An interview with

JOSEPH F. TRAUB

Conducted by Thomas Haigh

On

19 and 20 March 2004

New York, New York

Interview conducted by the Society for Industrial and Applied Mathematics, as part of grant # DE-FG02-01ER25547 awarded by the US Department of Energy.

Transcript and original tapes donated to the Computer History Museum by the Society for Industrial and Applied Mathematics

ABSTRACT

In this interview, Joseph Traub begins by discussing his involvement, while a Bell Labs researcher in the 1960s, in the establishment (with Phyllis Fox and Morven Gentleman) of a project to create a portable library software library of high quality, tested routines for numerical analysis. He also gives more general commentary on the relationship of software to the discipline of numerical analysis, and on his personal experience with programming. He then discusses his role as founder of ACM SIGNUM, the special interest group for numerical mathematics. In the second half of the interview, Traub talks more generally about his work in the twenty years since OH 70, 89 and 94 were conducted. Topics include the further development of information-based complexity, his work on the computation of financial derivatives, his interest in quantum computing, his role in the establishment of the Computer Science and Telecommunications Board of the National Academies, his founding of the Journal of Complexity, his early concern over security of the national information infrastructure, and his external professorship at the Santa Fe Institute.

THOMAS HAIGH: Interview with Joseph F. Traub. This interview is being conducted by Thomas Haigh. It is the 19th of March, 2004, and the interview is taking place in Professor Traub's office at Columbia University. This interview follows on from three earlier oral history interviews, Charles Babbage Institute numbers OH 70, 89 and 94, and is intended to be read as a supplement to them. It is conducted as part of the SIAM historical project to investigate the history of scientific computing and numerical analysis. Professor Traub, good afternoon.

JOSEPH TRAUB: It's a pleasure to have you here.

HAIGH: One of the main functions of this interview will be to probe some areas which weren't covered to as great an extent in the earlier interviews, particularly related to the history of software, and particularly, standard libraries and routines in the history of scientific computing. There'll also be a chance to discuss your important work in a number of areas over the past 20 years since the interview was conducted, and also to amplify or to give the benefits of two decades more experience on some of the general kinds of issues that were discussed in the earlier interviews, for example the place of numerical analysis and scientific computing within computer science, so that's quite a wide range of material to cover there.

I wonder if it would be alright if we were to start by discussing the project you were involved in which is most directly to standard libraries for scientific computing, which was conducted during your time at Bell Labs during the 1960s.

TRAUB: That's fine.

HAIGH: So, perhaps you could begin by saying something about the motivation of that project. Why it was that at that point in history you, and presumably other people at Bell Labs, believed that there was a need for some kind of standard high level library.

TRAUB: The motivation for developing a standard library is that certain problems keep arising in many different scientific fields. Examples of such problems are: the numerical solution of ordinary and partial differential equations, integration, algebraic eigenvalues, large linear algebraic systems, and polynomial zeros

HAIGH: And exactly the same mathematical requirements existed in a very wide range of application areas?

TRAUB: Certainly similar requirements.

HAIGH: Now, you arrived at Bell Labs, I believe, in 1959?

TRAUB: September, 1959.

HAIGH: You talked in the previous interview about the work that you did on iterative algorithms.

TRAUB: In 1959 I got interested in the theory of how do you pick the best possible algorithm, what's the intrinsic difficulty of a problem, and for historical reasons the first area I looked at was iterative methods for solving nonlinear equations. In 1980 when Henryk Wozniakowski and I wrote our research monograph "A General Theory of Optimal Algorithms", we realized that the study of optimal iterative methods, is technically, a particularly difficult area. In 1965 the study of optimal algorithms was called computational complexity, but that phrase didn't exist when I started my work in this area.

HAIGH: Right. It's my impression from your earlier interview that in 1959 there wasn't a particularly strong or directed research effort in computing going on at Bell Labs. Or at least it wasn't that you turned up and then people said to you "this is what you must work" on and you had to fit it into an ongoing program.

TRAUB: I was one of the first people hired by Bell Labs who had received their Ph.D. for a computing thesis. I received my degree under the Committee of Applied Mathematics at Columbia, because there was no computer science department there. Indeed, there were no computer science departments until the mid 1960s. I believe that Purdue started their department in 1962, but Carnegie Institute of Technology, Stanford, and others started in 1965. I was hired by the Mathematics Research Center (computer science wasn't split off until quite a bit later).

The sixties were a golden time to be at Bell Labs. I was very lucky. A number of times in my life I have been,by luck, just at the right place and this was one of them. It wasn't entirely by luck because I always went for quality, and I chose Bell Labs because I was so impressed by its quality. In those days once you were hired by the Research Division at Bell Labs you were free to do any research you wanted. You were rewarded at salary time, particularly if your work had impact at the Laboratories or at ATT and, of course, great discoveries were made at Bell Labs during those days.

One of the sayings at Bell Labs was "he wrote the book". In my first few years at Bell Labs I buried myself in research which led to the 1964 monograph, *Iterative Methods for the Solution of Equations.* It should have been called *Optimal Iteration Theory*. The book has been in print ever since due to a number of publishers. When people write about optimal iterative methods they always cite this 1964 monograph and that always makes me feel good.

HAIGH: So, given this enormous freedom, why and how did you become interested in the issue of standard software libraries?

TRAUB: In 1966, I was on sabbatical at Stanford and when I returned to Bell Labs I was made supervisor of a research group. At the time, I was interested in exploring for what class of problems one could create global iterative methods; that is, methods that always converge. I also wanted a project that was good for the Labs, good for the community, and asked why people should have to write algebraic eigenvalue solvers, ODE solvers, etc. Let's try instead to identify the important modules and then build software with certain characteristics. It should be portable and high quality. It should be

independently refereed the way a scientific paper is independently refereed. The test to be performed on the software depended on the particular subject and the subject I was personally interested in was the testing of polynomial zero finders. Somewhat later, I got involved with building software in that area.

HAIGH: You talked about how much freedom researchers at Bell Labs enjoyed. Now, when you were head of a research group there was that like being a manager where you had people you could tell what to work on, or was it like being a department chair where your control was much more indirect?

TRAUB: Everything I did was my own work or with collaborators. For the software library project one of the participants was Morven Gentleman who was a member of the technical staff, and was, therefore, free to do whatever he chose. On the other hand there were people like Phyllis Fox, who we hired as consultants to work on this project.

HAIGH: And were the kind of libraries that you envisioned the project producing things that as you personally had worked on computations previously you had wished had existed for you to use?

TRAUB: Exactly. I'll give you a very concrete example of that. As part of my Ph.D. thesis, I had to solve a large eigenvalue problem. I went to Cornell University to meet with Professor Kato, who referred me to an algebraic eigenvalue solver. It occurred to me that we should have standard modules that we could use.

There were some very good people working on mathematical software. In particular, at Argonne National Laboratory, there was a man named Jim Cody who was an expert on mathematical software. I can't think of any university faculty that were building software. Perhaps it was not a way to get tenure at a good university.

HAIGH: That's an interesting point. So, the freedom from having to come up for a tenure review, and show that you'd published papers in which theorems were solved, and other mathematically respectable things were done, that in itself made Bell Labs almost a uniquely suitable place to pursue this kind of work.

TRAUB: Another such place was Argonne. Somewhat tangentially, starting in 1955 I did my own programming. My thesis was a huge programming project. Later, I'd spend a couple of weeks on some programming project, say for a polynomial zero finder or for my first book. Professors typically didn't program. They had their students program, and I remember looking down on these people who didn't do their own programming. But after a while I became the same way because programming means you have to bury yourself for weeks or months in a project. It is different for people who work in areas such as software engineering but as a theoretician I stopped programming. I had become guilty of the same behavior. Certainly I had no time for extensive programming after I got to Carnegie in 1971.

HAIGH: Do you ever miss it?

TRAUB: No. I act on the principle that if there is someone around who can help me do something, then I don't do it.

HAIGH: So, during the mid 1960s, when the library project was getting under way, would computer manufacturers such as IBM have supplied useful routines in any of these areas?

TRAUB: The general history of software libraries is that every time there was a major change in technology the manufacturers would repeat the same mistakes when it came to scientific software. This happened when we went from mainframes to minicomputers, and then to microcomputers.

HAIGH: Within Bell Labs, when the project was started, did the academic freedom extend pretty much just to be able to start the project on your own initiative, or did you need to persuade anyone that this was something that institutionally Bell Labs had a stake in promoting?

TRAUB: Yes, I must have gotten permission for this project since I was actually hiring consultants, but I don't remember any difficulty in getting the project approved.

HAIGH: And can you remember whether in justifying the project you would have stressed the general benefits to the academic community, or benefits to the specific kinds of computations that people elsewhere in the country were presumably doing to insure good functioning of the network?

TRAUB: By the net, do you mean AT&T?

HAIGH: Yes.

TRAUB: First of all there was no academic community; we're talking about the mid 60s, and academic computer science was just starting. The feeling was that this was a good thing for Bell Labs and good for the field, to have good standard software modules.

