2026 ASEE ANNUAL CONFERENCE June 21–24, 2026 | Charlotte Convention Center, NC
Host Sponsor UNC Charlotte
ASEE AMERICAN SOCIETY FOR ENGINEERING EDUCATION
Paper ID #53521

A Plug and Play Guide to Efficient Oral Assessments in Mid to Large Sized Manufacturing Courses

Dr. Sandra Walter Huffman, Tufts University; Massachusetts Institute of Technology

Sandy is a postdoctoral scholar at Tufts University working on Engineering Education research. She is interested in student modeling practices, knowledge practices, and sense-making activities in undergraduate engineering courses. She looks at the ways traditional classroom contexts influence student problem-solving methodologies, and ways task design and facilitation strategies can help students practice authentic engineering. She received her undergraduate and masters degrees in the Department of Mechanical Engineering at MIT, focusing in product design for emerging markets and engineering education research, accordingly. Her interdisciplinary PhD, also based in the Department of Mechanical Engineering at MIT, centered the sociocultural ways students engage with technical engineering material and suggested directions for a more integrated, authentic approach to helping students develop technical modeling practices that utilize course material in unfamiliar contexts.

Kaitlyn Becker, Massachusetts Institute of Technology
Dr. John Liu, Massachusetts Institute of Technology

Dr. John Liu is the Director and Principal Investigator of the MIT Learning Engineering and Practice (LEAP) Group, which investigates the intersection of human-centered technologies, engineering workforce and education, and digital learning. He is a Principal Research Scientist in MIT's Mechanical Engineering department and a Scientist in the MIT Digital Learning Lab. He leads education and workforce development efforts for MIT's Initiative for New Manufacturing. He was the Director of the Principles of Manufacturing MicroMasters program, an online certificate program that has now enrolled over 200,000 learners across the globe. His work has received recognition from venues such as ASEE, IEEE DEMOCon, and ACM CHI. He has co-authored dozens of publications and currently serves as an executive guest editor for Manufacturing Letters. His research is supported by the Department of Commerce, Department of Defense, Massachusetts Technology Collaborative, Schmidt Futures, National Science Foundation, MIT, and industry partners.

A Plug and Play Guide to Efficient Oral Assessments in Mid to Large Sized Manufacturing Courses

Introduction

Authentic assessment, an assessment method designed to simulate real-world engineering problems, is a tool many university instructors use to help students develop professional engineering practices and prepare them for their future careers [1], [2], [3], [4]. While traditional assessment features well-defined single-solution problems and rewards algorithmic memorization, authentic assessment centers conceptual understanding, multifaceted problem-solving, and design thinking to reward a deep, functional understanding of course content [5], [6], [7], [8], [9], [10]. These practices are even more important now that AI systems can solve most university engineering exam problems [11].

Short oral assessments are one way to conduct authentic assessment. This form of assessment can resemble design reviews, consulting sessions, a workplace brainstorming session, or a technical interview. It emphasizes the types of engineering communication and real-time problem solving that engineers could expect in their day-to-day work. Despite its merits, many university instructors hesitate to conduct short, authentic oral assessments in their classes. This is in part due to a lack of training and the perception that authentic oral assessments cost significant time and are difficult to standardize across students [12], [13].

In this paper, we attempt to alleviate these limitations by presenting a guide to short authentic oral assessments for university manufacturing courses. We outline two 7-minute oral assessments, a *Design for Manufacture* assessment and a *Data Interpretation and Troubleshooting* assessment. These assessments, originally described in detail in *Reflections on the implementation of short, authentic oral assessments in a university manufacturing course[4], [14]*, each align with a four-competency five-point rubric that allows instructors to quickly grade assessments and have transparent, standardized grading across all students. Rubrics can be found in Appendix A and Appendix B. Additionally, we describe a process for using AI to generate datasets for the second assessment, and discuss the merits and shortcomings of this approach. These time-saving tools have made oral assessments a more sustainable assessment tool for these instructors. Although they do not believe it feasible to reduce the total time commitment (including grading) below 10 minutes per student plus 2-3 hours for brainstorming, calibrations, and revisions, they find the time commitment reasonable and sustainable.

In implementing these short authentic oral assessments, the teaching team has centered authentic engineering practice in the classroom component of their course, developed better relationships with students, and helped students achieve their own goals. Students reported a positive experience taking the assessments, and appeared to understand the merits of authentic orals. In the past year, current and former students have reported that the oral assessments helped them

prepare for and navigate technical interviews. The short, authentic, oral assessments provide a clear benefit to the teaching team, and they plan to continue to conduct them in their class in years to come.

“Mad-Libs” “Plug-and-Play” Assessment Format

Over the past two years (4 assessments) the teaching team has worked to find their rhythm in the creation and implementation of the oral assessments. We have developed a “Mad-Libs” or “Plug-and-Play” format for two common manufacturing topics: *Design for Manufacture* and *Data Interpretation and Troubleshooting* to save time creating future assessments, making them more sustainable from a time commitment and effort invested standpoint. We share this system here for any instructor hoping to conduct oral assessments, but worried about the time commitment.

We encourage instructors to build off of this template as they see fit; as you become more comfortable with oral assessments, you may use this more as a brainstorming tool than a guide. These scripts are designed to fit a wide range of products in the teaching team’s manufacturing course. They align with the specific content of the course. Questions should be altered or adjusted to fit different curriculums. Instructors found it productive to generate more questions than needed. These extra questions gave opportunities to vary the assessment between students, and helped instructors imagine a range of student responses. For some questions, additional branches to scripts were created. This was helpful for when students went in unexpected directions or moved faster than expected.

One essential element of these oral assessments is the inclusion of a storyline. Picking a storyline helps contextualize the oral assessment in an engineering setting. When both the instructor and student are playing a role, it is more likely that students will engage with the problem, rather than looking for the instructor’s approval. This makes the assessment more authentic and helps students practice giving engineering advice as an engineer rather than as a student.

Note: the script makes reference to all rubric lines except *Communication*. Engineering communication — the ability to communicate clearly, articulating precise concepts with appropriate field specific language and to articulate gaps in understanding and navigate a logical path forward when stuck — is an essential engineering practice and a skill that can be developed by all students. The teaching team believes that an emphasis on engineering communication (as defined above) is one of the key strengths of these assessments.

Assessment 1: Design for Manufacture

The first assessment is oriented towards design for manufacturing. This assessment focuses on a product: how it is made, what tradeoffs may be needed to make adjustments, and the mechanics of manufacturing parts at scale. Students are asked to analyse a product in the context of a

fictionalized storyline and answer questions as a “consultant,” giving manufacturing advice to the instructor’s fictionalized role.

Step 1: Choose a Product

These are typically cheap, easy to access products such as small metal/plastic toys, food containers, or desktop organizers. Visible signs of manufacturing method such as ejector pin marks, flash, or gate marks will help generate conversation during the assessment. Having many slightly different versions of the product and choosing a couple for each student at random helps vary the assessments. This variation helps keep assessments distinct for instructors, making grading easier while ensuring integrity among students attempting to share information about the assessments.

Step 2: Create a Storyline

Help pull students into an authentic engineering situation with a story. Be ready to introduce yourself “as” a toy designer asking for support, a CEO ready to scale up your small company, a skeptical inspector, or a confused line engineer. It may be a role you’ve held before or one you’ve only imagined, just be ready to explain a few sentences about the situation and why you’re asking help from the student.

Step 3: Fill in the Script

Regardless of the product or storyline, the assessment follows the same path, and matches the grading rubric, which can be found in Appendix A. Adapt this script to fit your tone, story, and content preferences, and ask follow-up questions where relevant.

- Invite the student in and introduce the storyline (about 30 seconds)
- {Warm-Up} How was this made? How do you know that?/Show me what you’re looking at. (about 30 seconds)
- {CRQFS} (about 1 minute) Pick 1:
    - To be able to [INSERT: goal from story], I really need to bring the cost down. How would I do that while maintaining [CHOOSE: quality, rate of production, this material, etc]?
    - I want to have a variety of [INSERT: product name] so that I can [INSERT: goal from story]. What sort of flexibility does this process allow for?
    - I love this [product name], but I want to make it more sustainable. How might I do that? What sort of tradeoffs do I need to make?
    - I want to make [INTERT: product subpart] [CHOOSE: stiffer, smaller, larger, smoother, squishier, more durable, etc], how would I do that? Would that effect the [INSERT ONE: CRQRS]
- {DFM} (about 1.5 minutes) Pick 1:
    - Could I instead [INSERT: another manufacturing process] this part? (if not) What would I need to change about the design to do that?

