Math, with an Attitude

Linda Petzold

July 11, 2002

Abstract

We describe from experience how it is possible to start out in the mathematical sciences and end up working on practically everything.

In this address to the Olga Taussky Todd Celebration of Careers in Mathematics for Women, I will begin by briefly giving my story. That is, how it is possible to start out in the mathematical sciences and end up working on a wide variety of problems in engineering and science, and how those problems have in turn influenced the mathematics. Although I am pleased with what I have managed to accomplish so far, in retrospect it has been mostly unexpected and unplanned. In the second part of this lecture, I will try to give some advice based on experience on how, with a more aggressive and well-informed approach, one should be able to do better.

I received my PhD in Computer Science from the University of Illinois at Urbana-Champaign in 1978. My thesis research was on the development of efficient numerical
methods for highly oscillatory ordinary differential equations. I chose to study numerical analysis and computer science because: I enjoyed mathematics but at that time the job market in mathematics was rather bleak, I loved computing, and I was interested in applying my work to the solution of engineering and scientific problems. This seems to have been a good decision, although working in this field is not at all what I thought it would be — in addition to the analysis and the computing that I expected, there is a lot of interaction with engineers and scientists on a wide variety of topics that can also be very rewarding.

For my first job, I went to Sandia National Laboratories in Livermore California to work as a numerical analyst in the applied mathematics group. I wasn’t sure what I should be working on, and my supervisor's advice was to ‘make yourself useful’. So I set about to talk to the engineers who shared our building. At the time, I found it difficult to approach people and ask what they were working on, and even more difficult to imagine how I could make a worthwhile contribution. But apparently this was my job, so I did it. The people I talked to were mostly doing computational combustion, and modelling of fluid flow systems in solar central receiver power plants. These seemed like interesting problems, and there were differential equations that needed to be solved, so I was off to a reasonable start. Both the combustion and solar energy problems could be expressed, after semi-discretization of the spatial derivatives, as systems of differential-algebraic equations (DAEs),

\[ F(t, y, y') = 0, \quad y(t_0) = y_0. \]

In both applications the engineers were using, or planning to use, rather primitive methods for the time integration. So I thought here is an area where I might be able to contribute,
by building a state of the art DAE solver.

I wrote such a solver, and at first it was a huge failure! We tried to use it in simulation of a solar power plant, which by that time was becoming a rather urgent problem. After a few time steps, though, the solver mysteriously failed. I had the feeling then, and still do, that if you build software and your friends and colleagues are kind and courageous enough to try to use the first versions of it for solving their research problems, you should be prepared to give it your full attention when that software fails. I spent the next month doing nothing but looking for the bug that must have caused this failure. After examining and testing everything, including at one point all of the elements of the local Jacobian matrix for this large-scale problem, I was finally convinced that there was no bug in the code that could have caused this failure and began wondering if there was something strange about the problem.

At about the same time, a report from Boeing Computer Services [1] came to my attention, about some new results on the numerical solution of linear constant-coefficient DAEs with applications in circuit analysis. The report showed how a result from linear algebra could be used to decompose the linear constant-coefficient DAE system into simpler sub-systems which could then be analyzed from the point of view of mathematical structure and convergence of numerical methods. Although there were convergence results for some simple numerical methods in the report, the decomposition showed that problems could be written in the form (1) which had properties that were quite different from those of standard-form-ordinary differential equation (ODEs) $y' = f(t, y)$. These problems, which were distinguished by their high index, a property of the DAEs which is apparent when they
are written in the decomposed form, have a mathematical structure which is quite different from that of standard-form ODEs. For example, here is a very simple index-2 DAE:

\[
y_1 = g(t) \\
y_2 = y_1.
\]

(2)

Notice that this differs from a standard-form ODE in some fundamental respects: the solution is completely determined by the right hand side and any initial values must be consistent with the right hand side, the solution is less continuous than the input function \( g(t) \), and the DAE has a hidden constraint (that is, the solution must also satisfy \( y_2 = g'(t) \)).