HAIGH: You would never have someone come along and say, "you know, we've got this problem in making the telephone system work, can you help us with it"?

TRAUB: No one came to me for help in making the telephone system work. Several times I was asked for help with the numerical solution of partial differential equations. On occasion, I was asked to act as a consultant to some other group, which of course I did.

HAIGH: As I understand what you've said about the project, its three distinctive features would be: that the software itself would be peered reviewed and tested very carefully; that it would be in a higher level language; and that you would try and cover a broad range of different areas.

TRAUB: That's correct. We were hoping to use the best software we could find. What actually happened is we found that we were writing the software ourselves.

HAIGH: So that the idea had been that it would be more like editors of the journal, but in practice you found that you were providing most of the content yourselves?

TRAUB: We were writing the papers in this journal, yes.

HAIGH: That's interesting. And had this been the first attempt, that you know of, to apply the idea of peer reviewing to software?

TRAUB: The idea was that it should, in a way, be a hostile review like a tiger team. These days. if you want to prove the security of your system, a tiger team tries to break in. I'm not sure the words tiger team were being used in the 60s.

HAIGH: I don't think it would have been. I think even the idea of a walkthrough or a code review was only popularized a few years after that.

TRAUB: Apropos of that, but something I think was quite unique in those days, was that Phil Sherman and I built a macro compiler that counted machine cycles. We had no idea how long the basic modules we used all the time took to run, such as sine and cosine routines. That may have been one of the first examples of performance evaluation which later on became a big field.

HAIGH: That's published?

TRAUB: It was a Bell Labs technical report,. [*A Macro Compiler that Counts Machine Cycles*, 1961 (with P.M. Sherman)].

HAIGH: One of the other crucial goals was portability. Now theoretically one of the advantages of a high level language, like FORTRAN, would be that code written in it would be portable. Now, as anyone who's ever tried to move the high level program from one system to another knows, in practice that doesn't necessarily follow. I was wondering at this point in time what kinds of things did you have to do in order to try and make the routines portable?

TRAUB: This is described in the paper with Morven Gentleman. We talk about software that was as portable as possible, and furthermore, we built a system which could go from one language to another automatically. It's all described in a paper by Morven Gentleman and myself. [*The Bell Laboratories Numerical Mathematics Program Library Project*, Proceedings of the 1968 ACM National Conference, 485-489].

HAIGH: It seems, as I read it, that it suggested that the library would include routines written in different high level languages and provide a mechanism to allow programming in one language to call a library written in another language. So, that, you might not remember, but do you have an idea of whether that would have been a very novel idea at that point in time?

TRAUB: Portability, I think, was fairly novel. I think there were really four novel things about the project. First, construct a library of important modules; secondly, portability; thirdly, the refereeing and certification process. And finally, documentation. Have you ever seen one of these Bell Labs reports we published as part of the project?

HAIGH: No.

TRAUB: Each report is about half an inch thick.

HAIGH: Was the idea that people would take these routines and use them as-is, without any modifications, essentially as a black box?

TRAUB: That was the intention. This project was done around 1968-1969, and I left Bell Labs in 1970, to go to the University of Washington and then to Carnegie-Mellon to become the head of the CS department. Morven Gentleman accepted a position in Canada around the same time. So, the project was only in existence for a relatively short amount of time.

HAIGH: Yes. So this might be an appropriate moment to ask you to compare this project with the projects that followed in the 1970s, such as LINPACK and EISPACK. Those efforts did produce software which was widely used, continuing in some areas to the present day. Would you say that those later project were perhaps more specialized, but comparable in intent?

TRAUB: Oh, much more ambitious because these were huge projects. For instance, Jack Dongarra, who I'm sure you're interviewing –

HAIGH: Yes.

TRAUB: He has spent a substantial amount of his professional life on this. There's also a project at Oxford called the Numerical Analysis Group (NAG). I think our claim to fame was that we were probably one of, and perhaps, the first such project but these others were much more sustained and were done by people who did this for a substantial part of their careers.

HAIGH: That's true. Although I think there's one sense in which your project might have been more ambitious, because EISPACK for example, which had been the first of those projects, was focused on eigenvalues, whereas, from the list of things that the Bell Labs library intended to include it seemed a broader project.

TRAUB: That's right, but we only did about three: linear algebraic equations, ordinary differential equations, algebraic eigenvalues and eigenvectors. So we only did a number of these projects and then Morven and I moved on.

HAIGH: So, as it materialized it was more limited in scope. Would you say that if you had known how much work it would be, or how much effort would be required, that the project might have been less ambitiously scoped to begin with?

TRAUB: It was partially a research project and part utilitarian. We were testing various items: the translation between languages issues, the portability issue, the certification issue, and so on. I think I mentioned in a report that we didn't realize how much work it was for each one of these.

HAIGH: Yes.

TRAUB: But I find that's generally true. I've had a lot of experience writing papers, writing reports and yet with all my experience, it always seems to take a factor of two to ten times as long as I think it's going to take, always. In the paper on the Bell Laboratories Numerical Mathematics Program Library Project Morven and I drew up a tentative list of problem areas. Included were problems in numerical linear algebra, differential equations, curve-fitting, zero-finding, minimization, quadrature, random number generation, etc. We also listed the criteria for a program to be in Library One, which we referred to as the first level. A summary of the three criteria is: Each program covers an important area, it has been certified, the calling sequence and diagnostic messages have been carefully designed so the program is easy to use. And then I moved on as I tend to. I work hard on some things and then after a few years move onto the next challenge, something new.

HAIGH: Yes.

TRAUB: I love to do that. I still continue to do that. The wonderful thing about computer science is for 50 years one can do that.

HAIGH: …but were there any practical problems that made the work difficult that were related specifically to it being one of the first attempts, perhaps the first attempt, to produce a high quality standardized numerical analysis library? You know, were there things you found that made it harder produce quality portable software than you thought it might be?

TRAUB: It really took longer than we expected, each module that we built became a major project.

HAIGH: The writing or the testing?

TRAUB: The testing and the documentation. Phyllis Fox should be able to tell you more. You can also try to reach Peter Businger. I hired him as a fresh Ph.D. from the University of Texas and he did a lot of this work. Then he decided he wanted to become a lawyer and went to work at Baker Botts, and was actually involved in some of our patenting work on computing financial derivatives in the 1990s. He is now retired from Baker Botts which is a very large law firm in Rockefeller Center and they might be able to tell you how to reach him.

HAIGH: OK.

[Start of Tape 1, Side B]

HAIGH: Now, you've mentioned a couple of people who were doing the work on the project. You co-wrote the paper with W.M. Gentleman.

TRAUB: A statistician. A student of John Tukey, a very smart guy.

HAIGH: And what was his role in the project?

TRAUB: I had the idea of certification and portability and Morven was responsible for trying to translate automatically between languages. He went to a university in Canada.

HAIGH: Now, how about getting the software that was produced out into use. How was that to be accomplished?

TRAUB: That's an excellent question. The output were those extensive Bell Labs reports.

We thought of it as journal publication. The "journal" was titled *Numerical Mathematics Computer Programs.* It was numbered by Volume and Issue. For example, Vol.1, Issue 1 was authored by Peter Businger. Its title was "NSEVB-Eigenvalues and Eigenvectors of Nonsymmetric Matrices".

HAIGH: One of the ways in which the project is, I think, interesting historically is that it appears to have many characteristics in common with the later open source software projects. Particularly this idea (which it appears, wasn't fully achieved), that would be something akin to a call for algorithms, or a call for software, and that submissions would be received and reviewed and integrated somehow. That actually is a contrast with EISPACK, where they knew from the beginning that they would be writing the software and disseminating it to the world, they never had the idea that there would be a two-way process. How did you get the algorithms that were out there? Did you call for people to send them?

TRAUB: Of course when one works in a scientific field, there are people networks, and people must have known I was interested. I don't recall how we communicated, this was before e-mail, before the web, before we were sending stuff around electronically. So it would have been tape or card decks.

HAIGH: *Numerische Mathematik,* has been mentioned as an important venue for algorithms. Do you have anything to say about that?

TRAUB: My recollection is that *Mathematics of Computation,* which earlier had been called *Mathematical Tables and Aids to Computation* contained software.

HAIGH: Yes. There is one more question, that you alluded to earlier. Do you have any recollections of cases in which the software was definitely used outside of Bell Labs?

TRAUB: I know that the Jenkins-Traub software was indeed used but of course I had a special proprietary interest in that. The Bell Labs reports were certainly valued.

HAIGH: Do you have any sense of resistance to this? I mean you gathered a wide range of algorithms, found most of them wanting, put a little work into implementing what you thought was the best one, tested it and made it available. And it seems that it is not clear that it was widely adopted.

TRAUB: I was very much a part of that community, invited to various meetings, and in touch with various people when I was active in the field, and I certainly did not get any sense of resistance or unhappiness.