    - I want to [INSERT: some change that makes the product harder to manufacture OR changes the way it could be manufactured]. How will this affect the manufacturing process?
        - Is there something else I could do to [INSERT: goal of change] that would make [INSERT: manufacturing method] more possible? OR
        - Is there a manufacturing method more suited to this change?
- {Process Parameters} (about 2.5 minutes)
    - Can you please create a very quick/rough sketch of the tool (mold) used to make this part?
    - Choose one or more depending on time:
        - What is the rate limiting step for this process?
        - What dimension or parameter has the biggest effect on cooling time [or other parameter]?
        - Draw or indicate how plastic flows.
        - If I want to estimate the pressure drop, what is the characteristic length I would use?
        - Did the [buck/form] contact the outer or inner surface of the part? How can you tell?
- {Bonus/optional} for students who finish quickly or might need a challenge; could be any relevant challenge question, and is generally more contextualised and open-ended answers to this question can only help a student's grade, not harm it. Or, go back to a previous question where the student did not earn full points to give them a chance to improve.

Step 4: Rubric-Question Alignment

Review the questions and rubric together to check if they are in alignment. If they are not, make adjustments where necessary. Note: Instructors have found that changing the order of rubric line items to match the order you ask questions in the quiz is helpful for quick and straightforward grading.

Step 5: Practice

Practice with teaching assistants, students from previous semesters, and anyone else you have access to. Have some participants take the assessment genuinely. Invite others to come up with absurd or slightly off-kilter answers to throw off instructors. Practice test takers, especially former students, often needed reassurance that it was okay to provide these types of answers. Instructors found these "mistake" answers most helpful in their practice, and made that known to practice test takers as clearly as possible. Practice using the rubric during all practice assessments. All practice is good practice.

Assessment 2: Data Interpretation and Troubleshooting

The second assessment is oriented towards data interpretation and troubleshooting, but may contain other manufacturing content. This assessment focuses on data: identifying characteristics of data, speculating on the connection between the data and potential sources of process variation, troubleshooting and problem-solving potential issues, and making plans for how to confirm hypotheses and resolve problems. Students are asked to analyse one or more datasets in the context of a fictionalized storyline and answer questions as a "consultant," giving advice to the instructor's fictionalized role.

Step 1: Choose a Product and Dataset

The product for this assessment can be more complex or novel; only one or two objects are needed because the variation of this assessment is in the datasets. In the following section, *The Creation and Use of AI-Generated Datasets for Authentic Assessment,* we describe the process of generating fabricated numerical datasets using ChatGPT. Any numerical datasets that show a change over time or difference between two scenarios (plants, machines, seasons, runs, etc) can work well. Having several different datasets and choosing one for each student helps vary the assessments.

Additionally, you may choose to incorporate other types of data. For instance, you may be interested in student response to a toolpath drawing, profilometry scans, or defective parts or molds. While these are exciting data elements, they may not be feasible for every instructor due to instructor capacity and material availability.

Step 2: Create a Storyline

This assessment involves data, so the storyline should incorporate a plausible explanation of why raw data exists and is important. Maybe something went wrong on the plant floor. Maybe this is test data from a newly installed machine. Maybe we are trying to choose between plants and have data from each. Maybe we are scaling up a product and want to see if a new method produces appropriate results. Regardless of the situation, set a plausible role for both you and the students to play during the assessment as a way to get them out of an exam mindset and into one of authentic engineering problem solving.

Step 3: Fill in the Script

This script is more flexible than that of the *Design for Manufacture* assessment because different types of data and/or artifacts may be used. We recommend between one and three data sources for a short oral assessment, with fewer leading to more comfortable timing. For example, in our most recent assessment, we used injection molding molds and CAM paths from an old student project for our first data source, and AI generated datasets for the second data source. We spent about half the time on each data source.

- Invite the student in and introduce the storyline and artifact. Describe the *problem* you are (your character is) having. Then, present a data source or artifact. This could be the product, a defective part, a mold, graphs, or something else (30 seconds)

For each data set, choose questions from the following categories, with optional follow-ups. These categories are aligned with the rubric, which can be found in Appendix B. While it is not necessary to cover every category for each dataset, all categories should be covered by the end of the oral assessment. This should, in total, take the rest of the oral assessment (6 minutes, 30 seconds)

- {Data Interpretation} Pick 1:
  Note: the follow-up options shown under the first question can be applied to any
  - What are you noticing about the [data set OR artifact]?
    - *(for physical artifact)* Can you point to what's causing [the problem described in the introduction]?
    - *(for physical artifact or image)* Do you see anything that could explain [the problem described in the introduction]?
    - *(for graphed data)* Can you point to when the part stops being in spec?
    - *(for graphed data)* What is happening to the [CHOOSE: average, standard deviation, spread, ect] over time?
    - *(for multiple distinct graphed data sets)* What is the main difference between the two sets of data?
  - Do you see anything surprising or notable?
  - *(for graphed data)*Talk to me through what you're seeing in this data set.
- {Process Variation} Pick 1with in-depth follow-ups:
  Note: For our oral assessments, we spent the most time in this section as instructors believe it to be the cornerstone of course content. Follow-ups were based on the details of specific student responses.
  - What do you think could be happening with the process to bring about this change?
  - What do you think is happening and why?
- {Process Error and Problem Solving} Pick 1 with follow-ups as needed
  - How could you test your hypothesis?
  - What information would you need to confirm that [INSERT: student's idea] is the problem? How would you get that information?
  - What recommendation would you make to fix this problem?
    - If you made those changes, what tradeoffs are you making?
    - And/or what new challenges might arise (CRQFS, variation, aesthetics, tooling, etc.)?

Step 4: Rubric-Question Alignment + Step 5: Practice

Just as in the *Design for Manufacture* assessment, make sure to check the rubric and script for alignment, and practice as much as possible.

The Creation and Use of AI-Generated Datasets for Authentic Assessment

The first time these assessments were conducted, fictional datasets were created. These datasets took the form of very processed data; they were normal curves with shading and labels. Examples of these data sets are shown in Figure 1.

Figure 1: Example graphs from the first year of the *Data Interpretation and Troubleshooting* oral assessment

These datasets worked well for the assessment, but the teaching team hoped to get data more representative of something an engineer might encounter if they were troubleshooting a manufacturing problem. In their job, an engineer would likely be asked to make sense of raw data, so the inclusion of such data helps make the assessments more authentic. The team hypothesized that this less processed data could lead to interesting conversations around ambiguity and atypical data components. While it was possible to adapt old project data, this process was deemed too time-intensive for the assessment. The team decided to investigate if AI could be used as a time-saving tool to produce authentic-looking datasets. After a few attempts, successful datasets were created using ChatGPT. Appendix C shows a complete transcript of the conversation.

The initial prompt was important for setting the scope and starting in the right direction. Here, we break down the initial prompt. The prompt begins with a short description of the assessment: *"Hello- I am working on creating assessments for a university manufacturing class and would like help generating fabricated data sets. The students are going to be analyzing a production run of an injection molded yo-yo. Specifically, I would like them to analyze a scatter plot of diameters from a run (50-100 parts) of yo-yo rings. I would like several data sets for variety. In each data set, there should be a change over time. The students will be asked to troubleshoot what might have happened in the manufacturing process to cause this change."* While this description was short and vague, it gave enough context to keep the chatbot in scope. Then, the prompt moves to the request, *"I want to go back and forth with you to make sure everything is how I imagine it. Please ask if you have any clarifying questions, otherwise, please start with 3 scatter plots."* ChatGPT will try to give an output with incomplete information. Explicitly asking

for follow-up questions can help avoid unnecessary back and forth later in the query. ChatGPT did come back with several follow-up questions that led to reasonable datasets. Although its ideas about what the datasets represented were not always perfectly reasonable, it was able to come up with scatter plots that could easily be used in an assessment. Additionally, it was easy to go back and forth to make changes. For instance, if the way the data changed over time was too obvious or not obvious enough, it could be fixed in one exchange. One set of ChatGPT-generated datasets can be seen in Figure 2.