I wondered whether the solar power plant problem might be a nonlinear version of such a high-index DAE, and if so whether that could somehow be the source of the mysterious failure. After some time pondering the solar power plant problem, I decided that it was indeed a nonlinear version of a high-index DAE system. The high-index problem arises in the simulation of the flow of an incompressible fluid in the pipes in the plant. Roughly speaking, the pressure acts like the ‘index-two variable’ \( y_2 \) of (2). Notice that the smoothness of this variable depends directly on the derivative of the input function (or in a more general nonlinear case, on the derivatives of other problem variables), hence one might expect that discretizations (numerical ODE methods) which were designed for smoother problems and variables might not be appropriate for this problem, and in particular for the pressure variable. To try to understand whether this was indeed the source of the problem and get a handle on how to fix it, I studied the discretization schemes applied to the model equation (2). Although the results in [1] indicated that the backward differentiation formula (BDF)
methods I was using should converge for this problem, they were only applicable to linear constant coefficient DAEs and with constant time stepsizes. When I studied the index-two model problem for non-constant stepsizes, I found that there were problems with the formulas for variable stepsizes, and there were problems with the error estimates which are used to determine the next stepsize and decide whether the current time step has been successful in an ODE code. Indeed, when I tried my solver on (2), it failed in the same way as it did for the solar power problem!

To make a long story shorter, it was easy to see from the model problem that a reasonable strategy for solving the solar power plant problem with my adaptive code would be to remove the index-two variables (pressure at every node) from the stepsize selection and error test decisions. At this point, there was not enough theory to guarantee that this would give accurate solutions to the nonlinear solar power plant problem with the adaptive solver but at least the theory in [1] indicated that for the related index-two model problem (2) solved by the same formulas with a fixed stepsize, there was convergence. So we modified the solver to remove the pressure variables from these decisions. It worked fine! After extensive testing and carefully inspecting the solutions, we were convinced that they were correct except possibly for the pressure variables. Thus we could solve the problem, and in fact went on to develop a tool which could solve many related problems, which was important to our employer.

This success in solving the problem had another important benefit for me. It gave me the chance to go back and study the problem further to really understand it. When I decided to
do this, I got some well-intentioned advice from a number of well-known and well-respected colleagues that it was not a good idea to put much effort into this class of problems (DAEs), because it was somewhat ‘out of the mainstream’ of numerical ODEs and perhaps of only limited interest. I thought about this but then decided to go ahead with this work anyway because there were a lot of interesting research issues, and I could see how it could impact the solution of problems at Sandia. That turned out to be a good decision. I spent about 15 years working intensely on analysis, numerical methods and software for DAEs. Much of this work is summarized in the two books [2] and [3].

The original software which hadn’t worked underwent quite a few modifications and several rewrites while it was solving more and more challenging problems at Sandia. Eventually this code became the DASSL[2] software, which is available on the World Wide Web via netlib. I had written DASSL to solve a few problems at Sandia, but to my amazement even in the days before the World Wide Web, the Sandia engineers told their colleagues about it, and within a few years DASSL had hundreds of users worldwide, solving a wide variety of problems. This was an important development for me because, although in many cases I was not directly involved in the solution of these problems, it brought some challenging problems and interesting new applications to my attention and motivated a lot of my research. In 1985 I left Sandia to become group leader of the numerical mathematics group at Lawrence Livermore National Laboratory, where I worked with Alan Hindmarsh and Peter Brown to develop a code DASPK[2] which uses the basic time-stepping methods and strategies of DASSL, but incorporates methods for efficiently dealing with very large-scale DAE
systems that arise, for example, after semi-discretization of the spatial derivatives for two
and three-dimensional time-dependent partial differential equation systems.

