Assessing the Effectiveness of Open-ended Engineering Design Projects in a First-Year Engineering Programming Course for Improving Students' Problem-Solving Styles

Dr. John Alexander Mendoza-Garcia, University of Florida

John Mendoza Garcia serves as an Instructional Associate Professor at the Department of Engineering Education within the Herbert Wertheim College of Engineering at the University of Florida. He received his Ph.D. in Engineering Education at Purdue University, and his Master's and a Bachelor's in Systems and Computing Engineering from Universidad de Los Andes, in Colombia, and Universidad Nacional de Colombia respectively. He teaches and investigates the development of professional skills such as problem-solving, systems thinking, and design thinking. He worked in Industry before transitioning to academia.

Assessing the effectiveness of open-ended engineering design projects in a first-year engineering programming course for improving students' problem-solving styles

1. Introduction

Two of the engineering learning outcomes proposed by ABET are asking that students can "identify, formulate and solve complex engineering problems…" and "apply engineering design to produce solutions that meet specified needs…" That is why engineering schools are working on providing courses in which students must engage in solving open-ended problems to facilitate reaching these learning outcomes. However, most of those courses are typically at the end of the career path (Capstone design project) and maybe an introductory design course in their first year. In these courses, students must find a problem and work on defining a specific problem, which gets them closer to what the outcome expects. On the other hand, most of the courses engineering students take in their first year ask them to solve well-defined problems with a right answer (they might be able to take different paths to get to that answer, but there is still one right answer). Several engineering education thought leaders have called for incorporating the development of professional skills, like problem-solving for open-ended engineering design problems, across all the different engineering courses. Following such a call, I, the author of this paper, incorporated an engineering design project into the Computer Programming for Engineers course taught at University of Florida for two semesters, hoping that such instructional intervention positively impacts students' problem-solving skills.

2. Frameworks

2.1 Conceptual Framework

2.1.1 Social Problem-solving

There are many ways in which literature has defined problem-solving; still, assessment tools for measuring such skills are scarce. In this study, I used a model developed by D'Zurilla et al. [1] in which their team proposes a focus on the process of problem-solving and does not want to limit the process to particular kinds of problems but to those that "influences one's adaptive functioning in the real-life social environment" (and engineering problems are some of those.) D'Zurilla et al.'s model uses the term social Problem-Solving for such problems. The model develops the concepts of "problem-solving," "problem," and "solution," specifying that problem-solving "refers to the process of finding a solution." In contrast, "solution" refers to "carrying out those solutions in the actual problematic situations." The model comprises "problem orientation" and "problem-solving skills." Through these components, they developed the Social Problem-Solving Inventory, which has two scales: Problem-Solving Orientation Scale (POS) and Problem-Solving Skills Scale (PSSS). Problem Orientation refers to the "disposition" to problems; therefore, the scale has a measure for Positive Problem Orientation (PPO) and Negative Problem Orientation (NPO). Regarding Problem-Solving Skills, they developed the idea of 3 different styles that describe it: Rational Problem Solving (RPS),

Impulsivity/carelessness, and avoidance style. Based on this model, D'Zurilla et al. describe the problem-solving process as either constructive or dysfunctional (see a schematic in Figure 1).