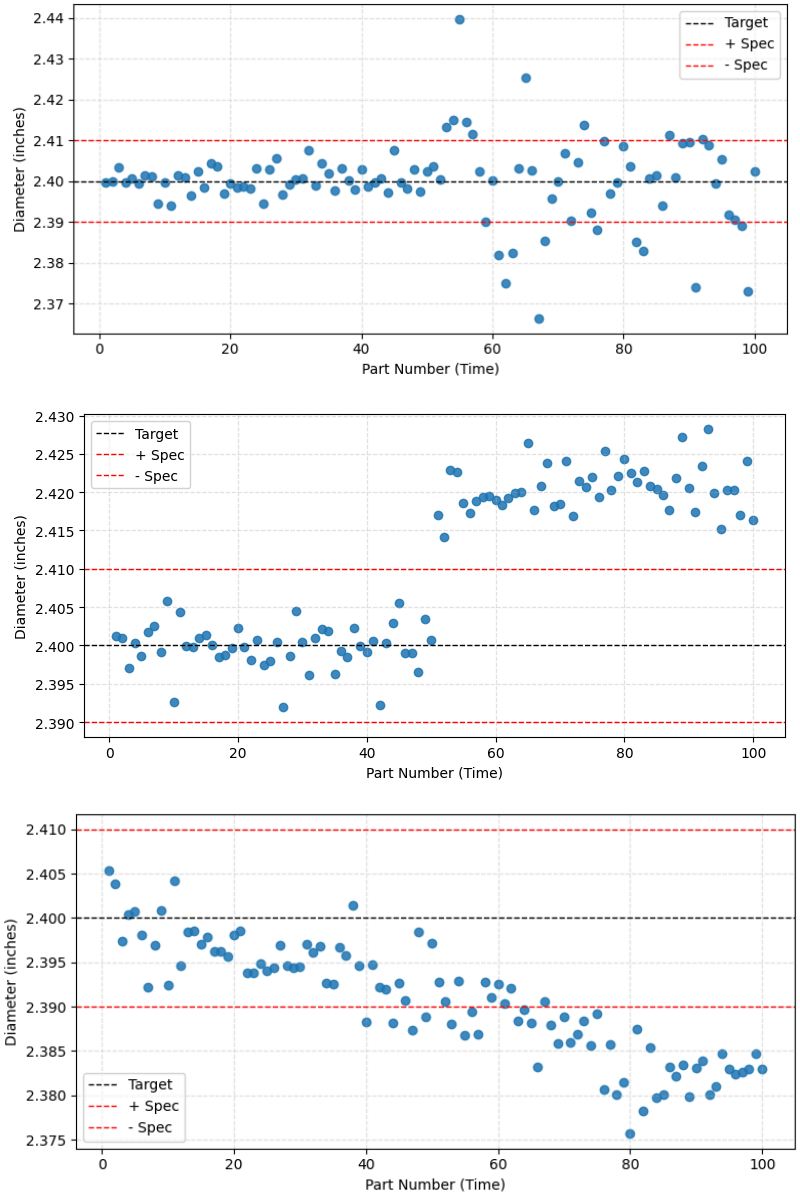


Figure 2: Example data sets generated by ChatGPT for the second year of the *Data Interpretation and Troubleshooting* oral assessment

While the datasets could be used directly (with titles cropped out), the instructor creating the assessment wanted to customize the charts. ChatGPT was able to provide code for the

scatterplots, which made this possible. The code was not perfect; it took the instructor about 1.5 hours to familiarize themselves with the code, debug it, and make custom graphs. The final code can be found in Appendix D. The instructor estimates this process would take about half that time (45 min) if they were to do it again in the future.

Overall, the teaching team benefited from using ChatGPT to fabricate datasets. Because datasets could be created and adjusted quickly, the teaching team was able to discuss different options and adapt the graphs to build consensus. They believe that the ability to rapidly iterate through ideas — and the conversations between colleagues elicited by these iterations — was a major advantage of the AI generated code. While ChatGPT occasionally created datasets that were not representative of reality and therefore couldn't be used in the assessment, it did create datasets that *looked* legitimate, helping to make the assessment more authentic to engineering practice. If an instructor is looking for a fast and easy way to create datasets and does not need control of the exact details, then this could be the perfect tool. However, if an instructor does want more control over all of the elements of the datasets, has a vision for exactly what they want beforehand, and is comfortable coding, it may be more reasonable to start with a python script.

Conclusion:

Here, we present two tools used by instructors in 2.008: Design and Manufacturing II at MIT to create efficient oral assessments in a mid-sized university manufacturing course: "Plug and Play" Style Assessment scripts and AI-generated datasets. We present these tools as well as additional notes to encourage others to implement short authentic oral assessments in their classes. Our goal is to help make this assessment method manageable for instructors with busy schedules and no formal training in alternative assessment.

Through the process of creating and implementing these oral assessments, the teaching team has formed stronger connections with students, built new skills, and had invaluable discussions about the connections between course components and engineering applications, and the tradeoffs surrounding authenticity and ambiguity in their class. We hope that by sharing our insights, we will inspire others to benefit similarly.

Acknowledgements:

We would like to thank the course lab instructors and teaching assistants for their help and feedback throughout the process of creating these assessments. We would also like to thank all of the practice test takers for volunteering their time to help make assessment better in 2.008. Lastly, we would like to thank all the students who participated in the assessments for being willing to try something new.

References:

[1] G. Wiggins, "The Case for Authentic Assessment," *Pract. Assess. Res. Eval.*, vol. 2, no. 1, Art. no. 1, Jan. 1990, doi: 10.7275/ffb1-mm19.

[2] M. Kim *et al.*, "Board 125: Work in Progress: Faculty Experiences and Learning Through Oral-Assessment Implementation in Engineering Courses," presented at the 2024 ASEE Annual Conference & Exposition, Jun. 2024. Accessed: Jan. 14, 2026. [Online]. Available: https://peer.asee.org/board-125-work-in-progress-faculty-experiences-and-learning-through-oral-assessment-implementation-in-engineering-courses

[3] V. Villarroel, D. Boud, S. Bloxham, D. Bruna, and C. Bruna, "Using principles of authentic assessment to redesign written examinations and tests," *Innov. Educ. Teach. Int.*, vol. 57, no. 1, pp. 38–49, Jan. 2020, doi: 10.1080/14703297.2018.1564882.

[4] S. Huffman, K. Becker, J. Liu, R. Zubajlo, and W. Seering, "Reflections on the implementation of Short, authentic oral assessments in a university manufacturing course," *Manuf. Lett.*, vol. 46, pp. 97–101, Dec. 2025, doi: 10.1016/j.mfglet.2025.10.011.

[5] S. B. Nolen, E. L. Michor, and M. D. Koretsky, "Engineers, figuring it out: Collaborative learning in cultural worlds," *J. Eng. Educ.*, vol. 113, no. 1, pp. 164–194, 2024, doi: 10.1002/jee.20576.

[6] M. D. Koretsky, C. J. McColley, J. L. Gugel, and T. W. Ekstedt, "Aligning classroom assessment with engineering practice: A design-based research study of a two-stage exam with authentic assessment," *J. Eng. Educ.*, vol. 111, no. 1, pp. 185–213, 2022, doi: 10.1002/jee.20436.

[7] R. A. Streveler, T. A. Litzinger, R. L. Miller, and P. S. Steif, "Learning conceptual knowledge in the engineering sciences: Overview and future research directions," *J. Eng. Educ.*, vol. 97, no. 3, pp. 279–294, Jul. 2008, doi: 10.1002/j.2168-9830.2008.tb00979.x.

[8] D. L. Schwartz and T. Martin, "Inventing to Prepare for Future Learning: The Hidden Efficiency of Encouraging Original Student Production in Statistics Instruction," *Cogn. Instr.*, vol. 22, no. 2, pp. 129–184, Jun. 2004, doi: 10.1207/s1532690xci2202_1.

[9] E. Wenger-Trayner, M. Fenton-O'Creevy, S. Hutchingson, C. Kubiak, and B. Wenger-Trayner, Eds., *Learning in Landscapes of Practice: Boundaries, identity, and knowledgeability in practice-based learning*. London: Routledge, 2014. doi: 10.4324/9781315777122.

[10] D. H. Jonassen, "Engineers as Problem Solvers," in *Cambridge Handbook of Engineering Education Research*, 1st ed., A. Johri and B. M. Olds, Eds., Cambridge University Press, 2014, pp. 103–118. doi: 10.1017/CBO9781139013451.009.

[11] B. Borges *et al.*, "Could ChatGPT get an engineering degree? Evaluating higher education vulnerability to AI assistants," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 121, no. 49, p. e2414955121, Dec. 2024, doi: 10.1073/pnas.2414955121.

[12] S. E. Brownell and K. D. Tanner, "Barriers to Faculty Pedagogical Change: Lack of Training, Time, Incentives, and…Tensions with Professional Identity?," *CBE—Life Sci. Educ.*, vol. 11, no. 4, pp. 339–346, Dec. 2012, doi: 10.1187/cbe.12-09-0163.

[13] D. W. Sunal *et al.*, "Teaching Science in Higher Education: Faculty Professional Development and Barriers to Change," *Sch. Sci. Math.*, vol. 101, no. 5, pp. 246–257, 2001, doi: 10.1111/j.1949-8594.2001.tb18027.x.

[14] S. W. Huffman, K. Becker, J. Liu, R. Zubajlo, and W. Seering, "Reflections on the Implementation of Short, Authentic Oral Assessments in a University Manufacturing Course," in *2025 ASEE Annual Conference & Exposition*, Montreal, Canada, Jun. 2025.