HAIGH: I'm interested in this because it seems there's kind of two models that you can approach the subject with. One, and it seems to be the thing that you had in the back of your mind, is that it this aggressive testing would be something more like the aggressive testing that you give to someone who is defending a Ph.D. thesis or who has written a paper. You ask them some hard questions, and then when the answer them, you know it's good and you publish it, and (unless there's some awful error discovered in it) it's finished and you all move on to the next thing. It's out there in the world, and you hope people use it, but you may never know for sure if they do or not. The alternative is the approach, which a lot of people in many areas of software seem to be using, where you write it and you do your best to test it but you never expect to discover everything that might be wrong with it. So you put it out, you release frequent updates, and you expect some feedback and incremental improvement. The impression that I'm getting is that the kind of mental model that you were applying to it was more the mental model of a publication.

TRAUB: I think that's very perceptive. As I said before it was something that Morven and I did for a couple of years and then we moved on.

HAIGH: Alright, so moving on somewhat from that, the special case of the production of the software that you're more familiar with is with the Jenkins-Traub algorithm for polynomial zero finding.

TRAUB: I was very pleased that it was regarded as the best piece of software for this problem and that it was adopted by Mathematica, IMSL, etc. Every now and then I still get e-mail asking me about this algorithm or saying that it gave good results.

HAIGH: Now in the previous interviews you've spoken, I think, about how you came to create the algorithm, what it was for, and the mathematical dimensions of it, but you didn't talk about it as a piece of software. So, let's begin with the question of whether was this was an algorithm that would only be practical with an electronic computer, so that with mechanical punch card machines or desk calculators it would not have been something that people could have used?

TRAUB: In the book "Numerical Recipes" (by Bill Press, et al.) there is a statement "The Jenkins-Traub method has become practically a standard in black-box polynomial root-finders, e.g., in the IMSL library. The method is too complication to discuss here, but is detailed with references to the primary literature, in Ralston and Rabinowitz". I think that answers your question. This could only be done on an electronic computer.

HAIGH: So, philosophically that's interesting because this method can't be realized except in conjunction with some piece of software.

TRAUB: I never thought of it that way but you're absolutely right.

HAIGH: So when you were originally working on it you must have had to write some code. Now, did you put a lot of effort into this code, or did you just rush, you know, to produce something that would show that the thing worked as quickly as possible.

TRAUB: Let me give you a little history. Starting in 1959 I became fascinated by iterative methods for solving nonlinear equations. I did analyses predicting the rate of convergence and then I programmed these examples to see how fast they actually converged. I'll give you an example of how fascinated I was by this. When my then wife, Susanne, was starting her delivery process for our first daughter, I would run up to Bell Labs to get one more run in (In those days it was all batch processing so I had to hand in all the cards and return later in the day to get the results.) and then took her to the hospital. As I look at the papers I published in the mid 60s there are sections on numerical testing, all of which I did by myself.

HAIGH: Yes.

TRAUB: I remember telling Dick Hamming about the early work on computing polynomial zeros and he says "well, how does it look, have you tested it," and I spent several weeks programming and showing him very nice results. In those days I did the analysis and then I did the programming. Today, my students do the programming. But I strongly believe you learn by building the software and computing and are often surprised and you learn from that. That has always been my philosophy; the only change is I did it myself in the 60s, and now I have other people, typically students, do it for me or with me.

HAIGH: So the way it was presented in the paper is: here's the algorithm, here's the test of it, look it works well, here are its characteristics.

TRAUB: It's the algorithm, the character of the analysis, proof of global convergence which is the big thing. But then Appendix A, Numerical Examples, goes on for three pages. ["A Class of Globally Convergent Iteration Functions for the Solution of Polynomial Equations", Math Com;, 1966]

HAIGH: So, that's in the appendix. Now it's often suggested by sociologists of science that the way a scientific paper presents something is not the same that the work is actually done, you know, that it's a very formalized presentation. And a minute ago when you were describing this it sounded very experimental, that this kind of mathematics was becoming more experimental, so that you would be using a computer as a kind of laboratory, testing, testing, writing code. So in reality was the algorithm you finished up with exactly the same one that you began with, or did you make any changes to the algorithm itself as a result of your experiments with the computer?

TRAUB: As I think about software, I realize that my thesis was a huge software project. The point of the thesis was that experimentalists were ahead of the theoreticians for this particular problem and I was trying to check whether the theory could predict what the experiments had found. That was in the 50s. In the 60s, I would certainly test. There would be the theory first, but then testing. But you're asking, "are there examples where the algorithm got changed because of the numerical results". Nothing was changed in the polynomial zero finding. The testing was a way of checking whether the predictions made in the theory were correct. So the paradigm was you did the analysis and you used the testing to validate the analysis and also to convince your readers. Any referee or reader would want to see some numerical data to support your claim.

In the 1990s our work on finance was different. That work, which we can talk about later, was different because it started with experimentation and the results of the experimentation were totally different from conventional wisdom. The attempts to provide a theoretical explanation came later.

HAIGH: And it that the case through the 1960s and the 1970s, and the other algorithms that you worked on and discussed in the earlier interviews, that the computer really was just being used to test and confirm things which mathematics had already drawn you towards?

TRAUB: Yes.

HAIGH: Thanks. So, on the other hand the production of code was very much part of the research and something you were doing personally. Did you consider yourself a good programmer?

TRAUB: Early on, I really invented several novel techniques for my thesis. I prided myself that I always did my own programming.

Earlier I told you about the macro compiler that counted machine cycles which I wrote with Phil Sherman. With Stan Brown I developed MERCURY, a computer-aided system for the selective dissemination of Bell Labs reports to employees. ["Mercury: A System for the Computer-Aided Distribution of Technical Reports" with W.S. Brown, JACM, 1969] Incidentally, we tried to extend these ideas to scientific journals worldwide. We argued that researchers should be able to get titles, abstracts, or papers published anywhere that matched their profiles. This was probably quite novel in the mid-sixties. ["The Future of Scientific Journals", Science, 1966, with W.S. Brown and J.R. Pierce.]

Let me tell you about a small programming achievement. We did a lot of software development at Bell Labs and Doug McIlroy presented me with a problem concerning our assembler. The problem was the following: in comes some code, how do you parse it and decide how to deal with it, which is the heart of the assembler. I came up with a beautiful, if I do say so myself, some 25 lines of code which did the job elegantly and fast. Have you heard the name McIlroy?

HAIGH: Software factories? I believe that from the NATO Software Engineering conferences, he's the guy who's credited with the idea of a "software factory" that would somehow assemble code from standardized components.

TRAUB: Exactly. Doug McIlroy is certainly a programmer's programmer. In fact he used to tell the story that someone asked him at a meeting, "well, what do you do," and he said, "I'm a programmer." But Doug liked my module so much, this was probably 1960 or 1961, that he took it and included it in the Bell Labs' assembler.

HAIGH: So you've mentioned that many researchers would not have done their own programming. Among grad students or other people who would do their own programming, do you think that this interest in elegant type code was shared or that to many people programming was just a chore?

TRAUB: I don't know. I loved to program. I had a summer job on Long Island in the 50s, at Sperry Gyroscope, and we were working on an airborne computer. The problem was that neither the hardware nor the software were ready. We were responsible for the software, so I was sent down to Patrick Air Force Base in Florida (near what's now the Kennedy Space Center), to do the programming and to test it. I loved programming, and I was quite good.

I was quite a good chess player, captain and first board of the Bronx Science chess team. It's some of the same skills, I think.

So from 1955 to maybe 1968-1969, I really loved to program. Let me tell you how I got hooked on computing. For my thesis I worked for six months starting from a mathematical model of the helium atom and writing a program to compute the energy and other parameters of the atom. I took the cards from the IBM 650 and loaded them on the printer. The printer started spewing out approximations to the ground state energy of helium. I was using a variational principle which means I was converging down to the ground state energy of the helium matter. Watching, after the six months of work, the numbers rolling off the printer, and seeing that the initial numbers approximated the experimentally measured ground state energy of the helium atom good to four places. That was the moment. That was very significant. (I still get chills down my back talking about it because it changed my life.) What I loved about the computer is you got the feedback, you worked and worked and out came the numbers and you got this feedback. So now I use the computer for the web, for e-mail, power point presentations, and so on, but I don't program anymore. I've always had facility with numbers, so it was very natural in a way, but the programming, that's what got me interested in computer science.

HAIGH: So, in that first paper you'd already outlined the algorithm, you'd proved that it worked and you'd presented data supporting this, now it seems that in conventional terms that would be it, wouldn't it? You know, there's a new method, we've shown it works, here it is. So, why was it that then you continued to produce a number of other papers and produced some different software?

TRAUB: The first paper I wrote on the subject, "A Class of Globally Convergent Iteration Functions for the Solution of Polynomial Equations", was published in *Math Comp* in 1966 and one of the nicest compliments I ever got was for this paper. I wrote in the cover letter that I thought this was a very exciting result and the referee's report stated, "if Traub thinks this is good let's publish it." This was very flattering. I was a Visiting Professor at Stanford and I met a student named Mike Jenkins and we started working together. My 1966 algorithm was a two stage algorithm but Mike had this very clever idea of using a shift which lead to a three stage algorithm which was a big improvement over my original two stage algorithm.