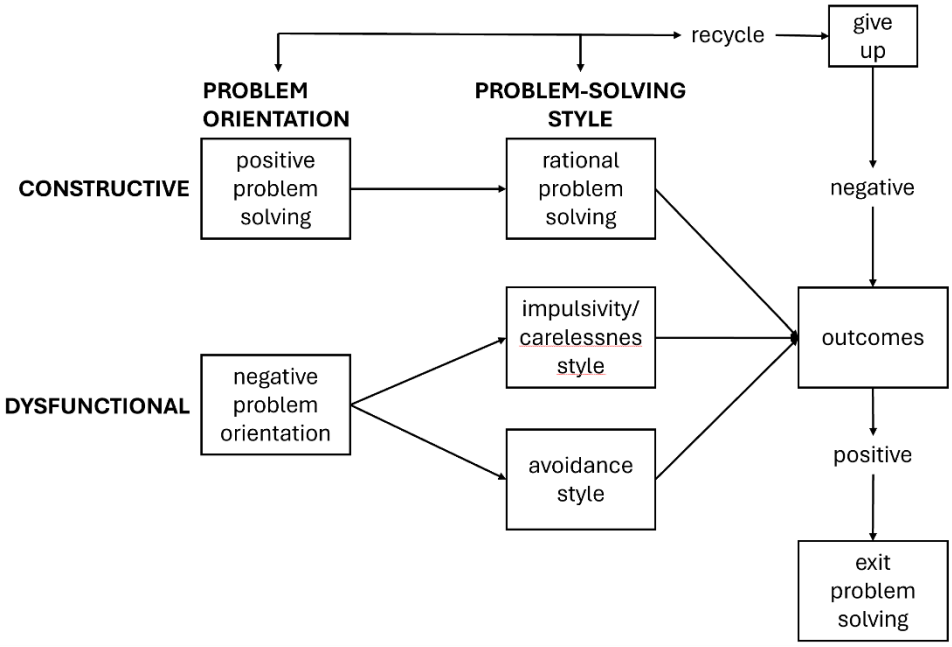

Figure 1 - Figure created by D'Zurilla et al. [1] showing their model for the social problem-solving process based on the five-dimensional model developed by their team (2002).

According to the model for the problem-solving process, a positive orientation leads toward a rational problem-solving way, which is more likely to produce positive outcomes. On the contrary, a negative problem orientation leads towards styles such as impulsive, careless, or just avoidance, which are more likely to generate negative outcomes and motivate the problem-solver to give up.

2.2 Hypothesis

A course like programming is expected to develop problem-solving skills, and following this model, an expectation would be that a typical programming course increases students' positive problem orientation and, because of the nature of the programming problems, would increase the rational problem-solving style level. Starting from that assumption, I hypothesize that adding an engineering design project to the course will make higher gains in these two styles while contributing to a higher decrease in the dysfunctional styles.

2.3. Assessing Problem-Solving Styles

D'Zurilla et al. developed the Problem-Solving Inventory, a Likert-type questionnaire in which participants self-report their perception of their "problem-solving behavior and attitudes." The number of questions started at 50, but D'Zurilla and other researchers have found ways to reduce the number of questions, keeping the consistency and validity of the questionnaire. In this study, I used the Social Problem-Solving Inventory-Revised Short-Form with 25 questions [2] based on participants' answers.

2.4 Contextual Framework

2.4.1 The course content

I am the instructor of record in one of the sections of this university's two-credit Computer Programming for Engineers course. In this course, students learn how to code solutions for programming problems using MATLAB as a programming language. The course is divided into 12 modules; the first five focus on developing general programming skills using decision structures and loops. The course's second part develops students' ability to work with vectors and matrices. For each module, students must complete three low- and middle-level coding problems and a higher complexity problem or problems as homework. A brief description of each module is offered below.

In module 2, students start learning about the MATLAB programming environment, how they can ask for input from the user, and how they can execute a process and create an output for the user with different formatting options. In the homework, students are asked to code the calculations to find the angles of a triangle given the length of three sides, which implies the implementation of a formula that uses the input from the user and delivers the angles of the triangle as output.

In module 3, students learn about the decision-making structures, specifically the conditionals if-else-end, if-elseif-end, and how they behave if nested. In the activities, they solve problems like finding the amount customers are charged for their water usage based on a tiered system (e.g., between 0 and 5,999 gallons, they must pay $2.35 per 1000 gallons, and between 6,000 and 20,000 gallons, they must pay $3.75 per 1000 gallons. For the problem, the user inputs the water usage, and the code will return the money owed. There is another activity in which students implement a currency converter in which the user inputs a transaction amount in dollars; for example, the code calculates the equivalent in Yens or Canadian dollars. In the homework, students are asked to implement the code that decides the cost of an ice cream based on user selections for size, flavor, and toppings.

Regarding modules 4 to 6, students learn applications of indefinite (while-loop) and definite loops (for-loop). Some of the problems in these modules are the calculation of mathematical series after using several terms and another is creating patterns. For example, in module 4, students are asked to calculate the pi using the Wallis formula, which involves the addition of 1 or more terms. Therefore, in the code, the user is asked how many terms they want to use to make the calculation (e.g., 500), and the code will generate the output using the number of terms the user provided as input (for 500 terms, the output is 3.14002068). Similarly, in module 5, for-loops, the student must write code that calculates the factorial of a number (they cannot use the factorial function pre-defined in MATLAB; these series in module 6 are more complex because of the use of nested loops. The other kind of problem is patterns. In this area, students are asked to create different shapes using nested for-loops and an if structure for deciding if printing and what to print. For example, the border of a square made of *, or the multiplication chart that students in elementary school use.

