well. You should have a good idea how much material you want to cover. And when you plan the course you should allot a certain number of class periods for each topic. If you are teaching undergraduates, they depend on your course for learning a certain body of material (that may be prerequisite for a later course). Don't shortchange them.

A test should be designed for the allotted time slot. You can rationalize giving a two-hour exam in a one-hour time slot by saying to yourself that there is so much material in the course that you simply *had* to make the test this long. This is nonsense. The point of the exam is not to *actually test* the students on every single point in the course, but to make the students *think* that they are being tested on every point in the course. Ideally, the students will study everything—but your test amounts to a spot check. Even if you had a four-hour time slot in which to give the exam, you couldn't really test them on everything, now could you? See Section 2.10 on exams.

If you give a two-hour exam in a one-hour time slot, then you run several risks: that students will become angry, demoralized, alienated, or all three. Telling a student not to worry about his grade of 37/100 because the average was 32/100 does not work (see also Section 1.2). Students are unable to put such information into perspective.

1.10 Computers

Computers are everywhere, especially in mathematics departments. Software for teaching mathematics is also everywhere. Most publishers of basic mathematics texts are absolutely convinced that they cannot market their products without making extensive software resources available to mathematics instructors. It is not clear, as of this writing, that much of this software is actually being used in the classroom. This is so in part because most instructors are not conversant with what is available, are not comfortable with these products, or simply cannot be bothered. There is little objective information available as to which of these products, if any, is a useful teaching tool.

One form of software for the classroom that is being heavily touted these days is the `Mathematica` notebook. Briefly, a `Mathematica` notebook is a self-contained environment by means of which students can interact with the computer over various mathematical issues *without* knowing anything about computing or computer languages. What the student sees when he boots up a `Mathematica` notebook is plain English, and mathematics, on the screen. The computer poses questions to the student and offers guidance in helping the student to find the correct answer(s). When the student errs, the machine provides hints (and encouragement!). (`Maple` has a similar environment called the "Worksheet".)

All of this hardware and software raises fundamental issues about the way that mathematics is taught and the way that it ought to be learned. I drive my car every day and am perfectly comfortable in doing so while not knowing in intimate detail how it works. Ditto for my computer, my telephone, my television, and so forth. Some would say that a measure of how civilized we are

is how many black boxes we are willing to use unquestioningly. To what extent should this point of view be allowed in the mathematics classroom?

Most of us were trained with the idea that the whole point of studying mathematics is to understand precisely why the ideas work. To make the point more strongly, this attitude is what sets us apart from laboratory scientists. We make no statement unless we can prove it. We use no technique unless we fully grasp its inner workings. We would not dream of using the quadratic formula unless we knew its genesis. We would not be comfortable using the fundamental theorem of calculus unless we had seen its proof.

Yet in many colleges these days the linear programming course consists primarily of learning to use `LINDO` or some other canned computer package for applying the simplex method and its variants. The statistics course consists primarily in learning to use `SAS` or `MINITAB` or another commercial product. The undergraduate numerical analysis course consists in learning to use `IMSL`.

It has been argued that the use of programmable calculators and software such as `Mathematica` notebooks in the calculus classroom will free students from the drudgery of calculation and will allow us to teach them how to analyze multi-step word problems (see, for instance, [STE2]). Thus, it is hoped, our lower-division calculus classes will be more closely tied to the way that mathematics is used in the real world. I see this as double talk.

It is certainly the case that the less able students are so hampered by their inability to take the derivative and set it equal to zero or to find the roots of a quadratic equation or to calculate the partial fraction decomposition of a rational expression that they have little hope of successfully analyzing a multi-step word problem whose solution includes one or more of these basic techniques. But there is no evidence to support the (apparent) contention that a person who is unable to use the quadratic formula will somehow, if these technical difficulties are handled for him by a machine, be able instead to analyze conceptual problems. Using the quadratic formula is easy. Analyzing word problems is hard. A person who cannot do the first will also probably not be able to do the second—with or without the aid of a machine.

Consider the following analogy. A software wizard approaches a frustrated piano student and says "For $100 I'll give you some software that will play scales and emulate hand positions for you. That way you can go straightaway to playing Tchaikovsky." Few people would lend credibility to such a story, yet in the context of mathematics education we sometimes delude ourselves into using analogous teaching techniques.

