring the impact on the diameter ratio in the design. Needless to say, in a typical solid mechanics class, we spend multiple lectures teaching how to calculate K_{fb} and K_{fs} . Secondly, ChatGPT starts the calculation with assumed values of M_{\max} and T_{\max} , which must be determined for the given problem at hand. For someone unaware of the theoretical steps of the calculations, it is hard to address these deficiencies. However, ChatGPT gives a good starting JupyterLab notebook that can be modified to account for the missing parts. All we need is to add a few

sections in the code to (i) draw the moment diagram and determine its maximum value and location, (ii) calculate the stress concentration factors taking into account actual diameter ratios, (iii) calculate the endurance limit accurately, and (iv) check sensitivity of different parts of the calculation to a change in diameter.

In modifying the code, the instructor discusses different steps and translates the equations into the notebook. The discussion makes the overall solution process interesting and engaging. The resulting working code is shown in Fig.3. It is easy to change the diameter in the code and see which values change and how much as a result of that change. In lectures, it plays an excellent role in showing the students interconnected steps and emphasize the critical steps that matter the most for the shaft design.

```
from numpy import *
from math import *
import matplotlib.pyplot as plt
def unit_step(x):
    return heaviside(x,1)

# Given information
L = 10.0; a1 = 2; a2 = 7.75; d3= 12; d4 = 2.67
S_ut = 68; S_y = 57
W23_r = 197; W45_r = 885
W23_t = 540; W45_t = 2431

#Reactions at the support
R_az = -W23_t*(L-a1)/L + W45_t*(L-a2)/L; R_bz = -W23_t*a1/L + W45_t*a2/L;
R_ay = W23_r*(L-a1)/L + W45_r*(L-a2)/L; R_by = W23_r*a1/L + W45_r*a2/L;

# Moment and torque equations
My = R_ay*unit_step(x)*x - W23_r*unit_step(x-a1)*(x-a1)**1 \
    - W45_r*unit_step(x-a2)*(x-a2)**1
Mz = R_az*unit_step(x)*x + W23_t*unit_step(x-a1)*(x-a1)**1 \
    - W45_t*unit_step(x-a2)*(x-a2)**1
Mr = (My**2 + Mz**2)**0.50
T = W23_t*(d3/2)*unit_step(x-a1) - W23_t*(d3/2)*unit_step(x-a2)

# Assumptions
d = 1.9; r_keyway = 0.02*d; Kt = 2.14; Kts = 3.0

# Step-1: calculate sigma_a' and sigma_m'
M_a = max(Mr)/1000; M_m = 0
T_a = 0; T_m = max(T)/1000
sqrt_a = 0.246 - 3.08e-3*S_ut + 1.51e-5*S_ut**2 - 2.67e-8*S_ut**3
Kf = 1 + (Kt - 1)/(1 + sqrt_a/sqrt(r_keyway))
sqrt_a = 0.190 - 2.51e-3*S_ut + 1.35e-5*S_ut**2 - 2.67e-8*S_ut**3
Kfs = 1 + (Kts - 1)/(1 + sqrt_a/sqrt(r_keyway))
sigma_a_prime = sqrt((Kf * (32*M_a/(pi*d**3)))**2 \
    + 3*(Kfs * (16*T_a/(pi*d**3)))**2)
sigma_m_prime = sqrt((Kf * (32*M_m/(pi*d**3)))**2 \
    + 3*(Kfs * (16*T_m/(pi*d**3)))**2)

# Step-2: Determine Se
ka = 2.0*S_ut*(-0.217)
if d <= 2.0:
    kb = 0.879*d**(-0.107)
elif d>2.0:
    kb = 0.91*d**(-0.157)
kc = 1; kd = 1; ke = 1; Se_prime = 0.5*S_ut
S_e = ka * kb * kc * kd * ke * Se_prime

# Step-3: Determine nf
if sigma_m_prime <= 0:
    n_f = S_e/sigma_a_prime
elif sigma_m_prime > 0:
    n_f = (sigma_a_prime/S_e + sigma_m_prime/S_ut)**(-1)
print("FOS: n_f = ",n_f)
print("FOS: n_y = ", S_y/(sigma_a_prime + sigma_m_prime))

FOS: n_f = 1.527081172765945
FOS: n_y = 2.6522161917024207
```

Fig. 3. A JupyterLab code generated to solve the design problem (only a part of the overall code is shown for brevity).

IV. INITIAL ASSESSMENT OF THE APPROACH

All the students were asked to provide feedback to assess the effectiveness of the SII approach. Based on responses from 152 students, as demonstrated in Fig. 4, the students liked seeing how JupyterLab makes it easy to understand the solution process and, more importantly, the significance of different parts in the overall design process. However, many students found it challenging to learn and use the software packages, likely due to a lack of training in Python-related coding tasks. While the students did take a Python-based course in their freshman year, they may not have been exposed to the structure of JupyterLab and its functions in the context of solving engineering problems.

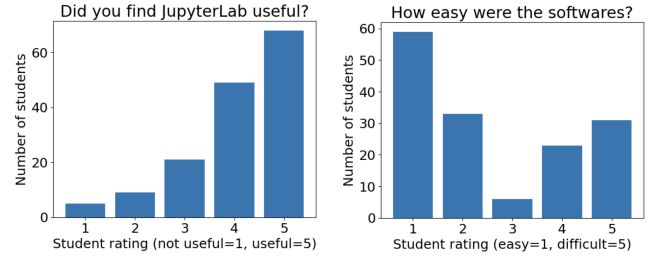


Fig. 4. Student response to effectiveness of JupyterLab and easiness of learning the software. Students used it for solving homework, quiz, and exam problems. They were allowed access to codes/software during the exam.

V. DIFFICULTIES FACED AND REMEDIES

Associate difficulties include extra time required to prepare the lectures (although once done it can be reused), unfamiliarity of TAs with the approach, and reluctance to learn JupyterLab. These were addressed by conducting recitations focusing on use and application of ChatGPT and JupyterLab. Students suggested recording the sessions (one or two one-hour sessions) that show the application of software integration. In spite of the steep learning curve, students appreciated the experience of using the tool.

VI. CONCLUDING REMARKS

In conclusion, JupyterLab-based solid mechanics has proven to offer a leap forward in the way solid mechanics is taught. Based on the initial implementation of the ChatGPT and JupyterLab combo (in teaching Solid Mechanics in Machine Design), the software tools not only helped students learn the material better and more effectively but also drove them to apply their understanding of the course content in modifying the code. Students who were able to complete it successfully gained three things: (a) how to use and evaluate ChatGPT to generate the initial solution, (b) how to apply theoretical understanding to modify the code to make it work correctly, and (c) how to use the code to analyze different design situations. By providing a versatile environment where code, equations, visualizations, and narrative text coexist seamlessly, JupyterLab facilitated a more engaging and effective approach to learning and research.

REFERENCES

- [1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [2] T. M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997, vol. 1.
- [3] H. Jin, E. Zhang, and H. D. Espinosa, "Recent advances and applications of machine learning in experimental solid mechanics: A review," *Applied Mechanics Reviews*, vol. 75, no. 6, p. 061001, 2023.
- [4] P. Jupyter, "Jupyterlab: A computational environment for interactive and reproducible computing," <https://jupyter.org>, 2023.
- [5] R. G. Budynas, J. K. Nisbett *et al.*, *Shigley's mechanical engineering design*. McGraw-Hill New York, 2011, vol. 9.
- [6] W. D. Pilkey, D. F. Pilkey, and Z. Bi, *Peterson's stress concentration factors*. John Wiley & Sons, 2020.