CALCULUS ON THE WEB

GERARDO MENDOZA AND DAN REICH

ABSTRACT. Calculus on the Web is an interactive web-based system for learning basic mathematics. The system presents modular units for study, practice and "hands on" experimentation for a great variety of topics. It also provides on-line management of classlists and assignments, and automated homework grading and grade reports for students and instructors.

1. Introduction

It is the nature of basic mathematics courses, in particular calculus, that learning gradually takes place as a student works out a large variety of routine exercises illustrating the ideas and methods under consideration. It is through practice that a student will apprehend the patterns underlying specific problems, and through many of these that he will gain a genuine understanding of the subject.

It was this principle, coupled with the fact that the only way to ensure that students actually do assigned homework is if it is then graded, that justified for us embarking on the project that has become Calculus on the Web.[1]

In this note we describe some aspects of COW. Section 2 is a description of the system from the student's point of view and Section 3 describes some of the features of the instructor's side of COW. Section 4 delineates some of the internal workings and organization of the student's side of the system. An index of the current contents of COW is included as the end.

The notion of having something on the web that our students could use to work on problems in calculus and get immediate feedback grew from conversations we had during the Fall of 1996. The seed idea, promptly dismissed, was to have a gallery of pictures illustrating certain ideas in Calculus III, which the first author was teaching at the time. This, and the natural extension to have "just" a book available through the web, quickly grew into the concept we then tried to implement. A prototype developed that winter for the first third of the Calculus III curriculum was installed and tested in the Spring of 1997. The first versions of the current system and the instructor's side of COW[2], and most the material for a first course in calculus, were developed in the Summer of 1997 and used in the Fall of that year at Temple University. By now COW includes problems in about 310 topics of which 269 provide an essentially complete coverage of calculus in one and several

Partially supported by NSF through grants DUE #9752335 and #9952234.

Graduate students involved in the initial work were supported by a grant from Philadelphia AMP, the NSF-supported Alliance for Minority Participation.

Additional support for graduate students was also provided by Temple University.

Address: Department of Mathematics, Temple University, Philadelphia, PA 19122.

e-mail: gmendoza@math.temple.edu, reich@math.temple.edu.

[1] http://www.math.temple.edu/~cow

[2] http://www.math.temple.edu/~cow/report

variables and the rest cover some topics in precalculus and undergraduate linear algebra. At the end of this note is a listing of all available modules.

2. Overview of COW

COW is an open system, freely available for universal (anonymous, unrecorded, ungraded) use. More importantly, it can also be used by registered instructors to assign homework, manage classlists, generate grade reports and generally assists them in the way a teaching assistant/grader would help. These utilities are also web-based. The students in their classes can log in, see their assignments, have their work recorded, and see their own grade reports.

COW is organized as a set of books, each a collection of chapters. Within these is a set of sections, which contain the modules. The modules are the essence of COW. Each corresponds to a specific subject, so for example, a module on slope of a line may ask the student to find the slope of the line passing through two given points, and then check the student's answer for correctness and provide an appropriate response. Each module comes with an explanation of the topic. There is great variability on these explanations, according to subject and author of the module. COW is not intended to replace a textbook, but there is a tendency that newer modules explain more than older modules. Some modules are more sophisticated than others, not just because their subject may be more complex, but also because our view of how things can be done has evolved.

COW provides immediate right/wrong feedback as a response to an answer. To the extent possible, also helpful comments when the answer is wrong and how to fix a mistake. It never provides the answer to a (graded) problem. But it will give as many opportunities as needed for the student to give the right answer. Complex exercises are decomposed into steps with error checking at every step. This strategy helps the student learn complex methods. Whenever proper, COW provides graphs of the mathematical objects in the module or question. COW strives to have broad mathematical scope, to reach out to students of many kinds and levels.

To minimize the learning curve, COW uses a simplified, fairly intuitive mathematical syntax for students to enter formulas, and a simple interface design.

On the implementation side, COW provides fast responses in almost all cases. In particular it avoids unnecessary graphics or ornaments, and strives for simplicity in the page layout.

At the user end of COW no special hardware or software is needed, except for a computer, a browser and an internet connection. At the risk of overemphasizing the point, the greatest advantage of the internet-web scheme is that it can make an immense amount of computing power, not just information, available to anyone with a small (or even no) investment. From the point of view of COW, a browser is just a very convenient, practical, and predictable interface with the user.