My technical life has two parts; sometimes I work on algorithms and sometimes on complexity. This is the algorithm part of my life.

Because Mike was a Ph.D. student (even though he was a student at Stanford, I effectively was his advisor) I was able to ask him to write a portable high quality piece of software which was published in 1972 as "Zeros of Complex Polynomial (C2), Algorithm 419"in *Communications of the ACM*. Then Mike and I wrote another paper on solving polynomials with real rather than complex coefficients. Then we wrote a paper which appeared in the first issue of *Transactions on Mathematical Software* (TOMS), entitled "What Are the Principles for Testing Polynomial Zero Findings."

There are different principles depending on what kind of software you're testing. For instance, statistical testing, testing with hard problems, etc. What I said then, and which I repeated countless times, to generations of students, and I really believe, is you write a paper and ten people read it, you write a good piece of software and distribute it and it will be used by thousands of people. I really stress the importance of software. The reason our software became the standard is twofold. One, the algorithm was good, but perhaps not uniquely good. Two, we had this piece of really good portable software. For example, (this might have been quite novel in the early 70s) our software would test what kind of machine it was on and what kind of software the machine had and make suitable adjustments.

HAIGH: Was this another area where being at Bell Labs and not quite as subject to the need to produce a large number of papers in just the right journals and styles might free you up to work in a slightly different fashion?

TRAUB: I wrote the 1964 monograph and did the work on the Jenkins-Traub algorithm while I at Bell Labs but the paper on testing polynomial zero finders wasn't published until 1975 when I was already at Carnegie Mellon University.

HAIGH: What I mean is, once you'd got the algorithm out there if you'd been in the university at that point do you think you'd have received as much recognition for producing the software and improving it so that it actually becomes widely used, as you would for spending the same amount of time working on a different algorithm?

TRAUB: I've been on a fair number of tenure committees, also on the committee for electing computer scientists to the National Academy of Engineering, and the questions

that I asked in both cases, especially for election to the National Academy, is "what is the great contribution, what's the killer application," (you don't care just how many papers have been produced), and also "what's the impact this person's work has had." And I think being able to show that you have produced an algorithm and then a piece of software that's used all over the world, and that's sort of the standard, would count for a lot for election to the Academy, that wouldn't be enough, but it would be a plus and certainly a significant plus for say tenure.

HAIGH: Right. And what time period are you talking about when you describe those kinds of issues being a significant factor: 90s, 70s, 80s?

TRAUB: Well, you mean producing a piece of software that's widely used?

HAIGH: Yes. You were saying that from your experience making those kinds of evaluations of people that contribution to software is something that should be taken seriously. Now that's presumably changed over time so I was wondering how this happened?

TRAUB: No, that's still true today. I mean the criteria, for say, election to the Academy, the question you ask (not always followed in practice but at least in principle) is not how much has the person done, but how has this person transformed the field, what's the impact, and we ask the same for promotion to tenure. So that's always the question. Today if, yes this person is responsible for this, and furthermore this software is widely used, would still count very heavily. I don't think that's changed.

HAIGH: You don't think it's become more important over time? I wasn't suggesting that's it's become less important, I was wondering if perhaps it wouldn't have been as widely recognized in the 1960s as it would be today.

TRAUB: Or early 70s, when a lot of this stuff on the JT algorithm was done. There was a belief for a while that if you were doing systems work you might have trouble with the university tenure committee –

HAIGH: Yes.

[Start of Tape 2, Side A].

TRAUB: It was certainly not a problem at Carnegie where the department was known for its work on systems. At Columbia it has also not been a problem. In fact the department has never lost a tenure case at the university level and we have tenured a substantial number of faculty. When I was elected to the National Academy of Engineering in 1985 one of the first people I nominated was Bob [Robert E.] Tarjan, from Princeton, who was known for algorithm development. I had some concern that the people might ask whether this was engineering and I was ready to argue that, in the late twentieth century, the design of algorithms was as much engineering as the building of bridges and roads. I was ready with the argument, but it wasn't necessary. He sailed through.

HAIGH: Well, that's actually bringing these questions of disciplines and where this fits, which I will return to with a follow up question after one last question about algorithms. So, after these high quality implementations of your algorithm had been produced, with a version for the special case you mentioned, and they were more efficient and portable, and so on, you said this code was widely used. Now did users ever suggest to you any modifications or improvements in the software?

TRAUB: I would sometimes get a letter or an e-mail either saying something laudatory, or saying it had failed on the following example, but I don't recall that we actually modified the software because of something we heard. I knew that the software was being used by libraries such as IMSL and Mathematica and that was important to me.

HAIGH: And you never would have made any attempt to copyright the code?

TRAUB: No. It never occurred to me in the 60's and 70's to copyright or patent anything. Much later, in the 90's we did patent some of our discoveries in mathematical finance and we used that money to support our research.

HAIGH: But copyright is a legally separate class of intellectual property protection from patenting. Prior to 1976 you actually had to assert copyright and if you didn't it was public domain from the beginning, and you had to include the copyright symbol and the date. (After 1976 copyright was automatic, and explicitly applied to computer code).

TRAUB: I could make a general comment about credit for scientific work. What does a scientist work for? Certainly the scientist's reputation is important and how you acquire your reputation is by giving away your work. I'm eager to publish, I'm eager to give the work away and it would never occur to me to copyright or patent my discoveries. It was only at Columbia in the 90s that we patented several of our discoveries on computational finance. We always gave the computational finance software to anyone who was an academic and not using it for commercial reasons free of charge. The only people that we licensed the software to, or asked them to pay for the patent, were Wall Street companies.

HAIGH: I want to understand where this fits in a disciplinary context. For example, your career, your Ph.D. was from the Committee for Applied Mathematics –

TRAUB: Because there was no computer science –

HAIGH: Right, and at Bell Labs you were in the mathematics department–

TRAUB: Because there was no computer science, yes. [Laughter]

HAIGH: So –

TRAUB: The first department I was ever in was the department I chaired at Carnegie.

HAIGH: So… you've always been a computer scientist at heart but the discipline just didn't exist at that point!

TRAUB: I got hooked in 1955.

HAIGH: Right. And how would you say the relationship has been with mathematics?

TRAUB: In the United States most mathematicians were simply not interested in computing. Mathematicians could have probably played a substantial role in the shaping of computer science, and in Germany they did, and there are many more mathematical requirements for computer science students in Germany than in the US. That's also the case in other countries such as Poland.

Electrical Engineering was much more interested in computer science. Generally computer science is separate but most often in the United States it's in an engineering school. (Of course, there are some important instances of Schools of Computer Science or Colleges of Computing.) It wasn't clear at one time where computer science would wind up. For instance, at Stanford it started in Arts and Sciences and moved to Engineering. At Columbia there was an effort in Mathematical Statistics, an effort in Electrical Engineering, and finally it wound up as a separate department in Engineering. At Carnegie Mellon, it was in the Science College because my predecessor, Alan Perlis, wanted it in there. In summary, computer science in the United States is most often in the School of Engineering, and the discipline has not been heavily influenced by mathematics departments.

HAIGH: Right. And after you left Bell Labs and went to Carnegie Mellon and Columbia, you were chairing those departments, and were very much involved in establishing curriculums, and what sociologists of science would call boundary work, deciding what fit into a field properly and what didn't. I think it may also be true to say that scientific computation does not feature as prominently in computer science now as it did in its earlier days.

TRAUB: True.

HAIGH: So can you talk about how that's changed over time, and whether you personally have taken any strong positions, or tried to influence this?

TRAUB: Yes, yes, and yes. At the beginning numerical analysis was very strong in computer science. I bet if you look at the early presidents of ACM a fair number of them were numerical analysts, George Forsythe is one that comes to mind.

HAIGH: Householder –

TRAUB: Alston Householder, very much so. Then over time the influence of numerical analysis became less as other areas developed. There's been a fair amount of pressure, which I've argued against, to have computer science students take less and less calculus. I'm a member of the academic committee in this department and we've just been redoing the undergraduate program. We'll continue to require calculus, but there are people like Tony Ralston, who's actually a very prominent numerical analyst, who has argued that computer scientists don't need calculus. I have heard Donald Greenspan argue the same thing. In this department we require three semesters of calculus and

courses in scientific computation and numerical algebra. Cornell, where the chair is Charles Van Loan, has similar requirements. My answer to the question "Why calculus?" is that if you don't know calculus you can't talk to people in physics, chemistry, economics, finance, and most of engineering. So I argue that the reason students should study calculus is that computer science interacts with all these other disciplines. Calculus is the common language.

HAIGH: Yes.

