An interview with

JOHN R. RICE

Conducted by Thomas Haigh

On

23 and 24 March, 2004

West Lafayette, Indiana

Interview conducted by the Society for Industrial and Applied Mathematics, as part of grant (DE-FG02-01ER25547DE-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, John Rice, the W. Brooks Fortune Distinguished Professor of Computer Science at Purdue University, discusses his professional life and academic career. Rice studied at Oklahoma State University, before gaining a Ph.D. from Caltech in Mathematics. He interned at North American Aviation working on a computer guided missile project. After a post doctoral fellowship at the National Bureau of Standards, he worked for four years in the mathematics department at General Motors, part of its research division. In 1964 he joined the faculty at Purdue, at first serving jointly between the computer science and mathematics departments. Rice served for many years as department chair. The interview focuses particularly on his involvement in the field of mathematical software. Rice convened a series of Mathematical Software meetings during the 1970s, which are widely seen as crucial events in the emergence of a research community in this area. He was active in ACM SIGNUM, and was the driving force behind the creation of ACM Transactions on Mathematical Software, a central vehicle for the exchange of code and ideas in this field. He created the ELLPACK family of systems for elliptical problems. ELLPACK was used widely during the 1980s and allowed users to make high level definitions of their problems in a specialized language. It also served as an innovative framework for testing new mathematical algorithms and systematically evaluating their performance. This led him to promote the idea of “problem solving environments” for scientific software. Rice discusses other aspects of his career, including recent work on gas turbine simulation, and identifies some of his key research papers and books. He is the author of numerous edited volumes, of several books on the use of ELLPACK, and of two distinct series of introductory computing textbooks.

THOMAS HAIGH: This is an oral history interview conducted with Professor John R. Rice. It's conducted as part of the SIAM historical project on the history of scientific computing and numerical analysis. I'm Thomas Haigh. The interview is taking place on March 24, in the afternoon, and it's taking place in Professor Rice's office in the computer science department at Purdue University.

Good afternoon, Professor Rice.

JOHN R. RICE: Good afternoon.

HAIGH: So in this interview we'll be covering a wide range of topics including a general review of your career and intellectually wide interests and contributions. And then we'll also be looking in particular detail at your role in the production of standard packages for scientific computing in numerical analysis, particular the ELLPACK family of packages.

To start off with, could you talk in general terms about your early intellectual life, perhaps about your experience of school and how you first became interested in mathematics and science.

RICE: Well I was interested in mathematics as a high school student. My high school education was unusual in the sense that I lived in Ethiopia for three years during my high school years. So I went to a commercial school where I learned nothing for one year. I went two years to a French school which followed the regular French curriculum. Unfortunately I didn't speak French in the beginning, but in that school I reached the normal level of mathematics before I left. I also took tutoring in science, but not in mathematics. My father was a high school math teacher, so he knew enough about mathematics to teach me. I came back to the U.S. when I was 17, and entered college. I was majoring in engineering but I was very interested in mathematics and very interested in computing, even at that time. You couldn't take any courses in computing as such. As a sophomore in college I wrote a paper on electronic brains, which was how to build a decimal adder with flip flop devices. But then I left engineering. I went to mathematics because I could graduate sooner. Later, I wanted to get back into electrical engineering for a masters degree, but they refused to let me in because I didn't have the required courses in speech and such things, so I stayed in mathematics.

HAIGH: So returning a little, to cover some of those points in more detail. When did you first become aware that there was such a thing as the computer?

RICE: Actually I read an article in Time Magazine, must have been in 1948, or something like that, talking about using computers to solve scientific problems. I don't remember who the author was exactly, but it was describing Von Neumann's work, I believe, and talking about giant brains and that kind of thing. That's what seems to have triggered my interest in computing.

HAIGH: So at that point you would have been about 14?

RICE: Yes, something like that.

HAIGH: Now as a boy were you very interested in science?

RICE: I guess the answer's yes. Before I went to Ethiopia I lived in very small towns in Oklahoma, we're talking about towns with populations between 10 and 250, so there wasn't much of a science and engineering environment or curriculum, but I was interested in mathematics and science just as a general thing, but not loads, particular activities of that type.

HAIGH: Did you have any technology related hobbies like ham radio or chemistry sets?

RICE: Well my father would buy those kind of things for Christmas. He was not only a math teacher in high school but a science teacher, and sometimes the schools actually had a laboratory and I would get into the laboratory with him when school wasn't in session, and he would show me things. But I actually never took a course or any systematic program in a science or engineering topic.

HAIGH: And why did you choose Oklahoma State University?

RICE: Well I was from Oklahoma State University, basically. My grandparents lived there, my father went to school there, so when we left Ethiopia and came back to the States it was natural to go back to Oklahoma and that was just a straight forward thing for me to do, and that's what I did.

HAIGH: Can you talk a bit about the kinds of courses that you would have taken there as an undergrad?

RICE: I was taking the basic engineering curriculum, the French school is considerably more advanced than the American education, so I did skip a variety of elementary courses like the first year of physics, and I took accelerated chemistry. I hadn't had trigonometry so I had to take freshman trigonometry, and then took calculus, and then the usual history and English and such courses. And since I spoke French I was a real find for the French department, so I enrolled in French courses and they had very few students.

HAIGH: Would any of the courses that you did during your undergraduate education relate specifically to numerical analysis or specialized areas of applied mathematics?

RICE: I took a course in numerical analysis either as a senior or as a first year graduate student. I spent five years at Oklahoma State University, three years for a Bachelor's degree and two years for a Master's degree. The course I took in numerical methods I disliked intensely, which goes to show how things change. I also disliked statistics intensely when I took the first course, and I've since learned that's a very fascinating subject, but I really disliked numerical methods. My interest in doing something really got started when I started working as a summer student in the aerospace industry. I was in a group that was in the computing business, so to speak, trying to use computers to design airplanes, and guide missiles, and that kind of thing.