To maintain the goal of minimal requirements on the user's computer, COW uses fairly low level HTML. Computations are done by the server.[3]

The entry page of COW offers an overview button that leads to a student manual and other useful information, such as details on typing formulas using COW syntax.[4] Some

[3]It is not clear what the future of Java is. We use JavaScripts only in a very limited fashion, since they tend to crash older machines.

[4]The COW's syntax allows, for example, formulas such as `sin 2x = 2 sin x cos x` to be entered and understood. The generals scheme we use is quite natural if one forgets about the usual computer syntax. It simplifies the entering of formulas and is closer to the natural way of writing them.

information directed at instructors is also written in the first page of the overview section (more on this below).

3. The instructor's side

Instructor interested in using COW can contact either of us to set up an account. On our side this consists of nothing more than adding a line to a file with the name, a password, a mail address, and as many courses as the instructor wishes to have using COW.[5] Once an instructor has registered, he can login, set up his own classlists, design and assign homework, have the system grade it, and so on. His students will then login to COW,[6] see the assigned homework, and work on it.

The reporter allows for great flexibility. Reports can be generated not just on assigned homework, but on all the work done by the students. It can also generate reports on homework up to the current date, so instructors can check on problems solved late. A special mode permits the instructor to see how many problems of a given assignment were attempted without ever having been solved completely and also the number of attempts before getting the right answer. This particular feature helps the instructor determine in which topics the students may be having trouble. All reports can be seen on-line, or e-mailed to the instructor.

Assignments can be created, edited, renamed, their due date modified. The system also allows the instructor to set up special due dates for a specific student, thus allowing for special circumstances that make it impossible for the student to complete the work in time.

4. The system

COW consists of three major components. A "manager" that handles navigation and controls the interaction between the student and the "modules". A collection of modules. And a "reporter" that allows registered instructors to assign homework, get grade reports, and generally manage their class. Separate programs handle login and access to the online student manuals. All programs are written in Perl, and we use Maple to check the validity of a student's answer and to create pictures. The operating system we use is a version of UNIX.

The most important component is of course the collection of modules. These are files internally organized into a hierarchy of directories reflecting the layout that the student sees when using the system. For example, the Calculus I Book is a directory containing subdirectories named after the chapters of the book, each in turn containing subdirectories named after the sections of the chapter, and in these there are a number of files which are the modules in that section. The mere act of creating a directory or subdirectory, or placing a file in a section, will have the effect of making the item visible when the system is accessed via web browser. This strategy has proved to be a very convenient way of updating and extending the COW's scope while simplifying the maintenance and organization of the system.

A module is a file organized into segments. Segments are distinguished by special tags that mark their first and last lines. Each segment reflects some aspect of the behavior of the module. They are either Perl programs, or data. For example, each module has a a segment whose first line is `#problem` and whose last line is `#endproblem`. The data for the various problems in that module are stored there, one problem per line. The organization of the

[5] We plan to make it so that instructors need to register just once, and from then on they may set up their own classes.

[6] The first time they enter they'll need to set up a password.

data in each of those lines depends only on the module. Another segment is determined by the `#showproblem` tag. When a student first enters a module, or moves to a new problem in that module, the manager separates out the `#showproblem` segment and executes it. The result of the execution is a file whose contents is sent to the browser, with the effect that the student now sees a question posed, and space as needed to write an answer. The navigation buttons surrounding the question are provided by the manager. The actual data for the problem are read from the `#problem` segment of the module. If there is a picture to be displayed, the `#showproblem` segment can direct execution of the `#makepicture` segment.[7]

When a student types in an answer and clicks on "Check your answer", the manager separates out the `#answerhandler` segment, and executes it, passing along the student's answer and typically also information on the result of previous actions by the student pertaining that problem. Typically, the `#answerhandler` segment will test the answer for a number of syntactical errors, like missing parentheses and other malformations and that the proper variables (or no variables) are used, and convert the answer into Maple syntax. It will then print out a Maple program and pass along the student's answer and the specific data for the problem. It is the job of this program to determine the validity of the answer, and in case the answer is wrong, to determine why. The Maple program returns a diagnosis to `#answerhandler` and the latter produces an HTML page which is what the student sees as a response.