TRAUB: The field in which I work is the computational complexity of continuous problems. I think this work is the foundation of numerical analysis as well as being part of theoretical computer science which deals mainly with discrete problems. However, in numerical analysis that is a minority view because the numerical analysts don't see the value of studying computational complexity. On the other hand many theoretical computer scientists don't see the value of studying continuous problems.

HAIGH: Now you personally are teaching undergraduate courses in this area.

TRAUB: We don't have separate faculties for teaching undergraduate and graduates at either Carnegie or Columbia. Right now I am teaching an undergraduate course on scientific computation and a graduate course in numerical algorithms and complexity.

HAIGH: And how would the content of an undergraduate course differ now from perhaps twenty-five or thirty years ago?

TRAUB: For example, in the undergraduate course I include topics such as primality testing, cryptography, Moore's Law, quasi- Monte Carlo methods for computing high dimensional integrals, chaos theory, weather prediction, and global change. Thirty years ago I would have covered the calculus of finite differences and various formulas for performing interprolation. We don't use tables anymore so anything having to do with mathematical tables has been dropped. So, I keep adding new topics, a recent one being quantum computing. I talk about Moore's law, and why we may see the end of Moore's law in 10-15 years.

HAIGH: So that reflects developments in the field?

TRAUB: Oh, very much so. Every time I teach I drop some things and add other things.

HAIGH: And would students be learning basic algorithms and how to write something from scratch here, or would you actually be using some of the libraries and packages that are now available?

TRAUB: I give several homework assignments where the students use libraries. Another change - at one time a listed prerequisite of the course would be programming, now I don't list that anymore because I assume everyone knows how to program.

HAIGH: Okay. Now returning to the mid-1960s, and the question of the relationship between these things and computer science. You were involved in what became the special interest group SIGNUM.

TRAUB: In fact , I started SIGNUM.

HAIGH: Right. Why did you start it?

TRAUB: I like to create things and SIGNUM was one of the first organizations that I created. Incidentally, the organization was originally called SICNUM (Special Interest Committee for Numerical Mathematics) but ACM decided to standardize on SIG. The first editions of the newsletter had SICNUM on the front cover. At the same time, I was Editor for Numerical Mathematics for the Communications of the ACM. There was quite a bit of emphasis on mathematical software in the early issues of the newsletter. There were about a dozen people on the SIGNUM Board and because ACM and SIAM were the two leading societies in the area of numerical mathematics, I chose about a half dozen people who were prominent in SIAM and a half dozen people prominent in ACM for the Board.

HAIGH: And were there any other individuals who were particularly involved with the establishment of the group?

TRAUB: I got permission to start SIGNUM from George Forsythe who was the ACM President but I don't remember anyone else.

HAIGH: So at this point in time computer science already existed. Why did you create it within ACM rather than as a freestanding group or as a part of one of these other associations?

TRAUB: I certainly wasn't interested in starting another professional society and ACM seemed like a natural home for SIGNUM. I was also active in SIAM, indeed SIAM asked me organize a meeting on parallel computation in the early 70s.

HAIGH: Was the advantage of being a SIG was that you could use ACM resources and communication channels and so on?

TRAUB: Yes.

HAIGH: So what programs did SIGNUM pursue?

TRAUB: We had a subroutine certification group chaired by W.J. Cody which collected tests to be used for certification. We published a newsletter, and we had sessions at the national meetings. In those days the national meetings of ACM were very important scientifically. Also in those days, the IFIP meetings were important. I remember speaking at the IFIP Congress in 1965 and there were a couple hundred people in the audience, including many of the top numerical analysts. Now we have a triennial integrated meeting organized by a number of SIGs and these are very useful for networking.

HAIGH: But if at this point the national meetings were still small enough that you could present more work there, why was there a need for an interest group?

TRAUB: You ask good questions. As a special group within ACM we could organize our own sessions at the national meetings.

HAIGH: Okay. Well can you remember your impressions of whether SIGNUM was a success? Did it, in the early days, grow rapidly, was it enthusiastically received?

TRAUB: I think it was very successful. We got good submissions for the newsletter and many of the leading numerical analysts were members.

HAIGH: One last question. Other than the anecdote you mentioned, do you recall anything about relationship with the National ACM? Support that you received, or frustration caused, anything like that?

TRAUB: No, it went very smoothly. I have been a member of ACM forever and was made a Fellow of the ACM. I have something unique that I can show you here on my wall. It's a certificate naming me as an ACM Fellow. You notice it is signed by Gwen Bell who was the president of ACM when I was elected. She is the wife of Gordon Bell who is considered as the father of the minicomputer. You will see that on the certificate it says "Initial Recipient" and there's a story attached to that. I was made an ACM Fellow the first year in which Fellows were designated. Fairly recently I contacted ACM and said that after the first year, certificates indicated the year the person was made a Fellow but that there was nothing on my certificate to show that I was selected in the first year. They replied that they weren't going to do anything for the first year recipients but if I mailed them my certificate they would imprint "Initial Recipient" right on the certificate so that's unique. I'm probably the only person who has a certificate that says "Initial Recipient".

HAIGH: Okay. Alright then, that's the conclusion of the first part of the interview conducted on the 19th of March, 2004.

[Start of Tape 2, Side B]

HAIGH: Traub interviewed by Haigh. Session 2, 20th March, 2004. Continuing from our discussion yesterday of specific aspects of the early part of your career, we can now turn to discuss how your career as a whole has developed over the twenty years since the previous three oral histories were recorded. I understand that your interest in information- based complexity has continued to underpin your intellectual work during this period, so, I wonder if you could begin just by describing how this research agenda has proceeded, and the extent to which a new field has developed around it.

TRAUB: Just to recap. This really started in 1959, when somebody walked into my office in Bell Labs with a specific integral he wanted to evaluate, and I could think of many ways of computing it. I looked at the literature to see if there was a theory of the best way to solve the problem, that is, an optimal algorithm, and there was nothing. Lather that year I started looking at nonlinear equations, and had the insight that the

optimal order of iterative methods for solving nonlinear equations depended only on the available information, such as the function and its derivatives. That was an important insight. That led to my 1964 monograph, *Iterative Method for the Solution of Equations*. The first sentence of the introduction is "The general area into which this book falls may be labeled *algorithmics.*" Donald Knuth, who among other things, is the great historian of computer science, said that I had coined the word algorithmics to name the whole field.

In 1973, Henryk Wozniakowski from the University of Warsaw, sent me two papers which not only settled conjectures I had made earlier but extended them to the many variable case and even the infinite dimension case, and that was the beginning of the partnership that has now been going for over thirty years. I was so impressed with the papers that I invited him to Carnegie, and currently he is a full professor at the University of Warsaw, and a full professor at Columbia, splitting his time between the two.

In the late 70s, Arthur Werschulz gave a seminar where he used some of the ideas regarding the complexity of nonlinear equations for computing integrals. My reaction was that "there must be an underlying general theory because computing integrals is so different from solving nonlinear equations. Henryk and I always have a list of research questions and we call the list of questions regarding this general theory the "S" list for Special. In the late 70s we focused on developing this general theory which in 1980 led to the monograph *A General Theory of Optimal Algorithms*. It was really a general theory of optimal algorithms for continuous problems. In those days we called the subject analytic complexity to distinguish it from a very active area called algebraic complexity, which I also worked on, and which studied how fast could you multiply two matrices, how fast could you solve linear systems, how fast could you form the composition of polynomials, how fast could you do the discrete Fourier transform, etc.

By 1983 we published *Information, Uncertainty, Complexity*. In this last book we called the field "epsilon complexity". The motivation was that you cannot solve a continuous problem exactly; you can only solve it within an error epsilon. When my wife heard me refer to epsilon complexity and when I mentioned that epsilon was something small, her response was, "Oh, well that makes it sound pretty trivial". I was looking for a new name and I found it at Berkeley.

In order to be able to focus on research I spent summers, starting in the mid 70s, at Berkley talking to people like Richard Karp and Steve Smale. (If I was at Carnegie people would check with me on many things. When I was at Berkeley my Carnegie administrative time dropped to about 50%.) I discussed with Karp what this field might be called and he suggested "information-based complexity". This seemed like a good name since it's a branch of computational complexity for which information is central. So, the 1988 monograph with Greg Wasilkowski and Henryk Wozniakowski is called *Information-Based Complexity*. That differs from earlier work in that we now study the complexity of continuous problems in various settings: worst case, average case, randomized, asymptotic, etc.

HAIGH: And in these later works it continues to be true that you're addressing the fundamental complexity of the problem on an information basis, rather than the complexity characteristics of a particular algorithm?

TRAUB: Complexity always refers to a problem although the word is sometimes misused. When I look at the physics or chemistry literature people write complexity when they mean the cost of an algorithm. I find the same in the numerical analysis literature. In that way it's like the word chaos; it also has many meanings. Complexity has many different meanings but in theoretical computer science we talk about the cost of an algorithm and the minimal cost of any possible algorithm is called the complexity of the problem, so this is a property of the problem but not of the algorithm.

HAIGH: Right.