Appendix A: Rubric for Oral Assessment 1: *Design for Manufacture*

	5	4	3	2	1
CRQFS	Student **accurately and independently describes** the tradeoff between-the five factors (CRQFS) **with attention to detail and relevant nuances.**	Student **accurately describes** the tradeoff between the five factors.	Students can **describes some** tradeoffs between-the five factors (CRQFS) **without help from instructors.**	Student **struggles** to describe tradeoffs between the five factors (CRQFS), but can come to **limited** conclusions **with help from instructors.**	Student is **unable** to compare between the five factors (CRQFS) attributes, **even with significant help from instructors.**
DFM	Student **independently** explains the connections between design decisions and manufacturing processes, can **clearly** weigh tradeoffs, and uses **sound** engineering judgment to make **nuanced and explicit** suggestions.	Student explains some connections between design decisions and manufacturing processes, can weigh tradeoffs, and uses **some** engineering judgment to make **reasonable** suggestions, **with minimal help from instructors.**	Student explains the connections between design decisions and manufacturing processes, but has **difficulty weighing** trade offs or making suggestions **with some help from instructors.**	Student **struggles** to explain the connections between design decisions and manufacturing processes and has difficulty weighing tradeoffs. **Significant help from instructors is required.**	Student is **unable** to explain connections between design decisions and manufacturing processes and cannot weighing trade offs, **even with significant help from instructors.**
Processes & Parameters	Student **correctly** identifies the manufacturing process and influential process parameters, and **independently** makes **context-dependent** conclusions about the direction and degree of the effect of modifying interdependent parameters.	Student correctly identifies the manufacturing process and influential process parameters and makes **context-dependent** conclusions about the direction and degree of the effect of modifying interdependent parameters **with minimal help from instructors**.	Student is able to correct their process identification and identifies process and influential process parameters, but has **difficulty making** context-dependent conclusions about the direction or degree of the effect of modifying interdependent parameters **with some help** from instructors.	Student **struggles** to identify the process and influential process parameters or make context-dependent conclusions about the direction and degree of effect of modifying interdependent parameters. **Significant help from instructors is required.**	Student is **unable** to identify influential process parameters or make context-dependent conclusions about the direction or degree of the effect of modifying interdependent parameters, **even with significant help from instructors.**
Communication	Student communicates **clearly**, articulating precise concepts with **appropriate field specific** language to demonstrate understanding. Student may articulate gaps in their understanding if/when they get stuck, but are able to **independently** navigate a logical path forward. Student is on time to session.	Student communicates **clearly**, articulating concepts with field-specific language to demonstrate understanding. Student may articulate gaps in their understanding if/when they get stuck, but are able to navigate a logical path forward **with some help** from instructors. Student is on time to session.	Student articulates concepts with **some** field-specific language to demonstrate understanding. Student may articulate gaps in their understanding if/when they get stuck, but **struggle** to navigate a logical path forward, **even with help** from instructors. Student is late to session.	Student articulates concepts with **limited understanding**. Student may articulate gaps in their understanding if/when they get stuck, but are **unable** to navigate a logical path forward, even with help from instructors. Student is late to session.	Student **cannot** articulate concepts or demonstrate understanding. Student is late to session.

Appendix B: Rubric for Oral Assessment 2: *Data Interpretation and Troubleshooting*

	5	4	3	2	1
Data Interpretation	Student **precisely identifies** characteristics of a data set, **connects** those characteristics to the given **context**, and makes relevant and **specific conclusions** from data.	Student **accurately identifies** characteristics of a data set, **connects** those characteristics to the given **context**, and makes **relevant conclusions** from data with **minimal guidance.**	Student **identifies** characteristics a data set with some accuracy and can **connect** those characteristics to the given **context** and make **general conclusions** from data **with guidance.**	Student **identifies some** characteristics of a data set, but requires **significant guidance** to **connect** those characteristics to the given context and **guidance to make conclusions** from the data.	Student **is unable to identify** characteristics a data set and **unable apply it in context**. Student is **unable to make relevant conclusions** from the data with guidance.
Process Variation	Student is able to **logically identify potential** sources of variation **appropriate to the specific context**, and may be able to describe the **relative magnitude** of different effects.	Student is able to **identify potential** sources of variation **appropriate to the context** with minimal guidance.	Student is able to **identify potential** sources of variation **with some guidance.**	Student **struggles to identify** appropriate sources of variation **with significant guidance.**	Student **is unable identify** sources of variation, **even with guidance.**
Process Error & Problem Solving	Student can propose specific, appropriate solutions for characterization and mitigation of the source of variation and process defects.	Student can propose appropriate solutions for characterization and mitigation of the source of variation and process defects with minimal guidance.	Student can propose appropriate solutions for characterization and mitigation of the source of variation and process defects with some guidance.	Student struggles to propose appropriate solutions for characterization and mitigation of the source of variation and process defects with significant guidance.	Student is unable to propose appropriate solutions for characterization and mitigation of the source of variation and process defects, even with guidance.
Communication	Student communicates **clearly**, articulating precise concepts with **appropriate field specific** language to demonstrate understanding. Student may articulate gaps in their understanding if/when they get stuck, but are able to **independently** navigate a logical path forward.	Student communicates **clearly**, articulating concepts with field-specific language to demonstrate understanding. Student may articulate gaps in their understanding if/when they get stuck, but are able to navigate a logical path forward **with minimal guidance** from instructors.	Student articulates concepts with **some** field-specific language to demonstrate understanding. Student may articulate gaps in their understanding if/when they get stuck, but **struggle** to navigate a logical path forward, **even with guidance** from instructors.	Student articulates concepts with **limited understanding**. Student may articulate gaps in their understanding if/when they get stuck, but are **unable** to navigate a logical path forward, even with guidance from instructors.	Student **cannot** articulate concepts or demonstrate understanding.

Appendix C: Complete ChatGPT Conversation

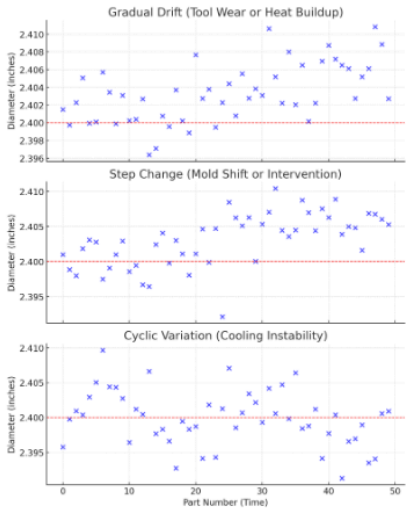
Hello- I am working on creating assessments for a university manufacturing class and would like help generating fabricated data sets. The students are going to be analyzing a production run of an injection molded yo-yo. Specifically, I would like them to analyze a scatter plot of diameters from a run (50-100 parts) of yo-yo rings. I would like several data sets for variety. In each data set, there should be a change over time. The students will be asked to troubleshoot what might have happened in the manufacturing process to cause this change. I want to go back and forth with you to make sure everything is how I imagine it. Please ask if you have any clarifying questions, otherwise, please start with 3 scatter plots.

Excellent project — this sounds like a great way to get students thinking diagnostically and tying data to process understanding. Before I generate the three data sets and scatter plots, I have a few clarifying questions to make sure I match your vision closely:

1. **Nominal size & tolerance:**
   What's the target diameter of the yo-yo ring, and what's a reasonable process tolerance (e.g., 56.00 mm ± 0.05 mm)?
2. **Measurement resolution:**
   Should the data look like real micrometer readings (e.g., 55.97, 56.03 mm), or do you prefer rounded values (e.g., 55.9, 56.0 mm)?
3. **Nature of "change over time":**
   Should the change reflect:
   - gradual **drift** (e.g., tool wear, temperature rise),
   - **step shift** (e.g., mold misalignment after an intervention),
   - **cyclic variation** (e.g., unstable temperature control),
   - or a mix of these across the three data sets?
4. **Number of parts per data set:**
   Do you prefer around 50 or closer to 100 measurements for each?
5. **Data presentation format:**
   Would you like the initial results as scatter plots (time vs. diameter) only, or should I also provide the underlying numerical data in a table or CSV?

Once I have those details, I'll generate three distinct data sets (and plots) that represent plausible manufacturing issues your students could diagnose.

Great! Answers are as follows:
1) the target diameter should be around 2.4 inches; standard deviation within 0.003 inches
2) The data should be real-looking readings down to three decimal places.
3) there should be a mix of these across the data sets. I will eventually ask you for more than 3 data sets with more possible reasonings
4) Lets go closer to 50 to start. I want to see how it looks and then see where we go from there.
5) scatter plots is fine for now



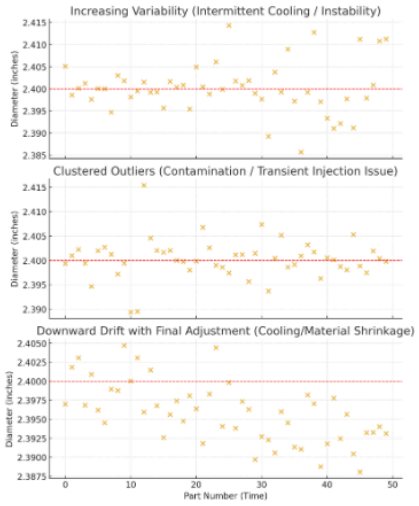
Here are the first three fabricated data sets represented as scatter plots:

1. **Gradual Drift:** Diameters slowly increase across the run, as might happen with tool wear or rising mold temperature.
2. **Step Change:** A distinct upward shift appears halfway through the run — possibly due to a mold misalignment or process intervention.
3. **Cyclic Variation:** The diameters oscillate periodically, suggesting instability in the cooling or pressure control.

