INTERACTIVE SKETCH INPUT TOOL

M. SEGADO AND J. FRENCH

Why sketching. One of the goals of Introductory Calculus is for students to develop fluency in representing functions both notationally and graphically. The latter presents a particular challenge in online courses, where textual or multiple-choice assessment items are the norm. To develop representational fluency in mathematics in such an online context, a tool that assesses student generated sketches of graphs online is essential [1]. [1]

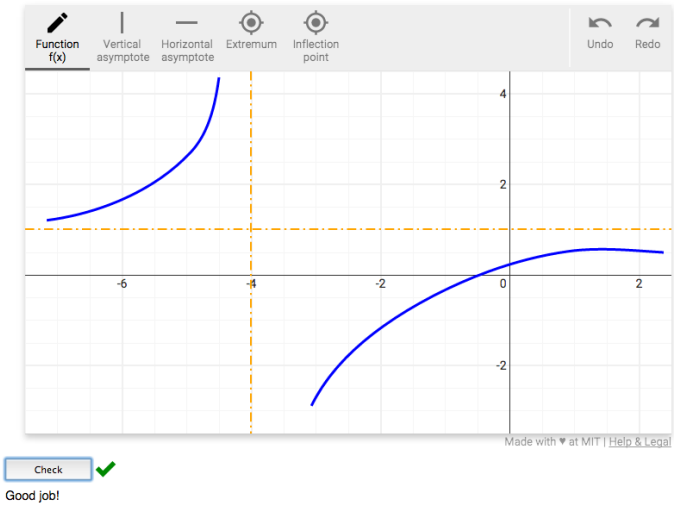
In order to address these shortcomings, we have developed an extensible, web-based "sketch input" tool along with a customizable python library of grading functions tailored to calculus learning objectives. This tool was used for the first time in Spring 2015 for the MIT course 18.01, Calculus I, and is currently being used by 18.01.1x, a Calculus MOOC on the edX platform with 21,000 registered students. The benefits of this tool are two-fold:

(1) Cognitive engagement is higher when students are actively constructing a sketch of a curve rather than passively identifying a graph [3].
(2) Automatic grading allows us to provide immediate feedback, which is a fundamental mechanism for promoting learning [4, 5, 6]. Immediate feedback allows us to target and correct known student misconceptions [7].

Due to the modular nature of the tool, we expect that it will also be extended to assessments in fields other than mathematics; at MIT, we currently have interest from the electrical engineering and computer science department, physics department, and mechanical engineering department who are interested in using the tool for both on-campus students and global MOOCs. It is also possible that such a tool could be used as a framework for more domain-specific sketch grading and shape-recognition systems (e.g., the desktop Mechanix software [2], or its budding successor, CourseSketch).

Designing the sketch input tool. At a high level, the sketch input tool consists of a student-facing web application and a collection of server-side processing and grading code. The student-facing portion, which is embedded in a course website, provides a set of semantically-meaningful drawing tools which may be customized for each problem. The data from student sketches is passed to the server for assessment, where problem-specific grading scripts make use of the provided library of grading functions to assess the correctness of the student graph and provide immediate feedback or hints based on the student's input. These functions handle the details of parsing student drawings data and expose a variety of functions that test slopes, curvature, and values, with built-in tolerances informed by actual student data.

The student-facing frontend of the tool draws inspiration from existing drawing and editing software (Adobe Creative Suite, Microsoft Office, and their ilk), but gives considerable weight to the semantics of what is being drawn, not just its appearance or the particular tools used to draw it; for example, "Function" or "Derivative" are used in toolbar labels rather than just "Pencil". This semantic labeling is configurable by the instructor for each problem (along with other details such as the axis scaling, labeling, and choice of drawing plugins) and emphasizes the language used in the course. Labeling of features such as critical points, inflection points, and asymptotes allows the grader scripts to further interpret student intent and identify inconsistencies and errors in the graph. A screenshot of the tool being used to sketch a rational function is shown on the right.



Date: July 20, 2015.

[1]Standardized Calculus assessments, such as the AP and IB exams, do assess student generated sketches of graphs, making assessments of student generated content a reasonable expectation of any AP or IB level calculus class.

To operate reliably on the wide range of devices used to access MOOCs, we required a freeform drawing mechanism that provides some smoothing to noisy student input from computer mice, trackpads, and touchscreens. This smoothing capability makes the tool less frustrating to use and considerably improves the aesthetic appeal of student sketches. An initial pass of real-time smoothing is achieved by modeling the pen as a rigid string dragging the drawing tool; further smoothing is applied when a line segment is complete by fitting a piecewise cubic spline to the data [8] which also has attributes that can be exploited in grading algorithms.

Modularity, data generation and iteration. In order to enable a wider range of applications, the sketch input tool is based around a modular architecture, with a JSON-based, human-readable data format used by all drawing plugins and grading libraries. Student data may be manipulated directly in this format, or rendered into interactive vector composites displaying hundreds or thousands of responses at once. We can use student generated input to refine our grading algorithms, developing an automated testing suite for the functionality that we desire in our grader. There is also a huge potential for research on these data as we can collect not only what the students draw, but also the order in which they draw the items on a graph and what their various attempts and corrections look like.