TRAUB: A major new focus of our work started in the 90s . The theory of information-based complexity is carried out in abstract linear spaces but the applications in science and applied mathematics are to multivariate problems. We had established that if you insist on a worst case assurance of a good answer many problems are exponentially hard in the dimension of the problem. The problems that occur in practice in economics, physics, chemistry, finance and engineering involve hundreds or thousands of variables, so you would be dead if you had a problem that was exponentially hard in the number of variables. So, a big issue is the "curse of dimensionality", which was coined by Richard Bellman in the 50s. Complexity theory didn't exist at the time but he just noticed that for certain problems that he was studying the difficulty of the problem grew rapidly with the dimensionality of the problem; he called that the curse of dimensionality. In computer science we say the problem is intractable. I believe the way you have an impact in science is to solve a problem that people care about. Dick Hamming taught me that in the 60s and it didn't take much to convince me of that.

A good example is information theory because Claude Shannon solved a problem that people really cared about. Incidentally, I once said to Claude Shannon why is it called information theory; it should be called the mathematical theory of communication. (I wanted information theory as the name of what I was doing.) Claude replied "you're absolutely right". When I looked at his famous 1948 book I saw that he used the phrase information theory.

I'll tell you about an application which has had a great deal of impact. This also ties into the software issues we discussed earlier. Around 1991, I had a student named Spassimir Paskov. Spassimir had never been out of Bulgaria until he came here to study for his Ph.D. He was a very good theoretician having represented Bulgaria in the Math Olympiad. I always like my students to be good in theory, of course, but also to be broader. So I asked him to work on the following project: I wanted him to do a big software project and I asked him to do the software on something that was quite cutting edge in the early 90s. That is, to work on a heterogeneous workstation cluster. Our department had 20-25 work stations from different manufacturers. Furthermore, I wanted him to work on a real problem rather than just a made-up problem. Wozniakowski and I had been working on computing high dimensional integrals. We were contacted by a

man named Irwin Vanderhoof, who had been chief actuary for Equitable Insurance. Irwin said a lot of problems in finance and insurance were modeled by high dimensional integrals. Through his contacts at Goldman Sachs, Vanderhoof got us an example of a collatoralized mortgage application. This involved numerical evaluation of integrals in 360 dimensions and the integrands were so complicated that it took a million floating point operation to evaluate the integrand at one point. So it was really important to minimize the number of valuations.

Wall Street believed that you did these integrals using Monte Carlo. I asked Spassimir to compare solving the integrals using Monte Carlo and quasi-Monte Carlo methods. At that time the top experts believed that quasi-Monte Carlo was no good for dimension bigger than about 12.

HAIGH: Why did the problem have so many dimensions?

TRAUB: Because you're dealing with baskets of thirty year mortgages and you make the assumption that interest rates, prepayment of mortgages, etc., change monthly. 360 is the number of months in thirty years and the 360 variables represent 360 months.

HAIGH: Right. Because, unlike a regular bond, a mortgage can be paid off early.

TRAUB: Paskov reported to the research group that quasi-Monte Carlo consistently beat Monte Carlo. We were really surprised because Wall Street believed that Monte Carlo was superior and the world experts believed that quasi-Monte Carlo was no good for dimension of about 12. We published the first paper on these results in the *Journal of Portfolio Management*.

We started making presentations to leading financial institutions in New York such as Morgan Stanley, Citibank, Goldman Sachs, etc. Generally people just didn't believe what we were telling them, that quasi-Monte Carlo always beat Monte Carlo for finance. Columbia got several patents on this work. We gave the software free to academic researchers and licensed it to commercial enterprises. Spassimir got his Ph.D. and went off to work at a number of financial institutions.

Here is an amusing story concerning Spassimir. While he was at Columbia he dressed as a student but after graduation he was working at a bank and when he visited Columbia he was dressed as a banker. Each year the American Mathematical Society chooses a number of the most significant mathematical ideas and Barry Cipra, who is a very good writer, did an article called "In Math We Trust" and in this article he has two pictures of Spassimir before and after.

After Spassimir left, Anargyros Papageorgiou continued that work at Columbia. Anargyros is a research scientist who is still part of our group and is now working on quantum computing. We had the results of Spassimir's experiments but no theory as to why quasi-Monte Carlo was so superior to Monte Carlo for financial instruments. This kicked off a whole cottage industry of people trying to explain why quasi-Monte Carlo was so superior to Monte Carlo for valuing financial instruments.

What's unusual about this is that in most of our work we have a theory and we use the computer to test the theory. Here it's just the opposite. Everyone believed Monte Carlo was best but we showed experimentally that quasi-Monte Carlo is far superior to Monte Carlo for financial computations. After we announced these results people started trying to develop a theory to explain the experiments. Notice the similarity to physics. Now, why did we make the discovery? It is that others were working with artificially constructed mathematical integrands while we dealt with a real world financial problem.

HAIGH: So, it's some pattern hidden in the actual data?

TRAUB: I'll give one possible very simple explanation. It might be due to the discounted value of money. That is, you would rather have a dollar from me today than the promise of a dollar from me in ten years due to the interest you can get on the money and due to inflation. Thus, the future value of money is discounted by 5-10% a year. Therefore, the variables representing closer times are much more important than the variables representing times more in the future. Thus the problem is very non isotropic. The number of important variables is less than the nominal dimension of the problem.

HAIGH: Are there any particularly important pieces of research that have been written by other people following up in this area?

TRAUB: It's a whole cottage industry. Many people have proposed explanatory theories. We've continued to work in this area. Wozniakowski and Ian Sloan introduced weighted spaces which are very powerful tools for studying financial and other problems. Also, Papageorgiou and Paskov have looked at "value at risk" using these ideas. Perhaps that's enough on finance.

About three or four years ago I got interested in quantum computing and what drives that field is the following: In the early 60s, Gordon Moore observed empirically that the density of features on a chip were doubling every one to two years (he kept changing his mind between twelve months, eighteen months, two years) and he had the great insight to say if this continues it's going to revolutionize computing. This has continued for forty years. The projections are that within ten or fifteen years we'll be down to single atoms. At that point Moore's Law using today's silicon technology will come to an end.

Dick Feynman observed in 1982 that to solve problems in quantum mechanics perhaps you needed a quantum computer. That is, a computer that operated on the principles of quantum mechanics. He conjectured that many problems in quantum mechanics could never be solved on a classical computer. Similar work was done in Russia in 1980 by Manin. Pioneering theory was done by David Deutsch at Oxford in 1985. In 1994 Peter Shor, who was then at AT&T Labs, showed that on a quantum computer you could factor large integers in time that was polynomial in the number of bits. We don't know the classical complexity of this problem but the best classical algorithm known is not polynomial. RSA and other cryptographic codes are based on one way functions. That is, they are based on the fact that while it is easy to multiply large integers, it's difficult to factor them. Peter's discovery lead to a huge amount of interest from US agencies such as DARPA and NSA. Also various countries such as Australia and Japan have national

efforts. There's also a large European effort. It's a very sexy area which involves primarily physicists but also some computer scientists and mathematicians. There is even the hope that additional understanding of quantum mechanics might come from trying to build quantum computers.

Peter Shor is responsible for me getting into quantum computing. After his colloquium at Columbia I said to him "you have shown us how to solve discrete problems on a quantum computer, who is working on continuous problems?". "No one" he replied. That motivated me to consider solving continuous problems on a quantum computer.

There are two major motivations for working on continuous problems. The first is that many mathematical models in physics, chemistry, finance, economics and engineering are continuous. Examples include partial differential equations, path integrals and the Schroedinger equation. The second motivation is that to know if quantum computers are more powerful than classical computers and, furthermore, to know how much more powerful, we need to know the classical complexity. We have decades of results on establishing the classical complexity of continuous problems. This may be contrasted with discrete problems, such as the traveling salesman problem, where we only have conjectures about the complexity hierarchy.

At about this time, Erich Novak published a seminal paper on the quantum complexity of integration in the *Journal of Complexity*. This was followed by important papers by Stefan Heinrich.

DARPA started a program on quantum computing and quantum communication called QuIST. I contacted Seth Lloyd, who was a pioneer in quantum computing, and proposed that we do a joint proposal to DARPA. So David Cory and Seth Lloyd at MIT and Henryk Wozniakowski and I have been working on quantum computing funded, in part, by QuIST.

There are two major obstacles to building quantum computers. The first is decoherence and the second is having enough qubits to solve problems that we could not solve on any classical computer. Currently the cutting edge is seven qubits. Seth Lloyd predicts that if we have 80 to 100 qubits we could solve problems which we could not solve on any classical computer. That does not include qubits needed for fault tolerance. It's not clear if we can overcome these obstacles but this is a very exciting moment and there are many different technologies being considered in various countries.