Would you like to see the underlying numerical data next (e.g., tables or CSV), or should we move on to designing a few more scenarios with other types of manufacturing issues (e.g., shrinkage, short shot, material batch variation)? ‹/›

sure - These look pretty good! Lets design a few more!

Thought for 8s



Nice — three more datasets added:

4. **Increasing variability** (instability in cooling or intermittent process control).
5. **Clustered outliers** (contamination, transient injection problems, or a clogged gate causing flash).
6. **Downward drift with final adjustment** (material shrinkage or cooling setpoint change, then operator tweak).

If you'd like next steps, I can:

- Export any of these datasets as CSVs (I can provide download links),
- Show the raw numeric tables (with three-decimal readings),
- Create more scenarios (e.g., bimodal shifts, tool chatter, gradual bias plus cyclic noise), or
- Draft prompt/questions for the students to guide their troubleshooting.

Which of those would you like me to do next? ‹/›

lets do a tool chatter one! Could we also do one where there is more than one machine (indicated by different shaped dots) where the machines could have the same settings, but be preforming slightly differently? I'm thinking both machines on the same graph, about 50 data points for each.

Here you go — one **tool-chatter dataset** and one **two-machine comparison dataset**.

✅ Tool Chatter

The variation shows a high-frequency oscillation on top of normal injection molding variation — the kind of pattern you'd expect from:

- worn bearings
- a vibrating mold half
- inconsistent clamp force
- loose tie bars

✅ Two-Machine Comparison

Both machines nominally run "the same process," but they show:

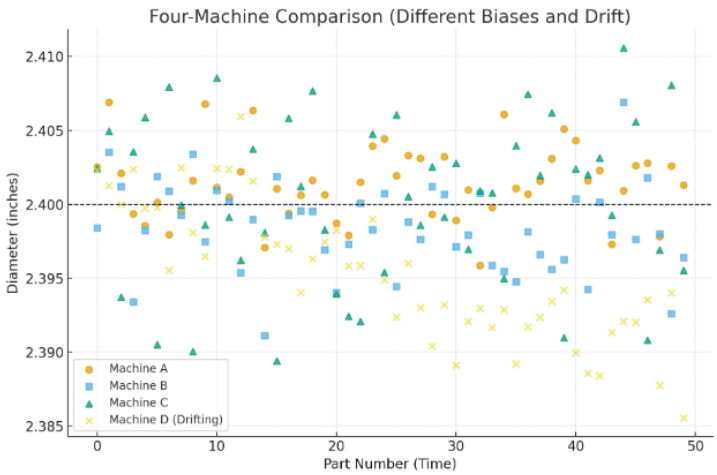
- Machine A biased *slightly high*
- Machine B biased *slightly low*
  This can lead to rich troubleshooting discussions about calibration, machine-to-machine variation, and mold–machine matching.

If you'd like, I can also:

- Generate the raw numeric data for either set
- Put both machines into **different colors AND different marker shapes** (right now shapes differ, colors are defaults)
- Add more machines (3–4 lines)
- Add an interaction effect (e.g., one machine drifts while the other stays stable)
- Create a student worksheet or assessment prompt

What would you like next? ‹/›

lets keep playing with the machines- can we add 2 more machines, and have one of them drift?



Here's the **four-machine comparison** you described:

- **Machine A:** Slightly high bias (running a bit large).
- **Machine B:** Slightly low bias (running a bit small).
- **Machine C:** Centered but with higher variability (older press or less consistent temperature control).
- **Machine D:** Gradual downward drift (e.g., cooling water slowly warming, or mold shrinkage increasing).

This gives a realistic snapshot of what a multi-machine process study might look like — great for discussions about process capability, matching tools to presses, and maintenance diagnostics.

Would you like the next version to show **each machine's trend line** (to make drift and bias visually clearer), or keep the raw scatter for student interpretation? ‹/›

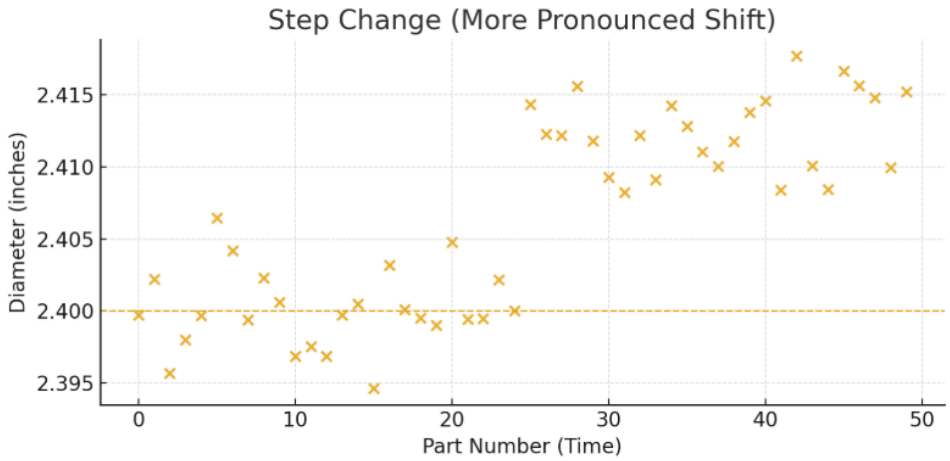
these are great for now! I want to check in with my colleagues and get back to you!

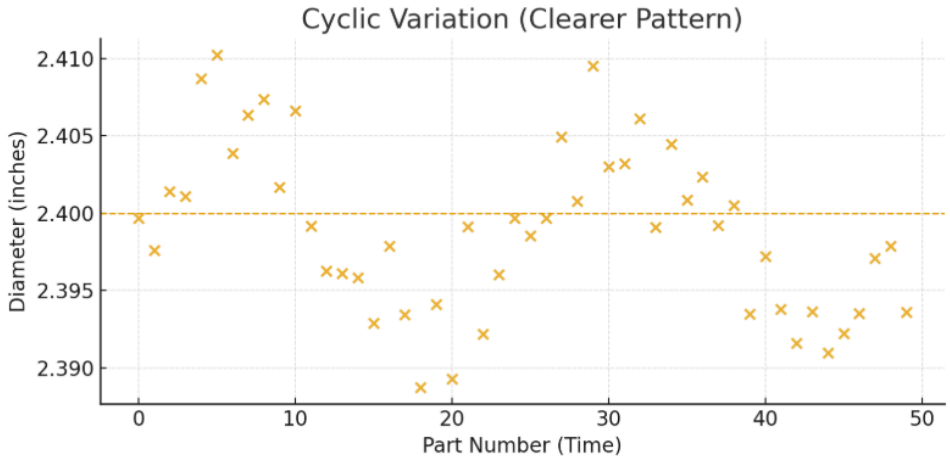
Perfect — I'm glad they're hitting the mark! 👍
When you're ready, we can easily:

- tweak ranges, variation, or visual style,
- add contextual info (like machine parameters or material batches), or
- build student-facing assessment prompts and answer keys around these plots.

I'll be here when you and your colleagues are ready to pick it back up.

Hello - I want to come back and try some more iterations. It seems like some of the graphs may be too hard to interpret for students. Could we try another iteration of the Step Change graph and the Cyclic Variation Graph? I still want variation, but it should be slightly more obvious. It is ok to go out of spec to get a slightly clearer graph