Grading student graphs. Assessments for these types of problems involve 3 typical types of problems:

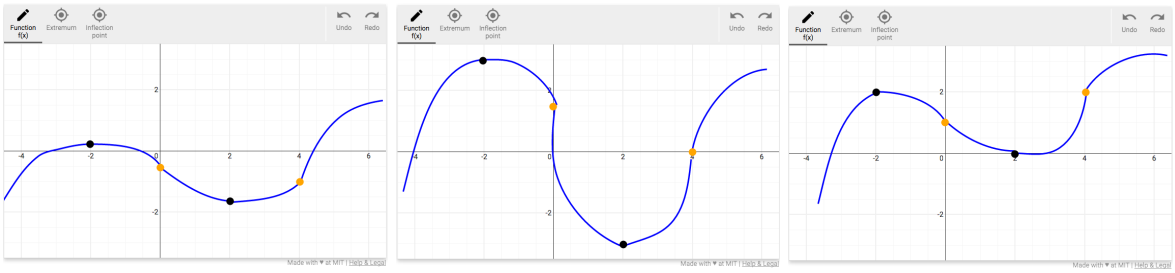
(1) Given a function data, sketch the graph of the function.
(2) Given a sketch of the function, sketch its derivative function.
(3) Given the derivative of a function (graphical or mathematical), sketch the function.

Sketching of curves in a calculus class is based on students developing a qualitative sense of the behavior of functions based on local behavior of derivatives rather than plotting points for an "exact" solution. The qualitative nature requires that graders be robust enough to label import features, but grade the qualitative behavior of the function rather than exact points.

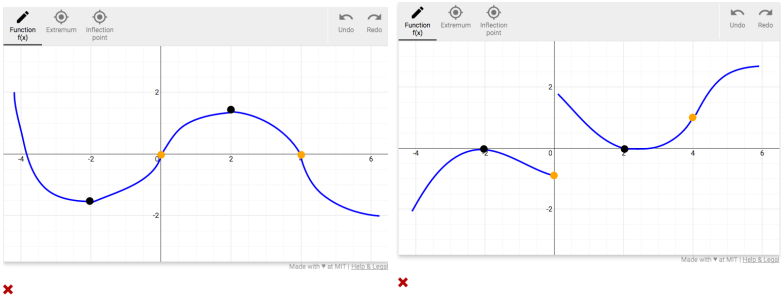
Consider a problem where a student is told that a continuous function f has the following properties:

- $f'(-2) = f'(2) = 0$
- $f'(x) > 0$ on the intervals $-4 < x < -2$ and $2 < x < 6$
- $f'(x) < 0$ on the interval $-2 < x < 2$
- $f''(0) = f''(4) = 0$
- $f''(x) > 0$ on the interval $0 < x < 4$
- $f''(x) < 0$ on the intervals $-4 < x < 0$ and $4 < x < 6$

The following three images are all qualitatively correct, and are all graded correctly by the grader.



The graphs on the right are graded as incorrect. The customizable grading library allows grading scripts to be scaffolded over time, providing varying levels of hinting, varying levels of expectation, as well as introducing new plugins as new concepts are developed.



Future Directions. The immediate goal for the tool involves rounding out front-end functionality with editing functions typical of editing software; the ability to select, delete, and manipulate the various drawn elements will make it easier for students to respond to grader feedback than a simple undo/redo. Once complete, however, we hope to pursue at least two other goals:

1) Interdisciplinary extensions. One key goal involves extending this tool to other disciplines. While there are several disciplines such as electrical engineer and economics that have interest in student generated sketches of functions, we would like to add an arrow drawing plugin, which could be used both in a vector calculus context, as well as to draw free body diagrams in physics, each based on the same grading library, but with different semantic specifications by the instructor. Our main project involves extending the tool to draw force body diagrams of mechanical structures for an introductory structural mechanics course.
2) Machine Learning grading script generation. Currently, the instructor must write grading scripts in Python. To make the tool accessible to instructors with a variety of programming backgrounds, we hope to develop a graphical user interface where an instructor can draw the correct graph, and by "grading" a series of computer generated graphs, will have a grading script automatically generated with appropriate feedback. This customizability would allow more instructors to use the tool without resorting to canned problems that may be outside of the scope of a particular class.

References

[1] Wiggins, G. & McTighe, J. in *Educative Assessment: Designing Assessment to Inform and Improve Student Performance*, Jossey-Bass, San Francisco, (1998).
[2] Atilola, O. et. al.. Mechanix: A natural sketch interface tool for teaching truss analysis and free-body diagrams. *Artifcial Intelligence for Engineering Design, Analysis and Manufacturing, 28* 169-192 (2014).
[3] Chi, M. T. H., & Wylie, R. The ICAP framework: Linking cognitive engagement to active learning outcomes. *Educational Psychologist, 49*(4), 219-243 (2014).
[4] Bransford, J.D., Brown, A. L. & Cocking, R. R. in *How People Learn.* National Academy Press, Washington, D.C., (2000).
[5] Hattie, J. & Timperley, H. The power of feedback. *Review of Educational Research 77*, 81-112 (2007).
[6] Shute, V. J. Focus on formative feedback. *Review of Educational Research 78*, 153-189 (2008).
[7] Ubuz, B. Interpreting a Graph and Constructing its Derivative Graph: Stability and Change in Students' Conceptions. *International Journal of Mathematics Education in Science and Technlology. 38*, 609-637 (2007).
[8] Schneider, P.J. An algorithm for automatically fitting digitized curves. in *Graphics gems*, 612-626. Academic Press Professional, Inc., (1990).