HAIGH: What company was that?

RICE: It was North American Aviation, which evolved through many generations, I'm not even sure what company it is now, it's probably –

HAIGH: I believe it's part of Boeing.

RICE: It could be part of Boeing, it became Rockwell International, and then somebody took it over then. It was a very large company at the time.

HAIGH: And they had a facility in Oklahoma?

RICE: No, it was in Los Angeles. So I went to Los Angeles for the summers, and the first summer I was there they were still using analog computers so I was involved with a group that was looking at analog computing, as well as some digital computers. But their underlying problem for the group I was in was actually to build a computer to guide a missile.

HAIGH: How did you come to be working for a company based in Los Angeles?

RICE: I went over to the interviewing office and talked to people and somebody was there from North American and offered me a job, I needed a job, so I went out there.

HAIGH: Was there a shortage of engineers at that point?

RICE: Yes, there was a big shortage of engineers in the U.S at that time, and scientists of all kinds.

HAIGH: Now during your time at the university for your bachelors and masters degree, did you come into contact with a computer?

RICE: Not at the university. I took one course the last year I was there, in digital logic. That was in the electrical engineering department, but there was no computer. But there were computers in North American Aviation. I worked for two summers at North American Aviation while I was based in Oklahoma, and then I went to graduate school at Caltech, which is actually in Los Angeles, so I continued working at North American Aviation at that time, they were happy with me and it was close by. Not only did I work in the summers, but the last two years I was at Caltech I was also a consultant during the winters for them.

HAIGH: So would the first of those summers have been around 1954?

RICE: I believe it's 1954, right.

HAIGH: Can you talk some more about the work that you did there?

RICE: Well actually the first summer even though it was supposed to get us into the groove of things, I spent working with desk calculators trying to calculate some numbers

to use to smooth in data, and when I came back the second summer, somebody pointed out to me that I'd made a mistake in about the third week and all the numbers after that were wrong so, they'd had somebody redoing them in the winter while I was gone. But in the meantime, computing was advancing even then, somebody had decided the thing to do is to write a computer program to compute those numbers, rather than having it done on desk calculators. It was a relatively trivial computer program, and that was the end of my desk calculator career, which I never regretted. And the second summer I was there I was really much more into doing something which you would call computing and working with real computers.

HAIGH: And were you writing programs for them?

RICE: The answer to that is "yes and no." The head of that particular department had this theory that writing computer programs was a menial task and that you could hire high school graduates, who'd majored in anything but were relatively sharp, and they would write the computer programs for the engineers. I was in the classification of being an engineer, and not a high school graduate. So people who were writing programs actually had a "programmer" working for them who actually wrote the programs. On the other hand, if you wanted to get anything done you actually had to understand the programs yourself and work with this person to make any progress.

HAIGH: So were you supposed to hand them a flow chart and say, go away and program on this?

RICE: That kind of thing. But I worked with the same person for the last three years I was there, I believe. She was a very intelligent and well educated person, not like the high school graduate type that the department had thought we should have, and we had a very good working relationship. I would describe to her what I needed to be done and she could understand equations and flow charts and that kind of thing, and she would find a assembly language programs to accomplish that. When they didn't work we would go over the programs trying to find out where they were failing, and it was very tedious programming in that environment because we were in central Los Angeles, and the computer was in west Los Angeles. The turnaround time for a run was three days, so you spent an inordinate amount of time checking and double checking your programs before you sent them away.

HAIGH: Did you ever actually see the computer?

RICE: I never saw that computer. It was an IBM 701. The department I was in had a smaller Burroughs paper tape driven computer (actually it was a Datatron at that time I believe). I didn't actually program that computer, but it was there. And actually North American Aviation decided to go into the computer business itself, and built a prototype of what we would now call personal computer, it was about the size of a desk. I wasn't involved with it, but the neighboring groups were, so I saw that activity going on. Again, we were trying to build a guidance computer, and I never saw that computer, but we had a simulator of some type of it so you try to write programs in this very constrained computer. It had a very small memory, and a very small instruction set.

HAIGH: So the computer would have been part of the missile itself?

RICE: Right. It was the guidance computer that would go in. North American Aviation at that time was working on a ram jet based missile not a ballistic missile, but so that was in the missile that would guide the ramjet into a target.

HAIGH: And the computer you were designing would be digital or analog?

RICE: It was a digital computer; my recollection is it had something like 650 words of memory.

HAIGH: By the way, can you remember the name of the programmer you mentioned?

RICE: The missile programmer?

HAIGH: Yes, you mentioned a woman that you worked with.

RICE: Oh, what's her name? Gee I'm getting old, I certainly know her name, Melba Nead, because I've kept up with her but I think, though, she's died because the last two years I haven't gotten Christmas cards from her. So I did keep up with her, her husband was a banker, and she was well to do, so to speak. She had worked with a Nobel Prize winning chemist, or something, at Caltech before she came to North American Aviation, so she was very much into the science environment, very sharp person. And she left North American Aviation about five years after I did and she went to the Jet Propulsion Laboratory and worked in their software groups for fifteen or twenty-five years, a long time. I knew several people at JPL and they knew her, and she was in that environment.

HAIGH: So is it your recollection that it was common to find female programmers in the 1950's in this technical area?

RICE: Well certainly in the department where I was in, not only should you be a high school graduate but our boss thought that you should be a woman, too. [Laughter] So yes, that was common, although they did have some programmers who were men of this type. By the time I left, I was there, at North American Aviation maybe five years, by the time I left they'd figured out that that system doesn't work, and they were moving on to having engineers write their own programs, or something like that.