In the meantime, DAEs had gotten ‘into the mainstream’. I believe that the software
played a large part in this. If you have developed a good tool, then engineers and scientists
will beat a path to your door to get to use it. As a result, more challenging problems will
come to your attention, you and others will develop more theory which will enable even
more powerful software for the solution of more difficult applications. It is a great cycle!
But I should caution that software development is a huge investment of your time that can
only pay off if there are important problems that could be solved better or more easily with
the new software. In this respect, I was really fortunate to be at Sandia and at Lawrence
Livermore National Laboratory during the first 13 years of my career, with relatively easy
access to interesting problems and with no need to worry about tenure. Another benefit of
working in a non-academic environment was that when I had my son in 1984, I didn’t have
to worry about a tenure decision (and I have a nice memory of a baby shower which was
given to me by my colleagues, who were all men).

In 1991, I decided to move to academia. I wanted to have students, more direct control
over my research funds, a wider range of problems to work on, and a lower overhead rate! I
went to be Professor in the Department of Computer Science at the University of Minnesota.
The research went well and I built a fine research group — it was not very difficult to
transition the research from the Laboratory environment and to obtain grant funding in
academia. The transition to teaching was more difficult (and tiring!) but after a while it was
successful. In 1997 I moved to the University of California Santa Barbara, to the Departments of Mechanical and Environmental Engineering, and Computer Science. I am still developing and analyzing numerical methods, writing software (lately for sensitivity analysis and optimal control), and solving engineering problems (lately simulation and optimal control for chemical vapor deposition of superconducting thin films, trajectory control of spacecraft, and tissue engineering — optimizing the fabrication of a bioartificial artery).

At this point in writing the lecture, I decided to try to write down some advice based on what I think I have learned, sometimes by trial and error, sometimes by luck, and sometimes by good advice, regarding career development and research. I cannot guarantee that this is good advice, but for the most part it has worked for me. If it can help even one person who is reading this, then the writing will have been worthwhile.

In research,

1. Look for problems with *impact* as well as *interest*.

2. Be *flexible* and *open* to new ideas and problems, even if you don’t understand everything at first.

3. *Follow your instinct.* Is the problem new and interesting? Is there a need? Keep in mind that the research community can be slow to accept change.

4. *Negative results* can be important — in their absence, a bunch of researchers will continue to work in the wrong directions.

5. Don’t hesitate to *write papers.* If it is interesting and/or useful, you have something
to say.

6. *Accept all invitations* (within reason). Visibility is extremely important in the research community.

7. *Make your work available and accessible.* Some ways to do this are by writing software, interdisciplinary collaboration, directing some of your papers to a wider audience, and via the World Wide Web.

8. In your lectures, emphasize the importance of the problem and your contribution. Keep in mind who is your audience. Some subjectivity is OK (even expected) in a talk, so give your opinion. A flashy title can sometimes be useful (I use this trick :-)).

9. For your *grant proposals*, believe in what you are doing and communicate why it is important and interesting. Remember that if you don’t ask, you don’t get.

10. When you are asked, take the opportunity to do some *professional service*. Serve as referee, editor, on NSF panels, . . . (but not to the exclusion of your research!) to see firsthand how the system works.

In managing your career,

1. *Focus* on your personal and professional objectives.

2. Balancing personal and professional life is a big issue, especially when you have children.

   It can (and should) be done, but you will have to be flexible and resourceful in managing your time.
3. Don’t be afraid to ask for advice.

4. In choosing an employer, go where it is interesting and friendly, where there is emphasis on science and recognition for good work, where you will have a chance for visibility and where your work could make an impact.

5. A mentor can be extremely important. I have had several great mentors in the early part of my career, all of whom happened to have been men.

6. Friends and collaborators are important, particularly over the long term.

7. Be confident (or at least project confidence — you might be amazed at how many of us have had to ‘fake it’ in the early part of our careers).

8. Be aggressive. Seek out the people and problems you want to work on, and GO FOR IT!!!!!!

This paper is based on a plenary talk at the Olga Taussky Todd Celebration of Careers in Mathematics for Women. Linda Petzold is a Professor of Mechanical and Environmental Engineering at the University of California, Santa Barbara.

References