Clearly, determining what is wrong with an answer is the hardest part. What a module should do in this respect is guided by our experience in the classroom. What it can actually do depends on how much structure the answer has (e.g. a number has no structure), and how many tricks we have accumulated up to the time of writing the module. An important aspect of our approach is that it allows for problems that admit many correct answers (for example, find two planes whose intersection is a given line, or a substitution in a multiple integral that converts a given region into one with sides parallel to the axes).

5. Acknowledgments

Most of the modules in the Calculus I Book were written during the Summer of 1997 by us and David Hartenstine, James Palermo, Aaron Robertson, and Daniel Russo, all graduate students of the Mathematics Department at Temple University. Their work was supported by a grant from Philadelphia AMP, the National Science Foundation supported Alliance for Minority Participation.

Development between January 1998 and March 2000, including modules for Calculus II written by Matthias Beck, Daniel Birmajer, Marc Renault and Aaron Robertson, was partially supported by the National Science Foundation's Division of Undergraduate Education through grant DUE #9752335, and by Temple University.

Further development is being partially funded by the NSF's Division of Undergraduate Education under grant DUE #9952234.

[7]There is also a set of subroutine libraries, in particular one to simplify the writing of modules. When a module needs to display a picture it invokes a certain routine whose output is the web-address of the image. This routine checks if the picture exists. If it does, it returns its address. If not, it executes `#makepicture`, stores the image in a directory reserved for the module for that purpose, and then returns its address. So if the picture is static it need not be created again the next time it is required, thus saving execution/waiting time.

6. Current Index of Calculus on the Web

1. Calculus Book I

1. Functions and Geometry
    1. Lines
        1. Slope of a line
        2. Equation of a line
        3. Parallel lines
        4. Normal lines
    2. Distance
        1. Distance
        2. Circles I
        3. Circles II
        4. A point and a line
        5. Circles III
        6. Circles IV
    3. Inequalities
        1. Simple inequalities
        2. Absolute value inequalities
        3. Inequalities and intervals
    4. Functions
        1. Composition of Functions
        2. Roots
    5. Plotting in the Plane
        1. Plotting points
        2. Sketching curves
    6. Trigonometry
        1. Recognizing the trigonometric functions
        2. Trigonometric Identities
2. Limits and Continuity
    1. Ordinary Limits
        1. Finding delta
        2. Basic Limits
        3. Basic limits, more examples
    2. Continuity
        1. Epsilon and delta
        2. A missing value
        3. Discontinuities of simple piecewise defined functions
    3. One-sided Limits and Asymptotes
        1. Limits at infinity
        2. Asymptotes
        3. One-sided limits
        4. More asymptotes
        5. Asymptotes of oscillating functions
    4. Special Limits
        1. Piecewise limits
        2. Trigonometric limits
        3. More trigonometric limits
3. The Derivative
    1. Slope and Tangents
        1. Tangent line slope
        2. Tangent line equation
        3. Differentiability
    2. Linearization
        1. Linearization
        2. Linear approximation
4. Techniques and Theory of Differentiation
    1. Powers, Products, Quotients
        1. Polynomials
        2. Power Rule
        3. Product Rule
        4. Quotient Rule
    2. Trigonometric Functions
        1. Simple trigonometric examples
        2. Trigonometric functions
    3. Chain Rule
        1. Fractional powers
        2. Powers of functions
        3. Trigonometric examples
        4. Chain Rule
        5. Products of powers
        6. Advanced Chain Rule
        7. More examples
        8. More Chain Rule
    4. Implicit Differentiation
        1. Implicit derivatives
        2. Slope of implicit curves
        3. Tangent lines
        4. Normal lines
    5. Theory
        1. Mean Value Theorem
        2. Inverse Functions
        3. Visual Mean Value Theorem
5. Applications of the Derivative
    1. Rate of Change
        1. Velocity
        2. Marginal cost
        3. Displacement
    2. Related Rates
        1. Rates of change
        2. Man and streetlight
        3. A sliding ladder
    3. Maxima, Minima
        1. Relative extrema
        2. Maximum value
        3. More on relative extrema
    4. Graphing
        1. Monotonicity
        2. Monotonicity and singularities
        3. Concavity
        4. Inflection points
        5. Graphing
    5. Optimization
        1. Maximal rectangles under curves
        2. More rectangles under curves
        3. Well designed gardens
        4. An Efficient Poster
        5. Folding a Box
        6. A Maximal Product
        7. Closest point on a curve
    6. L'Hopital's Rule
        1. L'Hopital 1

3. Calculus Book III