HAIGH: That's a very interesting bit of computer history. Now as you began to work with the computer, designing routines and collaborating on programming, did you have a feeling that this was something that you enjoyed, and that you would want to continue with in your career?

RICE: I didn't visualize myself as getting into the software business. I did visualize myself getting into the scientific business, including applications. My Ph.D. thesis at Caltech actually was derived from projects at North American Aviation. If you're simulating anything that's flying you have to have some experimental data that's called the coefficients of drag, coefficients of lift of aerodynamic bodies. Those things are just tabular data, and since memory is extremely expensive you would try to reduce the

amount of data by curve fitting. Curve fitting in those days was pretty primitive and ineffective so I got interested in the curve fitting problem, which mathematically is approximation theory, and began studying approximation theory on my own to try to understand how to do this better.

HAIGH: So in this case you were originally drawn to the mathematical theory as a way of solving problems that you were working on personally?

RICE: Right. Although my Ph.D. thesis doesn't look like it's related to that problem, but that's how I got to that point.

HAIGH: That's interesting. Is that because to get a Ph.D. you had to abstract away, and hide the relevance?

RICE: I don't know that it was that way. It just seemed to me it naturally led to it. I read this guy's paper and those guys' papers, and other papers, and I finally came up with this problem that is related to curve fitting, nonlinear curve fitting. Because the standard thing was something linear, like polynomial least squares with ten degrees, twenty degrees, or thirty degrees, which does not work in this application. So I began looking at nonlinear curve fitting schemes, and a small literature already existed in nonlinear approximation at that time and I had some ideas about other kinds of nonlinear approximation and what you could do, and that led to my Ph.D. thesis.

HAIGH: You mentioned that the motivation for this had been the limited memory capability of computers that were then available. So would you say that in this period the kinds of mathematics that people were becoming interested in were being influenced by the availability and capabilities of electronic computers?

RICE: I personally saw that there were a lot of people who were trying to find ways to solve differential equations, to get curves, which were more efficient because computers were slow and they didn't have much memory. You had to have a better method of doing things than what people did with tables. Your table could be twenty pages long and you would dig through it and find the right number. Once you tried to put it inside the computer you were looking for just faster and more compact ways to do things.

HAIGH: That's an interesting point then because compared to a desk calculator computers were much quicker (though obviously compared to what you've got now the computers were very slow) but as I understand what you're saying: they might have been quicker at calculating but compared to a human they effectively had much less random access storage to use.

RICE: Remember, you're not just trying to put the data inside the computer. In a missile application you had no choice of doing it desk calculating because there wasn't going to be any person there. Even when the 701 was running, you could have done it with a desk calculator. There was another attribute of desk calculating, which I alluded to earlier, remember in the third week I made a mistake and all my calculations were wrong. The group I was in at North American Aviation had literally a room full of people using desk calculators whose job it was to do a hand calculation to check the computer

programs. We're talking about twenty, thirty people sitting in there. Of course the computer programs were very often wrong, but that didn't last more than a couple of years because they discovered that what they were really doing was using the computers to check the hand calculations, because the errors in the hand calculations were much more common than the errors in the computer programs. So that group of people disappeared out of the operation.

HAIGH: And were the errors coming from programming mistakes or hardware glitches?

RICE: Well, they primarily were concerned about programming mistakes, but the hardware certainly was not as reliable as it is now. I'll give you a concrete example. I once was trying to do a calculation to prepare some data which involved a hundred equations of a hundred unknowns, which would have taken probably three hours of machine time or something like that. But it turned that what I really found out was how long the magnetic tapes go without making a failure. I never finished the calculation because there would be a tape error and the mean time between failures on magnetic tapes was around an hour to an hour and half, and so I never actually got an answer out of this calculation, so that was a serious source of error. I don't recall having hardware errors ever in, you know, that the adder got the wrong number. I think those kind of things did happen but I don't recall it happening to me. But tape errors were the big problem.

HAIGH: Okay. So switching back slightly chronologically; let's move forward to talk a little bit about the work you were doing at North American while you were in grad school at Caltech. Now how did you come to be in Caltech?

RICE: Well, I decided to leave Oklahoma State. I didn't want to stay there for a Master's degree but I was in ROTC, and I had to finish that. So even though I graduated from college in three years, I had to spend four years in ROTC so I spent two years getting a Masters degree. I very systematically surveyed all the departments of mathematics in the U.S., ranked them, and decided which ones that I would apply for, and so forth. Then one day the department head called me into his office, because he knew I was looking, and he said, "John, you're going to Caltech". [Laughter] He had had a very good friend that, who was the head at Caltech, and he mentioned that there was a very big shortage of scientific people of all types, including graduate students, and so he gave me this sales pitch about how Caltech was the best possible thing that could ever happen to me. It was almost at the bottom of my list for scientific analysis things, but I went to Caltech anyway.

HAIGH: So why was it at the bottom of your list?

RICE: Well, for example, the salary was incredibly low at Caltech compared to other places. Let's see, my stipend the first year was $1166, most departments were offering $1400 or $1500 a year stipends for graduate students. On the other hand Caltech did have the advantage that I had this connection with North American Aviation, it would be easy

for me to work in the summer, so which, you know, is not what would have happened if I'd gone someplace else. So anyway I decided to go there.

HAIGH: And was there anything that the math department at Caltech that was particularly well known for at that point?

RICE: I would say they had three world-class faculty members. At Caltech you're in a very tough competition, the math department never had the stature of the physics or chemistry departments. So it was never that good, but they had some really good people, pretty old, and they had a few young people. The middle group, of people who had been there more than five years and less than twenty-five years, was almost nonexistent. There may been three or four young people, it was a very small department.

HAIGH: So what was it like when you got there?