In modules 8 to 12 (module 7 is the exam), students are introduced to strings and matrices. In module 8, they learn about vectors in MATLAB, with topics like indexing, deleting values in a

vector, and using vector functions like *find, sum, isequal, isempty*, etc. As activities, they are asked to rotate a vector, delete the content of a vector based on specific criteria several times, and implement a sorting algorithm. In the homework, they are asked to implement the code that calculates the score that would be obtained by the user if throwing 6 dice (e.g., the sum of the dice, a score if they are consecutive, the score when you obtain 3 pairs, among others). In Module 9, students work on string vectors. In this module, we use the concept of ciphers, which is applied in several activities. As usual, activities start with low complexity, asking them to write code that calculates the number of words in a phrase. Activities also ask for the manipulation of strings, such as deleting the repeated letters or reversing the letters and punctuation in each word of a phrase. In modules 10 and 11, students are introduced to matrices through images. In the course we use digital images that use the Red, Green, and Blue model (RGB), in which each coordinate row or column holds a number that represents the intensity of each of these colors, which when combined, and drawn on the screen, show the color of a pixel to the human eye. These values typically range from 0 (no intensity) to 255 (maximum intensity). Therefore, they are asked to rotate or flip an image, create specific shapes like circles or a chessboard, count the number of coins in an image, find an object in an image, and decode an image by multiplying the numbers on each pixel (RGB) by a specific amount calculated based on the values already existent among others.

Finally, in module 14 (13 is exam 2), students are taught about functions, which they will use when implementing their solution to the final project. In this final assignment, students implement one step in the 2048 game. Specifically, a 4x4 board is given to them, and they must write a function that reads that board in the image and convert it to a MATLAB matrix. They must also write another function that, based on a board configuration, produces the new board after a movement in one of the directions: left, right, up, or down, and it also calculates the new score. The new board and the new score are returned to the executive function that called them.

2.4.2 Teaching style

The course is taught following a flipped classroom model in which students watch the lecture before class (e.g., at home), and in class, they come to work on the course activities. Activities are due at the end of the day the class is taught, and homework is due three or four days later. The course has been taught following this scheme for several years. Although encouraged to talk to each other, students are expected to create their solutions independently and not copy them from other students. The due dates in the course are mostly not flexible.

2.4.2 The Experiment: Including an Engineering Design Project

In fall 2021 and spring 2022, students in that course were asked to complete an Engineering Design Project in addition to their programming work. To balance the course workload, homework was optional in one semester, while activities were optional in the other semester. The expected product students had to create was an App developed in MATLAB in which students implemented their solution to a programming problem that they defined. For this project definition, students worked on two deliverables: problem scoping, in which they decide the topic of their project, and then define a problem and a project that deals with such a problem. Before

continuing, the instructor had to approve their problem statement and project goal. Since most of the students did not have programming experience before the course, and they were learning at the same time they were proposing the project, the problem they focused on was not expected to have high complexity in terms of implementation. However, they were expected to follow the design process when defining the solution they would implement. Once their project was approved, students had to follow an idea-generation stage in which they proposed different implementation ideas, followed by a stage of idea reduction and selection using strategies like pros/cons and the Decision-making matrix. The next stage was the creation of a rapid prototype in PowerPoint in which they graphically showed how their App would look and what interactions it would make with the user. In the next stage, students develop their ideas for coding the different functionalities using flowcharts or pseudocode, which they implement in MATLAB in the following 3 stages: Beta 1, Beta 2, and final presentation. For the final presentation, students, in addition to the program implementation, must write a scope document and create a video in which, in the first part, they present their project in a Shark Tank-style in which they present to possible investors, therefore, they introduce the problem they tackled, and the solution they created. The audience changes in the second part of the video; this section is for the instructor and focuses on the implementation. Part of what students are required to do for the project is to learn how to use the MATLAB App designer, following the videos provided by Mathworks, the company that developed MATLAB.