I think, as I have already indicated, that the computer can be used successfully to provide helpful graphical analyses. If you want to draw a surface in space and then move it around to analyze it from all sides, then there is nothing to beat `Mathematica`. If you want to illustrate Newton's method, or use the Runge-Kutta method, or implement Simpson's rule, then a computer is almost indispensable. But the use of the computer should be based on a firm foundation of conceptual and technical understanding.

Let me stress that most reformers concur with the statements in the last paragraph. It is incorrect—and certain passages in the first edition of this book could be construed as making this claim—to suggest that all the reformers want

to do is abandon lectures and let the kids spend their time horsing around with computers (see J. Jerry Uhl's Appendix, where he spells out in detail how his teaching methodology uses computers). First, there is no single reform dogma. Second, there are many different opinions—both among reformers and among traditionalists—about how best to use computers in instruction. My few experiences using computers in instruction have been rather positive. I would encourage others to be open to at least trying some of the different ways that they can be used in the classroom.

My own view about computers is that one of the highest and best uses of the computer in mathematics instruction is as the basis for *laboratory* work. I would like to see a lower-division mathematics class still consist largely of classroom instruction—either lecture, or group work, or discovery activities—but I would like to see the classroom work re-enforced by well-thought-out computer labs. Chemists have understood this notion for many years, and have made laboratory work an integral part of basic instruction. We should do the same. Mathematical ideas are abstract. Elementary mathematics students feel as though they are staring into a crystal ball, with nary a place to gain a foothold in the subject. We should use laboratory activities to make the subject more tactile for them.

For example, if the class is treating level sets for functions of two variables, then there should be a lab activity in which the student (with the aid of a machine) constructs level sets for various functions, and then sees how they are assembled to form the graph in three-dimensional space. The student should be able to "walk around" on the graph, and verify that the gradient of the function is perpendicular to the level sets. The student should be able to do barehands confirmation of the method of Lagrange multipliers. Properly viewed, this is fertile and largely unexplored territory.

There is an important point to be kept in mind. The computer lab activities should be tightly integrated into what is happening in the classroom—reinforcing what happened in the previous class and anticipating what will happen in the next. The lab activity might involve graphics, or numerical calculations, or computer algebra. But it should not be computer activity for its own sake—it should be part of the learning process.

We had better realize that making the highest and best use of computers in teaching our lower-division courses might entail rethinking our traditional classroom activities. I don't have all the answers here, and neither does anyone else. But if you try tacking some silly little computer activities onto your lectures, and if the end result is not particularly satisfying, then that is not necessarily the end of the story. You may have to rethink the way that the entire course is taught. Just as the computer activities are supposed to reinforce the ideas introduced in class, so the activities in class should reinforce the computer activities. It's a symbiotic process, and one which will require careful planning if it is to be successful.

We have had catastrophic experiences, at my own university, trying to introduce `Maple` or `Mathematica` or some other software into isolated courses. Students are not stupid. They catch on right away to the fact that this software is specific to the particular course, and as soon as the semester is over they are unlikely to see it again. They resent having to learn a whole new language—

especially since it will be used in just one course for only one semester.

What these life experiences tell me is this: If we are going to introduce serious software into the lower-division curriculum then we should do it globally instead of locally. Of course this is tough. We'll really have to plan. But it makes sense that we should tell students from day one that their entire lower-division mathematics curriculum will depend on `Maple` (or `Mathematica`, or another alternative) and that they need to master it right away. Having understood this dictum, they will comply (if only out of an instinct for survival) and the software will become a part of their *lingua franca*. They will (we hope) carry it (or the analytical skills attendant to it) on to the rest of their education, and their lives.

At my own university, when we teach sophomore-level ordinary differential equations, we commonly include a "computer unit" in the course. This means that there are four or five computer-oriented homework assignments that the students must do—in addition to the more traditional handwritten assignments (with separation of variables, first-order linear, second-order, constant coefficient, and so forth)—that are geared to the text and to the lectures. A typical computer-oriented homework assignment sets a specific ODE for the student to consider, and guides him through several steps of hand calculation—to a point where it becomes clear that further hand calculation is infeasible. And then it says, "Now let's do a phase plane analysis," or "Now let's look at Runge-Kutta," or "Now let's do a numerical or modeling experiment." My experience is that students have gone into this computer-oriented material kicking and screaming. But, at the end of the semester, they have come out of it convinced that they have learned a new set of tools, and a new view of scientific analysis. I take no credit for the construction of these computer-oriented problem sets, nor for their efficacy (they were created by my colleagues). But I think they are marvelous, and I would like to see this approach used in more undergraduate courses.