RICE: Well it was a huge difference from Oklahoma State University. All the graduate students were sharp people, and the undergraduates were incredibly sharp. It was a real shock, when I taught calculus, to realize that at least half of the students there were smarter than I was. I mean, at Oklahoma State where I taught calculus for two years, I never saw anybody I thought was as smart as I was [Laughter]. The courses were very challenging at Caltech, very small courses, but a lot of very good students and the undergraduates would take graduate courses. There were times when the top half of the course was undergraduates, and the bottom half was graduate students. It was also intimidating because the department had some really strange idea of how the world works. When I arrived there were seven new graduate students, there was one graduate student in the second year and one graduate student in the fourth or fifth year, and he was finished with courses and finishing his Ph. D. So you ask, where are all the graduate students? Well for the previous two years all students had failed the qualifying exams, and they only had one try. The first semester I was there, the one guy who was a second year student failed the qualifying exams and he left, so he didn't even finish off the year. So that left seven first year students and this one guy who was finishing up his Ph.D. thesis, which made very serious students out of the graduate students.

HAIGH: That must have been quite intimidating.

RICE: It was, and there were some obnoxious faculty members involved. What really annoyed me, and annoys me now, they weren't all that good that they should be lording it over the graduate students the way that they were. They really failed some people who were very good. I would say roughly half of the people who failed the qualifying exams there went on to other universities and got Ph.D.'s in mathematics from universities as good as Caltech, and some of those have been very successful. They had completely unrealistic expectations of the graduate students. And I think the year after I left there was, finally, not a revolution of the graduate students, but the department realized there must be something wrong and they changed their whole policy. Cleve Moler was a graduate student there, but by the time he got there they'd gotten rid of this idea that we only give Ph.D.'s to Von Neumann and up, [Laughter] and were beginning to pass people wholesale.

HAIGH: Were there any courses you took that particularly stand out, either in terms of your personal reaction of the material that you covered?

RICE: Well, I didn't take any courses related to my personal interests in computation. At one time I was going to write a work in partial differential equations which is also related to modeling airplanes, and so forth, and I took a course by someone who was visiting there, Finn let's see he had just come from USC, and I think he was at Caltech three or four years, and then he went to Stanford. Anyway, I decided that partial differential equations was not for me, and I had gotten more and more interested in this approximation theory business, and so I read a book on approximation theory as a reading course. Spitzer, who is a statistician, agreed to monitor the course. Then the next year I was looking for an advisor and Spitzer was actually leaving Caltech, I think he went to Cornell or someplace like that, and Arthur Erdelyi, who is a very senior applied mathematician (special functions, that kind of thing) agreed to take me on as a student and help me with getting my thesis right in approximation theory. He has a very interesting footnote: he was a professor at Edinburgh, when, who was a famous applied mathematician, wasn't Bateman, no I can't remember [Alec Aitken? TDH], anyway some super professor was there and forced Erdelyi out of the department and he came to Caltech, and he was at Caltech probably twelve, fifteen years. Anyway, when that super professor died Erdelyi went back to Edinburgh to take his position, his chair [Laughter], which just did wonders for him.

HAIGH: So it was a period of exile. And how was your relationship with him?

RICE: Well I would say it was very good, but very formal. He was not an unfriendly person, but he knew who the students were and who the professors were. He worked very hard at trying to improve my writing and my mathematics. I made the mistake, I realized later, when I finished my thesis in maybe December, when graduation wasn't until June, so we spent four months improving my English and my composition, exposition. I would write a draft of my thesis then he would give it back to me a week later all marked up; that went over and over for weeks that way. So he put a lot of effort into my thesis and improving it, he had incredible high standards about every aspect of the thesis; you know, in the wording, fonts, you name it, everything had to be right.

HAIGH: And you said that that was a mistake?

RICE: Well if I hadn't waited three months to finish, I mean, -

HAIGH: The work was expanding to fill the available time…

RICE: [Laughter] Right.

HAIGH: And you said that your thesis topic had come directly from your work in the aviation industry.

RICE: Right.

HAIGH: Were any of the faculty interested in those kinds of problems at that point?

RICE: No, Erdelyi was the closest. It's a kind of applied analysis area, and he knew that area pretty well, so he was certainly capable of helping, but he wasn't doing anything related to it.

HAIGH: Were any of the people in the mathematics department using computers in any way at that point?

RICE: Not when I got there. In my second or third year, John Todd came from the Bureau of Standards, but by that time I was through the beginning courses and already starting on research. Caltech had a very demanding pace for graduate students and in the second year you were supposed to start your research project and you didn't take any courses past the second year. There wasn't this five or six year time frame. They would not keep students longer than four years, and then you left with a Ph.D. or without one, but you left after four years. And like this student, I mentioned, when I first came there who was in, had been there for four years, he was actually still working but not supported by the department, he was trying to finish up this thesis. He was trying to prove the Riemann Hypothesis or something incredibly difficult, and didn't want to settle for something less than what he was shooting for, so he was staying around on his own while he worked on it.

HAIGH: Can you talk a bit about the content of the thesis and how it evolved into a suitability academic rigorous form from the problems that you were working with?

RICE: Well the subject matter is best understood in the terms of approximating functions by rational functions. If you think of this curve that you want to approximate, if you try to approximate it with the ratio of two polynomials this becomes a nonlinear problem because the coefficients in the denominator don't enter into the problem, linearly. That turns out to be a much more powerful way of fitting curves with a modest number of parameters, so I had gotten into rational function approximation. Now there was this large classic literature related to analytic functions, about the approximating of rational functions but it was all taking the approach of Taylor series expansions, as opposed to least squares fitting. There was a guy at UCLA named Motzkin, who had developed a theory of unisolvent functions which were nonlinear functions of N variables that had the property that if you had N points you could always interpolate at those N points. So it's a property that polynomials have, but they didn't have to be linear. So unisolvent functions were nonlinear functions that did that. That attracted me because I knew that I was dealing with nonlinear curve fitting. The drawback of unisolvent functions is that there are no interesting examples that are nonlinear. Lots of them exist but there're not anything that people want to use in practice. But it made a very elegant theory.