Regarding the project teamwork, students were required to work in teams. Such teams were created and assessed using the Comprehensive Assessment of Team Member Effectiveness – CAMTE tool developed by Purdue University. The tool was used to ensure that the teams could meet and were balanced in gender and skills. During the semester, teams were asked to assess their peers three times in the semester, two using CATME, which facilitates the evaluation of peers in a team, and in the final project defense in which the team meets with the instructor and is asked to evaluate their peer's work during the meeting.

3. Methods

3.1 Participants

Participants in this study are engineering students who took my section of the course Computer Programming for Engineers from the spring and fall of 2021 to the spring and fall of 2023. Students were mainly in their first year of engineering and were enrolled in different majors (several engineering programs require this class). Most of the students took the pre-test, while approximately half of them took the post-test, except for spring 2023, in which the same number of students who took the pre-test equaled the number of students who took the post-test. Table 1 shows in detail the number of students who took the survey the first week of classes (pre) and the last week of classes (post) per semester.

	Spr-21	**Fal-21**	**Spr-22**	**Fal-22**	**Spr-23**	**Fal-23**
Pre	21	49	17	47	41	42
post	10	24	7	34	41	25

Table 1 - Number of students per semester who completed the Problem-Solving Styles Survey the first week of classes (pre) and the last week (post)

3.2 Data collection

A pre/post-test was designed for this study to find changes in students' problem-solving styles after taking the course. Consequently, students were asked to answer the Social Problem-Solving Inventory-Revised Short-Form (SPSI-R) on the first and last week of classes. The survey was created in Qualtrics, and student scores were automatically calculated and shown to students. Results were saved without any identifier. Students were offered extra credit for taking the survey. Still, since participation in the study was voluntary and allowed anyone to get the extra credit, students were told that they could skip all the questions, and when reaching the last page, they could take a screenshot to get the extra credit. In this way, any student could get the extra credit regardless of whether they take the survey. The survey was anonymous to avoid possible student identification. Each semester's pre and post-test scores were gathered in an Excel file, and the data for this study was analyzed there.

4. Findings

In spring 2021, RPS increased from 10.86 to 11.46, while PPO rose from 11.29 to 11.54. This increase was also seen in the fall of 2021, from 12.41 to 13 for PPO and 11.29 to 11.54 for PPO. Likewise, most scores that were expected to decrease did so. In the spring of 2021, scores for NPO decreased from 6.31 to 5.79, and impulsivity/carelessness also reduced from 4.2 to 4.17. Still, the score for avoidance style increased from 4 to 4.25. In the same way, in the spring of 2022, the scores for avoidance style decreased from 6.29 to 5.71, and NPO also reduced from 7 to 4.86. On the other hand, scores for Impulsivity/carelessness increased from 6.69 to 6.91. A summary of these results is shown in Table 2.

	Fall 2021			**Spring 2022**		
	pre	**post**	**change**	**pre**	**post**	**change**
Positive Problem Orientation (PPO)	11.29	11.54	0.25	10.82	12.57	1.75
Rational Problem Solving (RPS)	10.86	11.46	0.6	12.41	13	0.59
Avoidance Style	4	4.25	0.25	6.29	5.71	-0.58
Negative Problem Orientation (NPO)	6.31	5.79	-0.52	7	4.86	-2.14
Impulsivity/Carelessness	4.2	4.17	-0.03	3.76	4.29	0.53

Table 2 - Scores in pre-posttest when an engineering design project was included in the course.

The relative magnitude of the change in the scores can be visualized in the chart below in Figure 2.

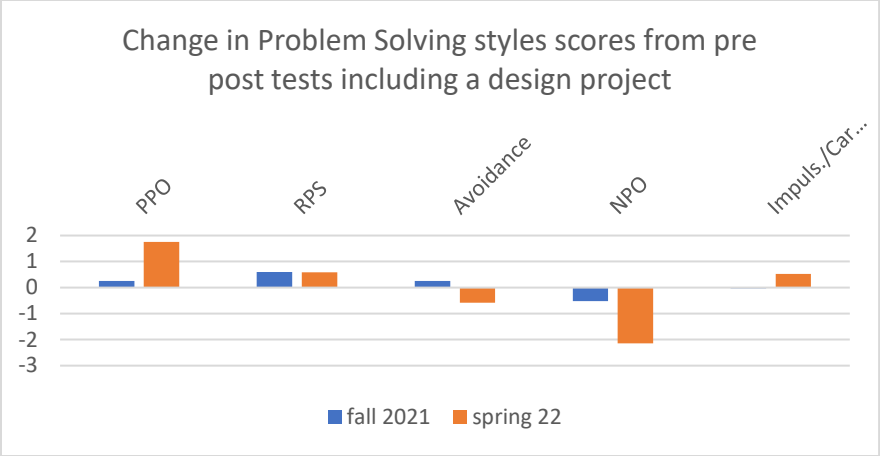

Figure 2 - Relative Magnitude change in pre-posttests for fall 2021 and spring 2022 - including an engineering design project.