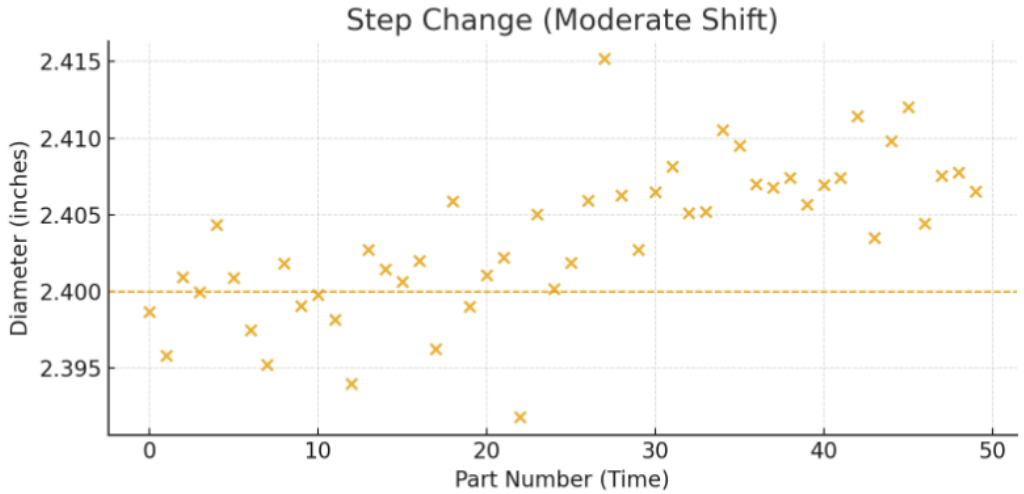


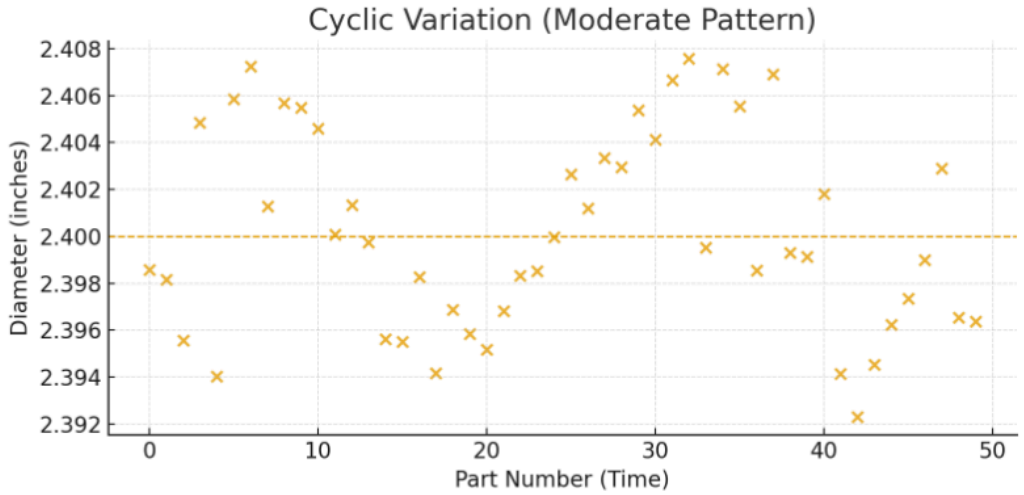
Here are clearer versions of your two datasets:

- **Step Change:** The shift is now about **0.012 inches**, making it much easier for students to recognize the process change (e.g., mold misalignment or operator adjustment).
- **Cyclic Variation:** The amplitude of variation has been increased to roughly **±0.007 inches**, so the oscillation stands out clearly — great for diagnosing something like cooling or pressure cycling.

Would you like these to include an "**out of spec" band** (upper/lower tolerance lines) for visual reference? It could help students connect the data patterns to quality implications.

those are more pronounced! I think we've gone a little too far. Can you do something in the middle?





Here's the **moderate version** of both datasets — the variations are now noticeable but not exaggerated:

- **Step Change:** A subtler shift of about **0.007 inches**, enough for students to detect a process change without being visually overwhelming.
- **Cyclic Variation:** The sine pattern now has about **±0.0045 inches** of swing, giving a clear but realistic repeating variation.

Would you like me to overlay **spec limits** (e.g., ±0.005 or ±0.01) on these to make them classroom-ready visuals? ‹/›

I think the format of these is ok. I'm wondering if getting closer to 100 pts instead of 50 would help. These are looking better

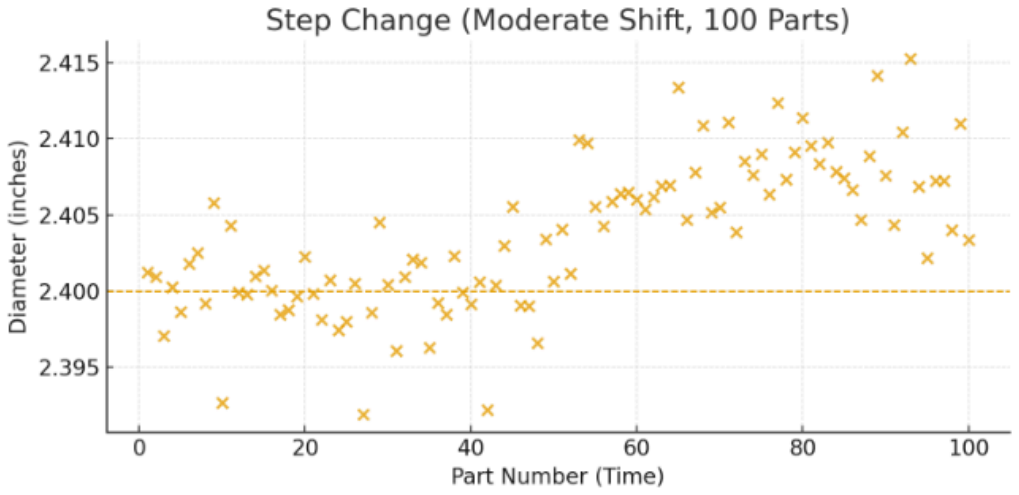
That's a great instinct — going to around **100 parts** will smooth out the scatter and make the trends easier to see without exaggerating the magnitude. It gives students more context for diagnosing gradual or repeating changes.

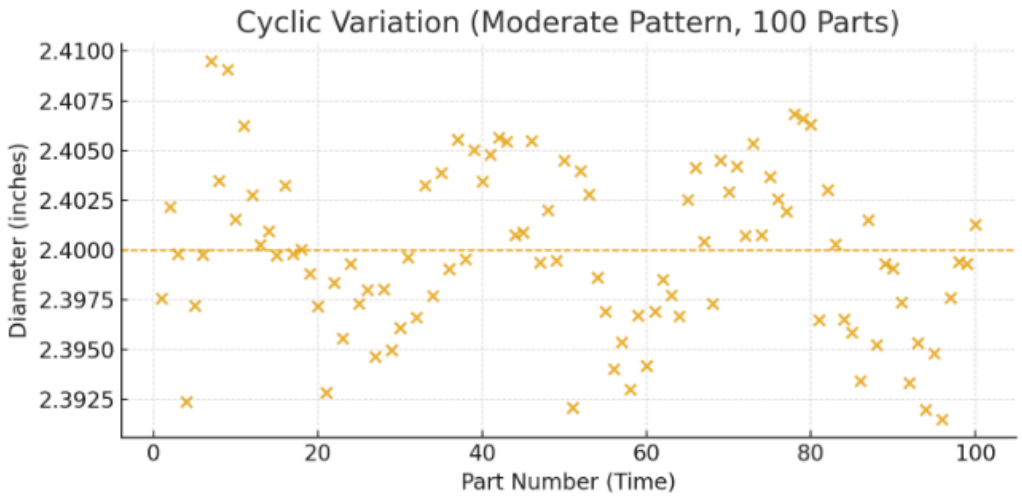
Before I regenerate them:

- Would you like me to **keep the same degree of variation** (the moderate ones we just made), or slightly tone them up/down now that you'll have more points?
- Should we assume **100 sequential parts** from a single run (so the x-axis is part number 1–100)?
- And do you want me to **keep the same format** (simple scatter, target line, no spec limits yet)?

Once you confirm those, I'll make the updated 100-point versions of the **Step Change** and **Cyclic Variation** graphs.

Thank you for asking for clarification. Lets keep about the same degree of variation, do 100 sequential parts on a single run, and keep the same format.





Here are your **100-part versions** — same moderate variation, now with smoother and more interpretable patterns:

- **Step Change**: A clean mid-run jump of about **0.007 inches**, easy to spot without looking artificial.
- **Cyclic Variation**: A realistic oscillation (≈ ±0.0045 in) that repeats several times through the run.

These should be great for student analysis. Would you like me to generate the underlying numeric data (for worksheets or analysis activities) next? ‹/›

these are great! Thank you! I'll be back tomorrow when my graph limit resets for more sets!

Perfect — I'm really glad these hit the sweet spot! 🎯
When you're back, we can easily spin up new scenarios (like drift, instability, or multi-machine comparisons) with matching clarity levels.

Have a great day — and good luck with the colleague review!

Hi - Actually, We want to make some more variations on these graphs ourselves so that we can move more quickly. Can you write some python code that we can use? We want to be able to control the number of points and be able to make several versions of similar graphs for: (1)drift up, (2)drift down, (3)increased variability, (4)decreased variability, (5)step up, (6)step down, (7)cyclic variation, and (8)two-machine comparison (with any number of problems). Can you help with that?

Absolutely — that's a great idea.
Here's a **flexible Python script** you can run locally to generate as many datasets and scatter plots as you'd like. It includes simple functions for each scenario (drift up/down, variability change, step change, cyclic variation, and multi-machine comparison).