If you try to interpolate with rational functions, almost all the times you can do it. But there will be certain combinations of points and values that you cannot interpolate with a rational function, so there's singularities in the interpolation problem. That was what my thesis dealt with. I parlayed Motzkin's idea and I called them varisolvent functions because it's clear that with a ratio of, say, two quadratic polynomials you can always interpolate at three points because you can set the denominator equal to one and then you

just have a quadratic polynomial on the top, and you can interpolate at three points. So there are a variable number of points that you can interpolate with rational functions. The question is to understand what happens to these: where are those three points, how do you know which case you're in, and that kind of thing. That was basically what my thesis dealt with, an analysis of the properties of approximating by rational functions, and this variations of number of points and how these singular situations could be identified, and how you could do something if you were in one of them.

HAIGH: And was the computer itself useful to you in doing this research and identifying these points, or was it all done with a pencil and paper?

RICE: No, it was all pencil and paper analysis at that time. But I had done a lot of hand calculations just trying rational functions and I had discovered, experimentally, that rational functions did a good job on these curves I was trying to fit.

HAIGH: And looking at your resume I see that from 1959 to 1961, you published ten single authored papers. Are these all related to your thesis work, or are you at this point developing new interests.

RICE: The journal papers, the first two there related to my thesis, the third one was actually a paper that I wrote that came directly out of my work at North American Aviation, because people solving differential equations were using Runge-Kutta methods and I came up with doing a way of Runge-Kutta methods which was called Split Methods.

[End of Tape 1, Side A] [Start of Tape 1, Side B]

And then as we go on, "Characterization of Best Approximations" [J.R. Rice, The Characterization of the Best Nonlinear Tchebycheff Approximations, *Trans. Amer. Math. Soc.*, 96, (1960), 322-340] was from my thesis. Then there's a paper on approximations by functions of the form ab^x+c. It turns out that my theory of varisolvent approximation applies not only to the rational functions but to exponential approximations where you have numbers to some exponent of x. So that was working on that kind of thing. Then there are some algorithmic papers along some of these lines. They were all, came out of my thesis topic but they were not in my thesis, that was ideas, and I just kept going in different kinds of approximation.

HAIGH: Are there any of those early papers you remember getting particularly strong reactions to?

RICE: The approximation theory community was pretty small, but nonlinear approximation was an idea that really appealed to people in that area because linear approximation had been, maybe the right word is worked to death, because functional analysis is based on linear approximation. There's really an enormous literature associated with linear approximation, so nonlinear was viewed as a quite interesting new way of doing things. So I think most of this work was well received.

Down here is a note on numerical solution of gas lubricated bearings, that also came out of my North American Aviation work because the guidance systems in these missiles had gas lubricated bearings and there are some equations associated with them I worked on for about a year trying to solve while I was working at North American Aviation. Those things finally came out in a paper in the *Transactions of the American Society of Mechanical Engineering*, and so on. But I stayed in nonlinear approximation for quite a while, if you look through my vita.

HAIGH: Okay. So then after you graduated in 1959, you spent a year as a postdoc in the National Bureau of Standards.

RICE: Right.

HAIGH: Was that connected with Todd's presence in the department?

RICE: It could be. I didn't have any direct contact with the Bureau of Standards, it was just an open competition and I applied. On the other hand, in retrospect, it's easy for me to believe that since Todd, who was already in contact with Caltech, this information might have been there in some way. But, again, there was an incredible shortage of mathematicians, so I think if you had kind of respectable credentials your chances of winning any kind of postdoc position were pretty good. So I think then to sort of differentiate, I had two offers from Big Ten universities as an assistant professor where I've never spoken to anyone at the department, I mean they just came in the mail, [Laughter] we're offering you an assistant professorship, blah, blah, blah, that was how bad the job market was in those days or how good it was.

HAIGH: And that was in 1959 when you finished up your Ph.D.?

RICE: Right.

HAIGH: Were you tempted by any of those offers?

RICE: I liked the idea of having a postdoc; I really didn't seriously consider those offers. I liked the idea of having a year being in Washington, doing what I wanted to do. I actually accepted employment at North American Aviation when I graduated in June. But I had already decided if I got the postdoc I would take it, then that came in August or September, I got word of that, so I left then and went to Washington.

HAIGH: What attracted you so much to research rather than to direct applications in industry?

RICE: Well I can't say what it was, I just liked doing research. I enjoyed working at North American Aviation and would not have been terribly disappointed if that would have been the only job I'd ever gotten. Because I knew the people there, I knew what they were doing, and it was a very dynamic time, a lot was happening in the aerospace industry in those days. I would not have been unhappy to work there, although it would be very different from being a researcher or professor. But I like the idea of being a professor too, so a postdoc put the decision off for a year.

HAIGH: So during the postdoc did you spend a lot of time talking to other people at the Bureau of Standards or were you actually just left to your own devices to work?

RICE: Well there was a small group there and they had a section, I guess, it was numerical analysis and there were probably five full time people or something, a small number. There was one other postdoc. There were lots of discussions going on and interchange of ideas, but I didn't actually work with anyone on any projects while I was there. So I just kept continuing all those things I'd been doing for my thesis, and things that came out of it, and I had a great secretary there who was very good at writing papers and typing things for me.

HAIGH: Is there any particular thing you remember learning there, any relationships you formed there that were important in later years?