In the spring and fall of 2022 and spring of 2023, the class was taught in the "traditional" way in which students were required to complete course activities and homework, solving well-structured and defined problems. In the fall of 2022, RPS decreased from 11.23 to 11.21, and PPO reduced from 11.17 to 10.76. in the Spring of 2023, although RPS increased from 11.27 to 11.51, PPO decreased from 11.61 to 10.93. On the other hand, the avoidance style decreased in the fall of 2022 from 5.85 to 5.41, but it increased in the spring of 2023. NPO scores increased in fall 2022 (6.69 to 6.91), spring 2023 (6.88 to 7.41), and avoidance from 5.46 to 6.73. A summary of these data can be seen in Table 3.

	Fall 2022			Spring 2023		
	pre	post	change	pre	post	change
Positive Problem Orientation (PPO)	11.17	10.76	-0.41	11.61	10.93	-0.68
Rational Problem Solving (RPS)	11.23	11.21	-0.02	11.27	11.51	0.24
Avoidance Style	5.85	5.41	-0.44	5.46	6.73	1.27
Negative Problem Orientation (NPO)	6.69	6.91	0.22	6.88	7.41	0.53
Impulsivity/Carelessness	5.35	4.32	-1.03	4.71	4.39	-0.32

Table 3 - Scores in pre-posttest when an engineering design project was not included in the course.

The relative magnitude of the change from the pre-test to the post-test can be seen in the chart in Figure 3:

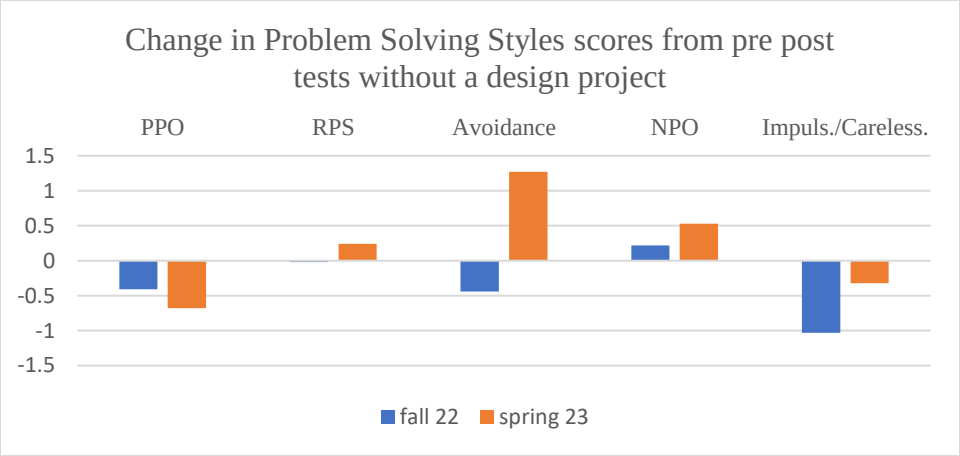

Figure 3 - Relative magnitude change in the scores of pre/post-tests in fall 2022 and spring 2023 - Regular course without engineering design project

5. Discussion

The expectations were that when in a regular programming course, students would increase their Positive Problem Orientation and Rational Problem-Solving scores and, at the same time, decrease their scores for the styles of Negative Problem Orientation, avoidance, and Impulsivity/carelessness. It was also expected that including an engineering design project would increase the gains in PPO and RPS while decreasing the scores for NPO, avoidance, impulsivity, and carelessness similarly.

The data suggests that the problem-solving style scores (RPS and PPO) increased when the engineering design project was included as expected. Similarly, in these two semesters, scores for NPO decreased as expected. On the other hand, when the course was taught in the "traditional" way, focused on asking students to solve only well-defined programming problems, the expected-to-increase problem-solving style scores (PPO and RPS) decreased. Likewise, NPO scores increased in both semesters, which was also unexpected.