Of course what I am saying here is not news to those—both of the traditional and of the reform turn of mind—who have been studying mathematical pedagogy professionally. The articles [BOU] and [HEI] discuss many aspects of computer use in mathematics teaching.

Certainly some of the most innovative uses of the computer in the lower division classroom are due to Ed Dubinsky and his coworkers [DUB]. Dubinsky uses the programming language `ISETL`, which has the attractive feature that its code is very much like the mathematical notation to which we are all accustomed. Dubinsky is a firm believer in the notion of Beth/Piaget [BPA] that each student must build the mathematical ideas for himself in his own mind. He argues that `ISETL` is a powerful tool in helping students to do so. In particular, the programming language cuts through the mental inertia that we have all experienced. Instead of sitting and staring at a new idea in bewilderment, the student turns to `ISETL` and *begins to try things*. There is no arguing with success. Dubinsky and his collaborators have calculus texts [DSM] and [C4L], an abstract algebra text [DUL], and a discrete mathematics text [DUF] that implement these ideas.

There are some instructional uses of the computer that I don't understand. Let me begin with an analogy. Imagine a chemistry instructor facing a classroom of 100 students, each with chemistry apparatus set up before him. The instructor says, "OK, today we are going to make nitroglycerine. Take your flask, and put

these chemicals in it. Now heat it over your Bunsen burner ..." Just imagine! Some student on the left, in an effort to keep up with the instructor's pace, fumbles and spills a caustic substance in his lap; a student on the right gets excited, shakes the nitroglycerine, and is blown to smithereens.

You chuckle. But what better sense does it make to have a mathematics classroom with a computer before each student and the instructor delivering commands to the students? Many students are not conversant with the computer environment. Perhaps their hand-eye coordination is not fully up to snuff. Many will not be able to keep up. Hit the wrong key and you get a screen full of chaotic gibberish, or you lose your work, or you accidentally reboot, or you erase your boot sector.

People need to perform laboratory activities in their own time and at their own pace. They need to have the leisure to make errors and to repeat steps. The chemists know what I am talking about. They let the students schedule their own labs, and do things (under mild supervision) in their own time. The proof is in the pudding: It works. It makes sense to me that, as we rethink the way that we teach mathematics, we should take into account what other sciences have learned from their experiences.

Now let me summarize my point. I am all in favor of computer labs to accompany mathematics classes. But I can make no sense of "electronic classrooms"—in which each student, or each pair of students, has a computer before him. How much could you accomplish in a 55-minute class if 45 of those minutes are spent fumbling around on computers?

Of course the computer also raises epistemological issues, and I hinted at some of these at the start of the section. Do we want our students to understand how to integrate by parts, or do we want them to know how to tell `Mathematica` to do it for them? Do we want students to understand the simplex method, or do we want them to know how to tell `LINDO` to do it for them?

I think that there are several correct answers to these questions, and I think that the appropriate answer for any given context depends on who is in the class we are teaching. Are we teaching future mathematicians, or future nurses, or future businessmen, or future chemists? See Section 3.9 for a discussion of the issue of suiting your teaching to your audience.

Peter W. Jones once made the following cogent observation. After being in school for a number of years, students learn that their instructor is fallible, and they learn that even the text is fallible. But if the calculator or computer says that something is so, then it must certainly be correct. I have had the experience of giving students assignments to complete in the computer lab, with the following result. After spending many hours, they would bring to me pages of gobbledegook—long streams of meaningless ASCII code. They assumed that this is what I wanted, because it is what the computer produced.

Perhaps the discussion in the last paragraph illustrates the following point. A live teacher does things in real time. But a computer does things so fast that it trivializes them. When something is trivialized, then understanding is lost. These statements are not made to downplay the potential value of the computer in the classroom, but rather to put it in perspective.

It is probably the case that students learning calculus from `Mathematica`

notebooks respond positively to the extra attention they are receiving, to the novelty of the teaching environment, and to the fact that making a mistake when interacting with the computer is less heinous than asking a dumb question in class. A good `Mathematica` notebook will never produce streams of nonsense symbols. So in that sense the environment is controlled and the flow of errors kept in check. In the `Mathematica` notebook environment, the student is probably more willing to try things and to experiment. Such an environment could serve as a catalyst for creativity. We should think carefully about how to capitalize on the special features that computers can contribute to the learning experience. However it is clear that we do not yet know all the answers, nor have we realized all the potential.