RICE: Well Phil Davis was the head of this section there, and he was interested in approximation theory so I had a number of discussions with him. I didn't know it at the time, but he was already starting to write a book on approximation theory, so there was certainly somebody there to talk about approximation theory. John Todd wasn't there anymore. They had good computing facilities so I was able to continue doing experimental things in approximation, trying things out, that kind of thing.

HAIGH: So would you say then that at this point that your research was becoming more experimental?

RICE: I don't think it was becoming more experimental because I've always been interested in experimental things. If you're going to write a Ph.D. thesis in mathematics you have to be theoretical for a while. I don't think that my interests, you know, moved towards experimental more than it was. I was always interested in how to get approximation to work.

HAIGH: So you always enjoyed working experimentally, but to get a Ph.D. thesis you had to prove some theorems?

RICE: Right. And I, you know, I was proficient at proving theorems as well as doing experiments, so I enjoyed that kind of thing. But it was sort of unusual for me not to go into academia after I left the Bureau. I got this offer from General Motors which I couldn't quite believe, but decided I had to take. I got a much larger salary than professors got and as I was told that I could do anything I wanted to, so it was like a postdoc, and it actually worked out that way. I spent four years at General Motors, my own interpretation of what happened was that they had this man named Henry Garabedian, who is the uncle of the well known Paul Garabedian, at Brown University. He'd been a professor at Chicago before the war, and he got involved in the nuclear engineering program during World War II; had worked in Bettis atomic power projects and the nuclear submarine projects, and so forth. General Motors decided they were going into the nuclear engineering business, and they hired a group of people in nuclear engineering and nuclear physics. They had them in Warren, Michigan, for maybe four years, and then maybe they decided they weren't going into that business. They all left

but Garabedian, and he decided he was too old to keep moving, so he just stayed there, and my own assessment was that they'd created a mathematics department at General Motors so that Garabedian would have an organization to justify his salary, because he didn't actually have anything to do.

HAIGH: Had that remained in Michigan?

RICE: Right, in Warren, Michigan. And so it was a small department and they had maybe four, five Ph.D. level students, and two or three other technical people, and then a secretary. But anyway, he said that if I would do research that would be fine, publish papers, so I said "okay I'll try it." It turned out that for four years I was not pressured to work on anything for General Motors. There were people in his group who did apply their work on General Motors' problems, but not everyone.

HAIGH: So that was optional?

RICE: In some sense, yes.

HAIGH: Was this part of the larger General Motors research division?

RICE: Well General Motors has a research division and it, at that time, was divided up into departments. I don't think there were smaller structures than departments, most departments were bigger, but this was a very small one. And then they had things like a computer operation which was a service department. But they were supposed to be looking at things like new engines, and one of the things that I observed there was that if you wanted to get ahead in General Motors research, the way to do it was to latch onto some problem that was of value to General Motors, make contacts outside the research division with people who needed that problem solved, and do something about it. But I wasn't interested in getting ahead in General Motors, so I didn't really do any of that, and I wasn't pressured to do any of that.

HAIGH: Did you interact with any other researchers?

RICE: I interacted very much with Carl deBoor, I don't know if you know his name. He was a graduate student at Harvard and he left Harvard after one year and came to General Motors to work. He didn't have a Ph.D. then. and he left General Motors about the same time I did and he went back to graduate school at Michigan, got a Ph.D., and became a member of both the National Academy of Science and National Academy of Engineering. He and I were very close friends. Obviously he was very bright guy, and he just retired from Wisconsin this past year.

HAIGH: I see you've had a number of publications together over the years.

RICE: Right. So we talked about mathematical things at great length all the time. Another person who was a student of Birkhoff's, Birkhoff was a consultant for General Motors, so deBoor had started out as a student of Birkhoff, and Birkhoff got him a job at General Motors. And then Robert Lynch, who I've also written papers with, finished his Ph.D. at Harvard and came to General Motors Research Lab, and stayed there, I think, for

three years. And he had a situation similar to mine, he was not oriented towards working on General Motors problems, But deBoor, not having a Ph.D., actually was required to work for a living, and he worked on approximating automobile body surfaces,. Of course that was very closely related to my interest, even though I didn't work on that project, but he was working with engineers who were trying to actually design automobiles on computers, instead of molding clay which was the current technology. He and I had a lot in common mathematically, and we became very good friends, and remained good friends all these years. And Lynch was a professor in this department for thirty-three years or so, he left General Motors and went to the University of Texas, and then came here a few years later. Then he was interested more in solving partial differential equations than in approximation, but he did do some work in approximation.

HAIGH: What were the computing facilities at the research division like?

RICE: They were world class for the time; they always had the latest and biggest computers. When they were designing automobiles on the computers they had very elaborate special devices built, which took up a room this size, where you could make displays of automobiles twice the size of that blackboard back there. Huge screens and they put millions of dollars into that, it never worked [Laughter]; they didn't skimp on spending money there.

HAIGH: You'd mentioned that previously you'd had a three day turnaround on your jobs. So by this point how long would it take to get something run and receive the results?

RICE: Oh, you were talking about, if you were lucky, maybe an hour. It would depend a little bit on the load and how things went, but you might get a run back in an hour, on average. So you could get several runs in a day. Still things were rather primitive in the sense that you're using cards. We had a system on which compiling FORTRAN programs was time consuming, so people learned how to make binary corrections. Everyone that was in computing in the department would actually be able to take a FORTRAN program, read the assembly language code, and figure out how to make changes in that. Then they would punch up binary cards themselves that they would put at the end of the deck to patch the code. There was a queue if you had substantial key punching. If you only had a few cards you could do it yourself, but if you wanted a program punched you put it in a queue and somebody key punched it and it came back, so you tried to avoid that delay.