Engineering students are expected to develop high levels of engineering problem-solving skills, which ultimately impacts motivation and retention. According to the findings in this study, when teaching programming to these students, there are teaching opportunities that can be implemented to improve students' problem-solving styles, such as Engineering Design Projects. These kinds of projects are effective for this purpose if they follow a Project-based Based Learning approach, which is *"characterized by students' autonomy, constructive investigations, goal setting, collaboration, communication and reflection within real-world practices* [3]*."* The results also show that the nurturing of problem-solving styles in engineering students can go hand-in-hand with the learning of technical Engineering skills. Including such opportunities for students to work on engineering design projects in non-engineering design courses, but in more traditional engineering technical courses is an idea that is worth considering. In such a case, instructors and course developers, when defining their course goals, along with the development of technical skills, could include a better response to problems (e.g., using problem-solving

styles), and they can also talk about students' motivation towards engineering. Instructors could also track the number of students who usually drop their class and assess if the change impacts students' retention. In this way, replacing some course content to include an engineering design project can be better justified and aligned with ABET SLOs [4].

The impact of PBL and engineering design projects on motivation and retention has been previously studied. [5] found that engineering students taught through PBL strategies perceived that the course content is useful, a key factor in learning and motivation to learn [6]. Similarly, in [7], the authors studied the retention of mechanical engineering students, finding that students' persistence increased to 79% after they created an introductory engineering design course.

Limitations

The results from this study are limited by the number of students who participated in the survey at the beginning and end of the course. In the pre-test, the number of participants in most cases is almost double the number of students who complete the post-test survey. However, this is mitigated by having scores from more than one semester. I also acknowledge that all the students were taking other courses; therefore, it is unknown if their problem-solving styles changed only due to their experience in this course. Finally, results are also limited to the discipline and the kind of problems that students were exposed to in the programming course.

Future work

Since I recognize that including a design project in the programming course is not always feasible, it would be relevant to test if different teaching styles could lead to a better outcome regarding the development of problem-solving skills. Also, since all the courses share a similar shell, it will be relevant to include additional programming sections taught by different professors to increase the sample and strengthen the evidence. It is also possible to study students' retention in a class before and after the change and track their persistence in engineering after several semesters.

References

[1] T. J. D'Zurilla, A. M. Nezu, and A. Maydeu-Olivares, "Social Problem Solving: Theory and Assessment.," *Social problem solving: Theory, research, and training.*, no. 1971, pp. 11–27, 2009, doi: 10.1037/10805-001.

[2] K. Sorsdahl, D. J. Stein, and B. Myers, "Psychometric properties of the Social Problem Solving Inventory-Revised Short-Form in a South African population," *International Journal of Psychology*, vol. 52, no. 2, pp. 154–162, 2017, doi: 10.1002/ijop.12192.

[3] D. Kokotsaki, V. Menzies, and A. Wiggins, "Project-based learning: A review of the literature," *Improving Schools*, vol. 19, no. 3, pp. 267–277, Nov. 2016, doi: 10.1177/1365480216659733.

[4] ABET, “Criteria for Accrediting Engineering Programs, 2023 - 2024.” Accessed: Mar. 31, 2024. [Online]. Available: https://www.abet.org/accreditation/accreditation-criteria/criteria-for-accrediting-engineering-programs-2023-2024/

[5] H. M. Matusovich, M. C. Paretti, B. D. Jones, and P. R. Brown, “How problem-based learning and traditional engineering design pedagogies influence the motivation of first year engineering students,” *ASEE Annual Conference and Exposition, Conference Proceedings*, 2012, [Online]. Available: https://www.engineeringvillage.com/share/document.url?mid=cpx_6e3d60139686bec03M67d72061377553&database=cpx

[6] D. Perkins, *Making Learning Whole: How Seven Principles of Teaching Can Transform Education*, First edit. San Francisco, California: Jossey-Bass, 2009.

[7] R. Roth, “Improving freshman retention through an introduction to engineering design course,” *ASEE Annual Conference Proceedings*, pp. 5653–5660, 2001, [Online]. Available: https://www.engineeringvillage.com/share/document.url?mid=cpx_765291100a99d6fddM60bc19255120119&database=cpx