What's surprising, I think, is that there's almost no one in numerical analysis working on quantum computing. And that is similar to what happened with parallel computing. Parallel computers became available around 1971-1972. When the technology changes you have to think differently. One had to invent new algorithms. I was giving talks in 1971-1972, saying, "oh here's an exciting new area and here's why". After one seminar a very prominent numerical analyst said to me, "Joe, you don't think we're ever actually going to use these devices?". About ten years later he jumped on the bandwagon and started working on parallel computers. So I see what's happened in quantum computing as a replay. I hope the very good researchers in numerical analysis and scientific

computing will soon start to explore the challenges of quantum computing. I just attended a Gordon Conference on quantum computing and the attendees were primarily physicists, some theoretical computer scientists, some chemists and some mathematicians. That's one of the things that makes quantum computing fun; it involves people from many different disciplines.

I'll return to information-based complexity. We have a wonderful international community. We usually have about two international meetings a year. We're just planning the next one at Schloss Dagstuhl which will be our eighth. We're inviting about 80 leading people from around the world. I started the *Journal of Complexity* and many papers in the *Journal* are on information-based complexity. We started with 300 pages and two issues in 1985, and now we're publishing 1000 pages in six issues.

HAIGH: So, how many people would you say actively research in this field now?

TRAUB: Really doing research, a few hundred all over the world.

[Start of Tape 3, Side A]

TRAUB: You have to be an expert in a number of different fields to work on information-based complexity. You have to be strong in mathematical analysis, approximation theory, computational complexity, information-based complexity, etc. For some of the work in information-based complexity you have to know low discrepancy theory which is part of number theory.

HAIGH: And with Wozniakowski you've continued to publish books in this area.

TRAUB: No, the latest book was with Arthur Werschulz, in 1998. I'll give you a brief history of how that came about. Each year the Accademia Nazionale Dei Lincee in Rome picks someone to present the Lezioni Lincee, which is a cycle of six lectures. I presented them in 1993. Lincee means lynx and they pride themselves on being sharp eyed, able to see what the significant new directions are; one of the early members was Galileo. I chose to present them at Scuola Normale in Pisa which is very strong in physics, mathematics, and computer science. The lectures are the basis of a monograph published by Cambridge University Press. The monograph was to be as non-mathematical as possible. I invited Art Werschulz to join me. Art had been a student of mine at Carnegie-Mellon. He is now a professor at Fordham University and his research life is here at Columbia. The monograph is on the order of a hundred pages with a rather extensive bibliography of some 400 items. Much of the book is devoted to high dimensional problems and breaking the curse of dimensionality but it also includes a bunch of other issues that I'm interested in, such as what is scientifically knowable and the value of information.

HAIGH: And that book would follow the lecture in being aimed at a more general audience?

TRAUB: Yes. And in fact it's in its third printing.

HAIGH: And have your other books been used as textbooks for courses in this area?

TRAUB: The book with Art is used in a graduate course but the other books are research monographs and not suitable as textbooks.

HAIGH: So is there any reason that you use the monograph form, I know in computer science, unlike history, often papers are the standard mode of academic communication and the book form might be reserved more for textbooks.

TRAUB: There's something very satisfying about writing a research monograph. Let me start by telling you the story behind my first book which was published in 1964. The story starts around 1961 when I sent Mario Juncosa, the editor-in-chief of the *Journal of the ACM*, a 140 page manuscript on what I knew so far about this topic. The fact that I sent him a 140 page manuscript as a journal article shows how naïve I was. Mario sent it back to me saying "it will be a long time before I read, this". Shortly before Prentice-Hall had inquired whether there was a book that I would like to write. Incidentally, George Forsythe, was the consulting editor for that series. So I signed a contract with Prentice and then worked for three years on "Iterative Methods for the Solution of Equations". That was a poor title; better would have been "Optimal Iteration Theory". After the book went out of print it was published by Chelsea and still later by the American Math Society. A few years ago I was asked by SIAM whether they could put it back into print but I had to tell them it had already been reprinted by AMS. I derive great satisfaction from the fact that the book has been in print continually for over 40 years. Anyone who writes on this topic always cites it.

I have a romantic idea about writing books. I think it's no coincidence that I am married to an author. Writing a book is something like immortality.

With a book you have the chance to really form a field. The 1964 book certainly did that and the series of books that I co-authored on information-based complexity helped form that field. Furthermore, each paper goes out for refereeing but with a book you're really in control. Of course, we always send drafts of our books to a substantial number of colleagues for their comments.

The various parts of computer science are quite different from each other when it comes to writing papers or research monographs. Much of the material is published as conference proceedings and that can be a problem at tenure time when you have to explain to people in other disciplines why the candidate has few published papers. These conferences can be extremely competitive; for some conferences the acceptance rate can be very low. It would be nice if someone wrote a textbook on information-based complexity; I don't think I will do that. There are certainly a number of people who could do an excellent job but they are all having too much fun writing research papers or research monographs.

HAIGH: So, you've mentioned the Santa Fe Institute, now they're very much associated with the idea of complexity so perhaps you could talk about your relationship

with them and about different, how much the two ideas of complexity have to do with each other.

TRAUB: The first time I visited the Santa Fe Institute was in 1990 when I was invited to give a week long series of lectures at the summer school. Students competed to be able to attend the summer school and it was a very interesting mix of students. Henryk Wozniakowski and I gave the lectures, which were published in the 1990 lecture series. At the time the Santa Fe Institute was housed in a convent belonging to Cristo Rey Church. The Sisters didn't need the space and rented it to the Institute. There was even some discussion as to whether my wife, Pamela McCorduck, would write a book about what the Institute was trying to do. The Institute was started primarily by people from Los Alamos National Labs, such as George Cowan, who was the first president. There was some discussion as to whether degrees should be granted and the decision was made not to have degrees. There would be a small core group of people who would be there for longer periods together with lots of visitors. After a short time at the convent the Institute moved to some offices on Old Pecos Trail. I loved coming to the Institute and used to say that I was in a continual state of hyper stimulation because there were such interesting people working there. They included Brian Arthur in economics, Stuart Kauffman, who won a McArthur, in theoretical biology, Phil Anderson and Murray Gell-Mann, both Nobel Laureates in physics, Ken Arrow, a Nobel Laureate in economics, and Chris Langton who was a founder of artificial life. Interaction among the people at the Institute was highly prized.

I remember a discussion about chaos and complexity and Seth Lloyd (the very same Seth Lloyd who is my collaborator in quantum computing and is a full professor at MIT) said he could identify some 30 odd types of complexity and that the word by itself didn't have meaning; it had to be proceeded by an adjective. So, for example, the field in which I work is computational complexity which studies the minimal computer resources needed to solve a problem. One of the big themes at the Institute is how does something complicated derive from something very simple? DNA is a good example of that.

HAIGH: Emergence.

TRAUB: John Holland has a book on emergence and that is one of the major themes. It continues to be a very stimulating place and one of the things that I like about it is that you've got economists, physicists, archeologists, sociologists, etc. etc. We have continued the tradition that there is a small cadre of people that are there for longer term appointments. Most of the people at the Institute are visitors. I'm an External Professor.

Still later the Institute moved to its third and permanent home A former ambassador to Japan had an estate in Santa Fe which the Institute bought.

HAIGH: So is there anything you think in your own work, any idea or theme, that you think you perhaps do differently as a result of having been in this kind of cross fertilizing environment?

TRAUB: It's very broadening and I like the multiple interactions. Another organization that I am involved with is the Global Business Network, GBN, located in Emeryville, near San Francisco. It's a for profit company and you probably know some of the key people such as Peter Schwartz , who wrote "The Long Boom", and Stewart Brand who started the Whole Earth Catalog and the Whole Earth Lectronic Link (WELL). Their business is scenario planning which is based on the idea that you can't predict the future so what you try to do is to construct perhaps three or four scenarios, and then try to see which trajectory the world is on. They do this for Fortune 500 companies, for Government agencies, for foreign companies, etc. That's their business, but then they have something called their network, which is supposed to consist of exceptional people, interesting people, and both Pamela and I are members. There's only about two or three couples who are members of the network. The network is broadening.

Pamela wrote a book called "The Futures of Women", which started off as a GBN project on that theme. The book was published in 1995 and that laid out four scenarios about women in the year 2015. One of the four scenarios is called "Fundamentalist Backlash" and we are certainly seeing that. I hope I contribute to the organizations that I am involved with such as the Santa Fe Institute and GBN.

HAIGH: And another institution that you've been involved with is the National Academy and its policy studies of computer science and engineering.

TRAUB: Right. I was elected to the National Academy of Engineering in 1985, and I was asked shortly after to form a computer science board. As I'm sure you know, the National Academy of Sciences was established in 1863, during the presidency of Abraham Lincoln, to give impartial advice to the Government. We try hard to present balanced and impartial advice which is why the Academy's studies and reports are so influential. There had been two earlier attempts to form a computer science board, both of which had failed so this was the third try. Frank Press, who had been science advisor to the president of the United States, and was now president of the National Academy of Sciences, asked me to form a board. The National Academies consist of three parts: the National Academy of Sciences, the National Academy of Engineering, and the Institute of Medicine. Membership is honorific. Selection of a member is very rigorous; in fact one of my nominees, Zvi Galil, was just elected so we're going out to celebrate.