HAIGH: So was this the first time you were using a high level language to program?

RICE: No, the Bureau of Standards was just getting FORTRAN when I came there. So I learned FORTRAN when I got there. By far the biggest change that's ever happened in programming was when FORTRAN replaced assembly language programming; it just changed the whole ratio of effort required to get something to work. In spite of all the shortcomings of FORTRAN, it was so much better than assembly language.

HAIGH: And did the GM computer center provide any libraries?

RICE: Well the libraries in those days were provided by manufacturers, so SHARE was already in operation to try to supplement IBM. But IBM had sines and cosines and linear equation solvers, some basic kinds of things. I don't remember a whole lot about the libraries as a whole, but I did use various things. During that period of time I became interested in the kind of things that SHARE was doing, and at that time I also became involved in the approximations needed for elementary functions. So that was input into libraries, like the sines and cosines, and the exponential and gamma functions, and so forth, which is an approximation theory problem. I became involved with people who were interested in that, including H. Kuki who worked for IBM or Argonne; he was really a sharp guy. And IBM had some general interest in this, but it wasn't one of their high priority items to provide good libraries. Of course, that was what led to the creation of SHARE, because in the beginning everyone had this idea: "well somebody's written this program why isn't somebody else using it, let's try sharing it." SHARE never really worked. It's still not completely clear why libraries don't work better than they do. In those days the main problem was that if you wrote a program you would give it a fancy name and send it off to SHARE and they'd put it in their library. So you could tell your boss your program was now in the SHARE library, and there was no quality control whatsoever. You may have a good program that only solved one problem, the one you were working on, and was useless to anyone else. But you'd give it a general name, or maybe it didn't work at all. The quality control issue was the big problem with the SHARE codes.

HAIGH: So did you have personal experience trying to use any of the functions from the SHARE library?

RICE: When we were doing this, this group that was involved in approximating functions was generally looking at other things too. People would try out what the cosine did, how fast it would run, how accurate it was, at all of those functions. So I certainly got into the business of looking at how the library routines that worked, that IBM provided. Many of them had some very serious flaws, flaws that stayed in those codes through different languages and different generations of machines, and so forth. For example, there was an IBM routine. I don't know where they got it; anyway it showed up to solve polynomial equations. There was a polynomial let's say of degree 5, that was like $x(x-2)(x-2)(x-3)(x-4)$ so that it has five real zeroes, a very easy straightforward thing. This routine would find four of the zeroes and the fifth zero would be a pure imaginary. Completely bogus, it wasn't in the round off error or anything, it was just completely bogus. That algorithm showed up in the PL/1 libraries, it showed up in the IBM libraries well into the 80s, because you could test it by just putting that polynomial in and seeing what it did and would always come up with the wrong answers. Those are the kind of things that motivate people to look for quality in libraries.

HAIGH: Do you know if at that point IBM was officially supporting the libraries or did it just provide them to use as-is?

RICE: Well I don't think they made any guarantees, but they certainly had a group of people who were supposed to be making the libraries work. Like Kuki, who was based in Chicago, but was an IBM employee. I think he left IBM and went to Argonne National

Labs sometime later, but anyway he was in the library business for IBM and there were people in Yorktown Heights, and so forth, who were also in computational approximation. But it seems that part of the problem is that there were the researchers in IBM that did things, and then there were the guys who made the systems that were shipped out to the customers, and they didn't talk very often.

HAIGH: Maybe their researchers were more like the General Motors researchers they were just too busy doing their own thing. [Laughter].

RICE: Yes.

HAIGH: Looking at your list of all early publications, your first one, and a number of subsequent ones, was with the journal of the Society for Industrial and Applied Mathematics, SIAM. How was SIAM in those days, was it a large organization?

RICE: Well I don't think it was; certainly not large like the American Math Society, but it was operating in a respectable level in the sense that it had journals and it had meetings. I went to Math Society meetings as well as the SIAM meetings, and the Math Society meetings had tons of people, almost none of whom you knew just because there were so many people. SIAM meetings were much smaller operations and you're much more likely to know people there because the community was much smaller, but still they had people who were in areas pretty far removed from mine, like optimization, and there were the symbolic algebra people were active in SIAM. You've mentioned SIGNUM on your list, and when SIGNUM started it was a little bit like HP, three guys in a garage. Joe Traub was the guy who I believe started it and when he decided to have it, I think maybe eight, fifteen people showed up one time for it, and said, "Gee that would be a nice idea why don't we try that," and that was within ACM, not SIAM.

HAIGH: Yes, he showed me the first issue of the SIAM newsletter and it listed the attendees. So we should talk about SIGNUM in a little bit more detail later. Before the foundation of SIGNUM had you been involved at all with ACM?

RICE: Certainly I got involved with ACM much later than with SIAM. I didn't even go to an ACM meeting until maybe 1966. By that time it was showing signs of being large; I went to a meeting in San Francisco and they had a huge exhibit area and such things as that. I was not attracted to ACM in the beginning; I think I joined at the first meeting. I was there with a good friend Saul Rosen, so I joined, but I'm a joiner and I didn't think of ACM as being a home for me.

HAIGH: And is that just because this kind of applied mathematics research wasn't at this point well represented within ACM?

RICE: Ah, that was probably it, yes, I think so, to a certain extent it was that. SIAM really reflected my personal interest and most of the people I talked to and so forth were in SIAM. Of course some of them were overseas, but SIAM was the natural place for my interest.

HAIGH: And did you take any kinds of formal roles within SIAM?