Today there is much discussion of self-discovery, and of students making their own conjectures, of students doing group work, and of students building the ideas in their own minds. All well and good; but how can a student discover the fundamental theorem of calculus for himself when it took great historical geniuses like Newton and Leibniz to do it the first time around? How can a person with no specialized training make conjectures? How can students carry on cogent group discussions if they don't know what they are talking about? How can a student with no experience in critical thought build ideas in his own mind?

These are serious questions, and members of the reform movement are finding ways to answer them. The computer can be a powerful tool in this quest. If two or three students are sitting together in front of a computer screen, going through the paces of a well-constructed lab, then they definitely have something to talk about. They have the catalyst for some group work. The computer lab can certainly give them grist for formulating conjectures. And the computer can pace them through a self-discovery process. Ed Dubinsky's use of the programming language `ISETL` ([BDDT], [DUB], [DUF], [DSM], [C4L]) provides evidence that the computer can be used to help students to reconstruct ideas in their own minds. The computer is a powerful new tool that is at our disposal. We must train ourselves how best to use it.

But let us bear in mind the techniques that we already know, and that are of proven value. Don't forget: If a student spends an hour with a pencil—graphing functions just as you and I learned—then there are certain specific and *verifiable* skills that will be gained in the process. By learning how to *create* a graph, the student will also learn how to *read* a graph. By contrast, `Mathematica` or `Maple` can create a flawless graph much more quickly and easily than can a human being, but these software products *will not teach the student how to read the graph.* If we keep these guiding principles in mind as we develop methods for using computers in the classroom, then I think we will end up with more valuable teaching tools.

S. Hildebrandt used to remark that a student who solves a problem two different ways will learn a great deal more than the student who solves two different problems, each in only one way. There is wisdom in this remark, and it is salient to the issue of teaching using computers. As already noted, computers do things too quickly, and tend to trivialize them in the mind of the user. The student who can solve a problem swiftly and easily at the keyboard is not likely

to want to engage in the conceit of solving it twice.

A wise man once told me that the computer is a solution looking for a problem to solve. It is obviously a powerful tool in the right circumstances. The generation of minimal surfaces of arbitrary genus by Hoffman, Hoffman, and Meeks ([HOF]) is a dazzling use of computers. The simulations that these three mathematicians performed would have been inconceivable without this advanced computer graphics technology. `Mathematica` is a powerful tool in the hands of the right user, as are `AXIOM`, `MACSYMA` and `Maple`.

Calculus is perhaps the most powerful body of analytic tools ever devised. All young scientists should learn calculus in essentially the traditional fashion so that they have these tools at their disposal. *Graphing a function* is one of the most basic processes of analytical thinking—analogous to finger exercises for the piano. The partial fractions technique is also one of the most far-reaching algebraic devices that we have. Integration by parts is perhaps the most ubiquitous and powerful tool in all of mathematical analysis. Letting a computer do these processes for the students abrogates much of what we have learned in the past three hundred years.

Nobody would be foolish enough to assert that one can learn to spell by using a spell-checker or to write by using a grammar-checker or to play the piano by listening to someone else do it. I sometimes wonder why people think that students will learn mathematics by letting `Mathematica` do the thinking for them.

Let me assure you that I have discussed this point with the presidents of large high tech corporations and they agree with me absolutely. Use of the new technology should be layered atop a traditional foundation. And, if you'll pardon my being a bit droll, that traditional foundation should consist of "readin', writin', and 'rithmetic". That is what works in the classroom and that is what works in the real world. In fact in one memorable conversation that I had at a board meeting of H. R. B. Singer (a company that produces products for electronic warfare), I said to the CEO, "There are a lot of new ideas in the air about teaching mathematics. We are wondering which ones to adopt. Are you content with the traditional mathematical training of the college graduates that you hire?" He said, "Their mathematical training is fine. I just wish that they could write English."

1.11 Applications

One of the most chilling things that can happen to an unprepared, unseasoned faculty member is to have a belligerent (engineering) student raise his hand and say, "What is all this stuff good for?" And one of the most irresponsible things that a faculty member can say in response is, "I don't know. That is not my problem." If you do not have an answer for this student question, then you are not doing your job.

I have found it useful in all of my undergraduate classes to tell the students about applications of the techniques being presented *before* the aforesaid chilling question ever comes up. This requires a little imagination. If I am lecturing