The body that does the studies is called the National Research Council. The Board creates committees which carry out the studies. Often the members of the committee are members of the Academies but this is not necessary.

So I started the Board and what was crucial was raising money, failure to do so is one reason that two earlier Boards had failed. I set out three criteria for us to undertake a study. First, it had to be a matter of national interest; secondly, there had to be one or more customers willing to pay for it, such as a government agency like the National Science Foundation or DARPA, or a company. The reason is that if someone cared enough to pay for it they might pay some attention. The third criterion was that we could identify someone who would chair the study. I was told by Frank Press that it was perhaps the most successful board at the National Research Council. It's still going

strong. The man who succeeded me as chair of CSTB is William A.Wulf, who's now the president of the National Academy of Engineering.

The history of the name is as follows: Initially I called it Computer Science and Technology Board and then Frank Press told me he was unhappy with the telecommunications board and he said "I want you to take that over". So to preserve our stationery and preserve the name CSTB, we called it Computer Science and Telecommunications Board, so same letters except the "T" changed meaning.

There are several reasons for the success of CSTB. I was able to put together an extraordinary group of 20 people, about half from academic and half from industry. I hired a terrific executive director, Marjory Blumenthal. Marjory recently became an associate provost of Georgetown University and Charles Brownstein succeeded her as executive director.

I focused considerable efforts on raising funds; the boards of the NRC are required to raise their own monies. I notice these days that people list with pride on their CVs that they served on one of our committees or are on the Board.

HAIGH: Yeah. And is there perhaps one thing that springs to mind as an example of where there's been a clear impact from one of the reports?

TRAUB: No, I don't want to try to select one of them because we have had many influential reports.

CSTB has been very successful and it's one of the things that I'm particularly proud of. I built theories and I built institutions and it seems to me an unusual combination. Among the institutions is the Computer Science Department at Carnegie Mellon which had about seven or eight faculty when I showed up there in 1971 and it was really in danger of going down the tubes. Many years later Allen Newell told me that the faculty had given me one year to turn the Department around; otherwise they might leave. By the time I left in 1979 we had 50 teaching and research faculty. I was proud to start the Computer Science Department at Columbia which now has about 33 faculty and is very well regarded. Indeed one of our faculty is now Dean of Engineering. I was the founding editor-in-chief of the *Journal of Complexity*. We just celebrated our 20th anniversary.

HAIGH: So at the time of the previous interview, I think you had spent about 17 years in university computer science departments, all of them as department chair. Was it a hard transition for you to stop being chair and to become a faculty member?

TRAUB: I love it. It's often said that the best job at the university is being a professor. I like building things and I am good at it. But it was hard keeping up with my research at the same time that I was building departments. I appreciate the freedom you have as a professor; of course, you teach and you serve on committees and do professional service but most of the time you set your own agenda.

HAIGH: Yes. Another thing that you've mentioned is your attempts to produce a real number model as an alternative foundational concept alongside the Turing machine. Can you talk about that?

TRAUB: Let me give you the arguments for the two models. What's beautiful about the Turing machine model is that it's so simple and yet the Church-Turing thesis states that any function that is intuitively computable is computable by Turing machine. It's important to realize that the Turing machine is an abstraction since it has a potentially unbounded tape so it's not a physical model. The model of computation used in scientific and engineering computation and in information-based complexity, as well as algebraic complexity and geometric complexity is the real number model. It is also an abstraction with the following assumptions: you can store the representation of a real number and you can do arithmetic operations on real numbers at unit cost without error.

In an article published in *Physics Today* called "A Continuous Model of Computation", I argue that scientists already accept real numbers when they built mathematical models. So, for example, the Schroedinger equation is defined over complex numbers. The second argument is the practical one in that you could never do analysis of algorithms or complexity of scientific problems on a Turing machine.

So those of us interested in the complexity of continuous problems decomposed the process as follows: we use the real number model to develop algorithms and complexity results. If you create an algorithm that you are going to run on a digital computer, you should do a stability analysis and I claim that modulo stability, the running time predicted using the real number model, will be comparable to the running time on a digital computer.

Colleagues used to say to me that the reason you have to use a Turing machine is you can't create a theory of complexity classes over the real numbers. But Lenore Blum, Mike Shub and Steve Smale showed you could study complexity classes, NP-completeness, etc. using the real number model. I think that people who work on Turing machines want to study the power of computation whereas I want to solve problems occurring in physics, economics, etc.

The complexity hierarchy is important for people working on discrete or finite-dimensional problems, whether they use the Turing machine or the real number model. For continuous problems we can use adversary arguments at the information level to establish the complexity of many problems. For continuous problems the complexity hierarchy is much less important.

The original Blum Shub Smale model couldn't deal with functions because it didn't have the notion of oracles. They had to limit themselves to finite-dimensional problems such as solving systems of polynomial equations. Erich Novak added oracles to his model and this enables the study of continuous problems involving functions.

HAIGH: Yeah. And are you aware of impact that this work has had physicists and other kinds of disciplinary practitioners?

TRAUB: You mean information-based complexity?

HAIGH: Yes.

TRAUB: We have had impact on a bunch of disciplines. Our current work on quantum computing is of interest to physicists. We have had substantial impact on low discrepancy theory which is part of number theory. We published a paper in *Econometrica* , co-authored by John Rust. *Econometrica* is the most prestigious journal in economics. We have had considerable impact on computational finance where we showed that quasi-Monte Carlo methods are far superior to Monte Carlo for the calculation of financial derivatives.

HAIGH: On the policy level you were also involved with publicizing the idea of information security, information warfare, how did you come to be interested in that?

TRAUB: A couple of things happened in the mid 80's that made me aware of how vulnerable the country might be to physical or electronic attack. The first is that in 1985 I was granted access to the floor of the New York Stock Exchange. It looked to me that the only protection was a guy with a big belly packing a revolver. I was then taken across the river to Water Street, where all the processing for the Exchange is done, and I noticed that there was very little security for those computers. I thought that if I were a terrorist I might try to throw a few grenades on the floor of the New York Stock Exchange. It wouldn't do too much physically because this is not where the data is stored, but the symbolic impact would be huge. Furthermore, real damage to our data would be done with an attack on Water Street.

The second thing I noticed in the mid 80's was that various mutual fund families used just a few banks so I did some checking on back-ups and it seemed that in those days there was only one place that everyone was using.

My concern in both cases was that the country's wealth is stored in electrons with trillions of dollars being moved around electronically. There seemed to be very little concern about our information infrastructure. Furthermore, planes, trains and the electric grid are all controlled by computers.

In 1986 I became the Founding Chair of the CSTB and I tried to get the Board to do a study of national vulnerabilities, both physical and electronic. Even though the Board had a number of visionaries I couldn't persuade them to do such a study. (Of course this could have been a black study. That is, that we could not release it publicly.) I thought about doing an op-ed piece or an article to draw attention to this but I didn't want to give terrorists any ideas by going public with these vulnerabilities. I decided, perhaps wrongly, to simply keep my mouth shut. Then by the early 90's people started becoming aware of the threats. But even when I attended a meeting of the Presidential Commission on Information Infrastructure Protection at Annapolis, a naval officer said to the Commission "I don't think there is any threat here". That seemed very shortsighted to me; we are the most advanced country in the world in using information infrastructure

and, therefore, we are the most vulnerable. Now cyber security has becoming a huge area but for a long time nobody seemed to care.

[Start of Tape 3, Side B]

HAIGH: It sounds then in general that's an area where you feel that, unfortunately, your comments failed to have an impact.

TRAUB: I made the conscious decision not to go public because I didn't want to give terrorists any ideas. . In retrospect, I could have met with Al Gore who at the time was a Senator from Tennessee. I met Gore in 1986 when he talked to the Board about the information super highway. I could have said to Gore "this is a national issue what is your advise, what can we do?".

HAIGH: Well, that brings us to a natural last question then. You've already talked about the institutions that you feel proudest having built, and I was wondering if, other than issue you just mentioned, looking back on your career there's any area where you wish, with the benefit of hindsight, that had pursued some intellectual idea, or done something differently?

TRAUB: Good question. [long pause] I certainly had the fantasy, wouldn't it be nice to live in multiple universes where you can take both paths instead of picking one, but I feel very lucky with the life I have had. It was luck that I had a buddy at Columbia in 1955 who sent me to an IBM research facility just off campus which got me interested in computers and that lead to the two most important things in my life, my wife the author Pamela McCorduck ,whom I first met when I was a Visiting Professor at Stanford, and my career. I'm almost moved to tears but who could have expected such a wonderful life and such a wonderful career. Born in Germany in 1932, getting out at the last moment in 1939, coming to the United States, and then the opportunities this country gave me.

HAIGH: Well, thank you for taking part in the interview. And that is the conclusion.

TRAUB: You're a very good questioner, you really asked some very unexpected and provocative questions.

HAIGH: Thank you.

[End of Tape 3, Side B]