You'll be able to set:

- the number of points
- the nominal target and standard deviation
- the magnitude of drift, step, or cycle
- the number of machines for comparison

```
import numpy as np
import matplotlib.pyplot as plt

#
============================================================
# General parameters
#
============================================================
target = 2.4
base_sd = 0.003

#
============================================================
# Helper functions to generate datasets
#
============================================================

def drift_up(n=50, target=2.4, sd=0.003,
drift_amount=0.008):
```

```
    """Gradual upward drift (e.g., heat buildup or tool
wear)."""
    drift = np.linspace(0, drift_amount, n)
    return np.arange(1, n+1), np.random.normal(target, sd,
n) + drift

def drift_down(n=50, target=2.4, sd=0.003,
drift_amount=0.008):
    """Gradual downward drift (e.g., cooling issues)."""
    drift = np.linspace(0, -drift_amount, n)
    return np.arange(1, n+1), np.random.normal(target, sd,
n) + drift

def increased_variability(n=50, target=2.4, sd=0.003,
factor=2.0):
    """Process grows more variable halfway through."""
    data = np.random.normal(target, sd, n)
    for i in range(n//2, n):
        data[i] = np.random.normal(target, sd * factor)
    return np.arange(1, n+1), data

def decreased_variability(n=50, target=2.4, sd=0.003,
factor=0.5):
    """Process becomes more stable halfway through."""
    data = np.random.normal(target, sd * factor, n)
    for i in range(n//2):
        data[i] = np.random.normal(target, sd)
    return np.arange(1, n+1), data

def step_up(n=50, target=2.4, sd=0.003, step_size=0.007):
    """Sudden step increase halfway through."""
    data = np.random.normal(target, sd, n)
    data[n//2:] += step_size
    return np.arange(1, n+1), data

def step_down(n=50, target=2.4, sd=0.003, step_size=0.007):
    """Sudden step decrease halfway through."""
    data = np.random.normal(target, sd, n)
    data[n//2:] -= step_size
    return np.arange(1, n+1), data
```

```
def cyclic_variation(n=50, target=2.4, sd=0.003,
amplitude=0.004, cycles=3):
    """Cyclic pattern (e.g., unstable cooling or
pressure)."""
    pattern = amplitude * np.sin(np.linspace(0, cycles *
np.pi, n))
    return np.arange(1, n+1), np.random.normal(target, sd,
n) + pattern

def two_machine_comparison(n=50, target=2.4, sd=0.003,
num_machines=2, issues=None):
    """
    Generate data for multiple machines.
    'issues' can be a list of strings like ['drift_down',
'step_up'] to apply to each machine.
    """
    if issues is None:
        issues = ['normal'] * num_machines

    time = np.arange(1, n+1)
    datasets = []

    for i, issue in enumerate(issues):
        if issue == 'drift_up':
            _, data = drift_up(n, target, sd)
        elif issue == 'drift_down':
            _, data = drift_down(n, target, sd)
        elif issue == 'step_up':
            _, data = step_up(n, target, sd)
        elif issue == 'step_down':
            _, data = step_down(n, target, sd)
        elif issue == 'cyclic':
            _, data = cyclic_variation(n, target, sd)
        else:
            data = np.random.normal(target, sd, n)
        datasets.append((f"Machine {i+1} ({issue})", data))

    return time, datasets

#
============================================================
```

```
# Plotting function
#
============================================================

def plot_data(x, y, title="Dataset", target=2.4):
    plt.figure(figsize=(8, 4))
    plt.scatter(x, y, alpha=0.8)
    plt.axhline(target, linestyle='--', linewidth=1,
color='black')
    plt.title(title)
    plt.xlabel("Part Number (Time)")
    plt.ylabel("Diameter (inches)")
    plt.grid(True, linestyle='--', alpha=0.4)
    plt.tight_layout()
    plt.show()

def plot_two_machine(time, datasets, target=2.4):
    plt.figure(figsize=(9, 5))
    markers = ['o', 's', '^', 'x', 'D', 'v']
    for i, (label, data) in enumerate(datasets):
        plt.scatter(time, data, alpha=0.8, marker=markers[i
% len(markers)], label=label)
    plt.axhline(target, linestyle='--', linewidth=1,
color='black')
    plt.title("Two (or More) Machine Comparison")
    plt.xlabel("Part Number (Time)")
    plt.ylabel("Diameter (inches)")
    plt.legend()
    plt.grid(True, linestyle='--', alpha=0.4)
    plt.tight_layout()
    plt.show()

#
============================================================
# Example usage
#
============================================================
if __name__ == "__main__":
    # Example 1: Step Up
    x, y = step_up(n=100)
    plot_data(x, y, "Step Change Up (Example)")
```

```
    # Example 2: Cyclic Variation
    x, y = cyclic_variation(n=100)
    plot_data(x, y, "Cyclic Variation (Example)")

    # Example 3: Two Machine Comparison with Issues
    time, machines = two_machine_comparison(n=100,
num_machines=3, issues=['normal', 'drift_down', 'step_up'])
    plot_two_machine(time, machines)
```

💡 How to use

1. Save this as `yo_yo_datasets.py`.
2. Run it in Python (e.g., VS Code, Jupyter, or directly in the terminal).
3. Adjust parameters inside each function call to make your own variants:
   - `n` → number of points
   - `step_size`, `drift_amount`, `amplitude`, etc. → control the magnitude of change
   - Add new "issues" to the list for multi-machine comparisons

Would you like me to also include an option for **spec limits (±0.005)** as a toggle in the plots? That can make it classroom-ready out of the box.

sure lets try that - I want to make sure it is easy to toggle off though. Thank you

Perfect — here's an updated version of the Python script with an **easy on/off toggle for spec limits**.
When `show_spec=True`, it will draw **upper and lower spec limit lines** around the target (you can set the width, e.g. `spec_tol=0.005`).
If you set `show_spec=False`, they disappear — no other edits needed.

```
import numpy as np
import matplotlib.pyplot as plt

#
============================================================
# General parameters
```

```
#
============================================================
target = 2.4
base_sd = 0.003

#
============================================================
# Helper functions to generate datasets
#
============================================================

def drift_up(n=50, target=2.4, sd=0.003,
drift_amount=0.008):
    drift = np.linspace(0, drift_amount, n)
    return np.arange(1, n+1), np.random.normal(target, sd,
n) + drift

def drift_down(n=50, target=2.4, sd=0.003,
drift_amount=0.008):
    drift = np.linspace(0, -drift_amount, n)
    return np.arange(1, n+1), np.random.normal(target, sd,
n) + drift

def increased_variability(n=50, target=2.4, sd=0.003,
factor=2.0):
    data = np.random.normal(target, sd, n)
    for i in range(n//2, n):
        data[i] = np.random.normal(target, sd * factor)
    return np.arange(1, n+1), data

def decreased_variability(n=50, target=2.4, sd=0.003,
factor=0.5):
    data = np.random.normal(target, sd * factor, n)
    for i in range(n//2):
        data[i] = np.random.normal(target, sd)
    return np.arange(1, n+1), data

def step_up(n=50, target=2.4, sd=0.003, step_size=0.007):
    data = np.random.normal(target, sd, n)
    data[n//2:] += step_size
    return np.arange(1, n+1), data
```

```
def step_down(n=50, target=2.4, sd=0.003, step_size=0.007):
    data = np.random.normal(target, sd, n)
    data[n//2:] -= step_size
    return np.arange(1, n+1), data

def cyclic_variation(n=50, target=2.4, sd=0.003,
amplitude=0.004, cycles=3):
    pattern = amplitude * np.sin(np.linspace(0, cycles *
np.pi, n))
    return np.arange(1, n+1), np.random.normal(target, sd,
n) + pattern

def two_machine_comparison(n=50, target=2.4, sd=0.003,
num_machines=2, issues=None):
    if issues is None:
        issues = ['normal'] * num_machines
    time = np.arange(1, n+1)
    datasets = []
    for i, issue in enumerate(issues):
        if issue == 'drift_up':
            _, data = drift_up(n, target, sd)
        elif issue == 'drift_down':
            _, data = drift_down(n, target, sd)
        elif issue == 'step_up':
            _, data = step_up(n, target, sd)
        elif issue == 'step_down':
            _, data = step_down(n, target, sd)
        elif issue == 'cyclic':
            _, data = cyclic_variation(n, target, sd)
        else:
            data = np.random.normal(target, sd, n)
        datasets.append((f"Machine {i+1} ({issue})", data))
    return time, datasets

#
============================================================
# Plotting functions
#
============================================================
```

```
def plot_data(x, y, title="Dataset", target=2.4,
show_spec=False, spec_tol=0.005):
    plt.figure(figsize=(8, 4))
    plt.scatter(x, y, alpha=0.8)
    plt.axhline(target, linestyle='--', linewidth=1,
color='black', label="Target")
    if show_spec:
        plt.axhline(target + spec_tol, color='red',
linestyle='--', linewidth=1, label="+ Spec Limit")
        plt.axhline(target - spec_tol, color='red',
linestyle='--', linewidth=1, label="- Spec Limit")
    plt.title(title)
    plt.xlabel("Part Number (Time)")
    plt.ylabel("Diameter (inches)")
    plt.grid(True, linestyle='--', alpha=0.4)
    if show_spec:
        plt.legend()
    plt.tight_layout()
    plt.show()

def plot_two_machine(time, datasets, target=2.4,
show_spec=False, spec_tol=0.005):
    plt.figure(figsize=(9, 5))
    markers = ['o', 's', '^', 'x', 'D', 'v']
    for i, (label, data) in enumerate(datasets):
        plt.scatter(time, data, alpha=0.8, marker=markers[i
% len(markers)], label=label)
    plt.axhline(target, linestyle='--', linewidth=1,
color='black')
    if show_spec:
        plt.axhline(target + spec_tol, color='red',
linestyle='--', linewidth=1)
        plt.axhline(target - spec_tol, color='red',
linestyle='--', linewidth=1)
    plt.title("Two (or More) Machine Comparison")
    plt.xlabel("Part Number (Time)")
    plt.ylabel("Diameter (inches)")
    plt.legend()
    plt.grid(True, linestyle='--', alpha=0.4)
    plt.tight_layout()
    plt.show()
```

```
#
============================================================
# Example usage
#
============================================================
if __name__ == "__main__":
    # Example 1: Step Up with spec limits
    x, y = step_up(n=100)
    plot_data(x, y, "Step Change Up (Example)",
show_spec=True, spec_tol=0.005)

    # Example 2: Cyclic Variation without spec limits
    x, y = cyclic_variation(n=100)
    plot_data(x, y, "Cyclic Variation (Example)",
show_spec=False)

    # Example 3: Multi-machine comparison with spec limits
    time, machines = two_machine_comparison(
        n=100,
        num_machines=3,
        issues=['normal', 'drift_down', 'step_up']
    )
    plot_two_machine(time, machines, show_spec=True,
spec_tol=0.005)
```

Appendix D: Finalized Dataset Generation Code

```
import numpy as np
import matplotlib.pyplot as plt

# ============================================================
# Defaults (tweak here or per-call)
# ============================================================
DEFAULT_TARGET = 2.4
DEFAULT_SD = 0.003   # base process sigma

# Optional: set a seed for reproducibility from your script/notebook.
def set_seed(seed: int | None = None):
    np.random.seed(seed)

# ============================================================
# Plotting helper (with easy spec-limit toggle)
# ============================================================
def plot_data(x, y, title="Dataset", target=DEFAULT_TARGET, show_spec=False,
spec_tol=0.005):
    plt.figure(figsize=(8, 4))
    plt.scatter(x, y, alpha=0.85)
    plt.axhline(target, linestyle='--', linewidth=1, color='black',
label="Target")
    if show_spec:
        plt.axhline(target + spec_tol, color='red', linestyle='--',
linewidth=1, label="+ Spec")
        plt.axhline(target - spec_tol, color='red', linestyle='--',
linewidth=1, label="- Spec")
        plt.legend()
    plt.title(title)
    plt.xlabel("Part Number (Time)")
    plt.ylabel("Diameter (inches)")
    plt.grid(True, linestyle='--', alpha=0.35)
    plt.tight_layout()
    plt.show()

# ============================================================
# Core generators (all include `factor` to scale variability)
# x always = np.arange(1, n+1)
# ============================================================

def drift_up(
```

```
    n=50,
    target=DEFAULT_TARGET,
    sd=DEFAULT_SD,
    factor=1.0,
    drift_amount=0.008
):
    """
    Gradual upward drift across the run.
    factor: multiplies the base sd for all points.
    drift_amount: total change from first to last point (inches).
    """
    x = np.arange(1, n+1)
    noise = np.random.normal(loc=0.0, scale=sd * factor, size=n)
    drift = np.linspace(0.0, drift_amount, n)
    y = target + noise + drift
    return x, y

def drift_down(
    n=50,
    target=DEFAULT_TARGET,
    sd=DEFAULT_SD,
    factor=1.0,
    drift_amount=0.008
):
    """
    Gradual downward drift across the run.
    """
    x = np.arange(1, n+1)
    noise = np.random.normal(loc=0.0, scale=sd * factor, size=n)
    drift = np.linspace(0.0, -drift_amount, n)
    y = target + noise + drift
    return x, y

def increased_variability(
    n=50,
    target=DEFAULT_TARGET,
    sd=DEFAULT_SD,
    factor=1.0,
    change_factor=2.0,
    change_at: int | None = None
):
    """
```

```
    Variability increases mid-run.
    factor: base variability multiplier before the change.
    change_factor: additional multiplier applied AFTER change point.
    change_at: index (1-based in plot; 0-based internally) where variability
increases.
               Defaults to halfway (n//2).
    """
    if change_at is None:
        change_at = n // 2
    x = np.arange(1, n+1)
    y = np.empty(n, dtype=float)
    # First segment
    y[:change_at] = np.random.normal(loc=target, scale=sd * factor,
size=change_at)
    # Second segment
    y[change_at:] = np.random.normal(loc=target, scale=sd * factor *
change_factor, size=n - change_at)
    return x, y

def decreased_variability(
    n=50,
    target=DEFAULT_TARGET,
    sd=DEFAULT_SD,
    factor=1.0,
    change_factor=0.5,
    change_at: int | None = None
):
    """
    Variability decreases mid-run.
    factor: base variability multiplier before the change.
    change_factor: multiplier AFTER change point (e.g., 0.5 halves
variability).
    """
    if change_at is None:
        change_at = n // 2
    x = np.arange(1, n+1)
    y = np.empty(n, dtype=float)
    y[:change_at] = np.random.normal(loc=target, scale=sd * factor,
size=change_at)
    y[change_at:] = np.random.normal(loc=target, scale=sd * factor *
change_factor, size=n - change_at)
    return x, y
```

```
def step_up(
    n=50,
    target=DEFAULT_TARGET,
    sd=DEFAULT_SD,
    factor=1.0,
    step_size=0.007,
    step_at: int | None = None
):
    """
    Sudden upward shift mid-run.
    factor: scales noise both before and after the step.
    step_at: index where step happens (default halfway).
    """
    if step_at is None:
        step_at = n // 2
    x = np.arange(1, n+1)
    y = np.random.normal(loc=target, scale=sd * factor, size=n)
    y[step_at:] += step_size
    return x, y

def step_down(
    n=50,
    target=DEFAULT_TARGET,
    sd=DEFAULT_SD,
    factor=1.0,
    step_size=0.007,
    step_at: int | None = None
):
    """
    Sudden downward shift mid-run.
    """
    if step_at is None:
        step_at = n // 2
    x = np.arange(1, n+1)
    y = np.random.normal(loc=target, scale=sd * factor, size=n)
    y[step_at:] -= step_size
    return x, y

def cyclic_variation(
    n=50,
    target=DEFAULT_TARGET,
```

```
    sd=DEFAULT_SD,
    factor=1.0,
    amplitude=0.004,
    cycles=3.0,
    phase=0.0
):
    """
    Repeating sine-wave pattern superimposed on noise.
    factor: scales overall noise.
    amplitude: sine amplitude in inches.
    cycles: number of half-cycles of π (i.e., sin goes 0..cycles*π).
            If you prefer full cycles across n, pass cycles=2*np.pi*k and set
linspace accordingly.
    phase: radians phase offset (useful for generating variants).
    """
    x = np.arange(1, n+1)
    theta = np.linspace(0.0 + phase, cycles * np.pi + phase, n)
    pattern = amplitude * np.sin(theta)
    noise = np.random.normal(loc=0.0, scale=sd * factor, size=n)
    y = target + pattern + noise
    return x, y

# ============================================================
# Quick examples (delete or adapt in your notebook/script)
# ============================================================
if __name__ == "__main__":
    set_seed(2027)

    # Step up
    x1, y1 = step_up(n=100, factor=1.0, step_size=0.02)
    plot_data(x1, y1, "", show_spec=True, spec_tol=0.01)

    # Cyclic
    x2, y2 = cyclic_variation(n=500, factor=1.5, amplitude=0.008, cycles=4)
    plot_data(x2, y2, "", show_spec=True, spec_tol=0.01)

    # Increased variability
    x3, y3 = increased_variability(n=100, factor=1.0, change_factor=5.0)
    plot_data(x3, y3, "", show_spec=True, spec_tol=0.01)

    # Drift up
    x4, y4 = drift_up(n=100, factor=1.0, drift_amount=0.02)
```

```
plot_data(x4, y4, "", show_spec=True, spec_tol=0.01)

# Drift down
x5, y5 = drift_down(n=100, factor=1.0, drift_amount=0.02)
plot_data(x5, y5, "", show_spec=True, spec_tol=0.01)

# Step down
x6, y6 = step_down(n=100, factor=1.0, step_size=0.02)
plot_data(x6, y6, "", show_spec=True, spec_tol=0.01)
```