RICE: No, I didn't. I don't think I ever had any nontrivial office in SIAM, and actually the creation of the ACM Transactions on Mathematical Software put me on a bad guys list for lots of people in SIAM –

HAIGH: Oh –

RICE: Because it wasn't a SIAM journal. And it was because SIAM has never been very graceful about admitting that computer science is a real discipline, and they're still having a hard time seeing that things like programming and systems have any intellectual content. I talked to SIAM at length on the Journal of Mathematical Software and it was that "software" word that was the problem they had. There were plenty of people in the power structure of SIAM who did not want software but they wanted the journal, and so they discussed other titles and that kind of thing, but I came to the conclusion that SIAM was not the right home for that journal and so I went with ACM, even though that was very difficult because ACM was in an anti academic mode –

HAIGH: What year was this?

RICE: Well the Journal started in 1975, which was the first issue. But the work started in maybe 1971. Let's see, the first the idea of the Journal came up in a meeting in Colorado. It was probably the NSF Workshop on Software and Algorithm Evaluation, University of Colorado, which was in Boulder, although the meeting was actually in a little town west of Boulder. The NSF Workshop in Granby, Colorado, the same year, that's where the idea of a journal was discussed. So there was three years of discussions and negotiations, and the challenges were much different in SIAM than ACM. SIAM had a well oiled machinery of publishing technical journals and ACM didn't. ACM basically had the *Journal of the ACM, Communications of the ACM,* and the newsletters of the special interest groups, but there were a couple of presidents of ACM who were absolutely and totally opposed to having academics being influential in ACM, and that meant not having any scholarly research oriented journals. Also, they were in a big financial crisis, their accountants couldn't add, so the people who were against the journals decided that they could accuse the journals of being the rat hole that all the money was going down, and since nobody actually knew where the money was going [Laughter] we couldn't argue against it very successfully.

HAIGH: Would you be thinking of Walter Carlson and Herb Grosch among them?

RICE: Yes, that's right. Particularly Grosch was against academic types, and in fact, you know, exactly the people he was against were the people running SIAM who were, in turn, against his type.

HAIGH: So that would have favored SIAM?

RICE: Right. But I concluded that the Journal itself would fare better under ACM, I don't know that I was nominated at the meeting in Colorado to do this, but I was on the ACM's Publications Board already so I knew a lot about ACM publications and I knew that the financial picture of publications being presented was grossly incorrect. I didn't know what the true picture was because nobody did, but I knew that journals, in fact, are

things that make money for societies and so I was engaged with the idea of counteracting Herb Grosch and this anti academic group.

HAIGH: So did the Journal come to some extent out of your involvement with SIGNUM as well within the ACM? Was SIGNUM related to the Journal in the early stages?

RICE: It wasn't in any official organizational capacity but certainly a lot of the people involved were there. But, again, one of the things that happened over this three year period was that there were new presidents of ACM so that it became feasible to try an academically oriented journal.

HAIGH: Let me see, Jean Sammett, would she have taken over at that point?

RICE: Probably, about that time, I don't remember who the presidents were.[1] I didn't deal with them directly to a certain extent, but certainly I knew Jean fairly well and she was not nearly the anti professorial type that Herb Grosch was.

HAIGH: And Dan McCracken, although maybe he was later.

RICE: He was in that area, he was not particularly favorable to the journals, but you couldn't be more against them than Grosch was.

HAIGH: Right. So would these conflicts be played out in the ACM council and would that be the body from which you would need approval?

RICE: They certainly had to approve it and one of the things that I spent a lot of time, was making up mock issues, financial projections, and such things. The NSF very much wanted to have this journal come into existence so there was a mathematical software conference which, whose proceedings would be the first issues of the Journal, and that way the NSF would actually subsidize the first issues of the Journal as proceedings of this conference, so the risks the first year were essentially negligible. It paid for the first year of the Journal.

HAIGH: Okay, so let's look back now. You had this almost entrepreneurial work within the organizational to get the journal accepted, and you've said where the original idea came from. On the intellectual level what was it that you needed this journal to do that wasn't adequately being done in existing venues?

RICE: Well somebody had had the idea that algorithms were important so *Communications of the ACM* had this algorithms department. Of course, in the beginning, you don't know how the world is going to work, but it two weaknesses. One is that many of the algorithms that got published there didn't have any impact. I mean, if you write an

[1] Editor's note: Anthony Ralston served from 1972 to 1974 and Jean Sammett from 1974 to 1976. Both had academic orientations. Grosch's term as president came from 1976 to 1978, though he was influential earlier in the decade as vice president and as a council member. His term as president was the most divisive in the association's history.

Algol program that computes N factorial you really haven't accomplished very much, and those kinds of algorithms were getting published there. The other thing that was happening was that it was focused on Algol and that was not what most practitioners were actually computing in. The fact that the Communications had these Algol algorithms actually became one of the main arguments for getting the Transactions on Math Software going, because they would take those things out of the Communications. A lot of people didn't want to read algorithms, which is sort of understandable, and so it saved money on the Communications. The algorithms editor had become Lloyd Fosdick, who was not an Algol fanatic, but was a mathematical software fanatic and so he actually wanted to publish useful algorithms in whatever language was appropriate. He wasn't against Algol, per se, but he recognized that algorithms in other languages were important, and maybe more important. He liked the idea of having machine readable versions of the algorithms. He was a proponent of that. Once the algorithms get longer you want to try to get rid of the key punch errors and that kind of thing, so he and I were natural allies in this because he wanted to get into a venue where you could publish big programs and yet have some sort of quality control that you think of in refereeing papers.

HAIGH: That's interesting. *Communications* published algorithms but they were algorithms obviously written in a specific real language, even it was one that not many people used. *Transactions* is about software. At that point how did you understand the distinction between an algorithm and a piece of software?

RICE: Well, the difference between algorithms and software is not an easy distinction to make and it's a philosophical distinction. In my view software is an algorithm, my own personal mathematical definition, not operational definition, of an algorithm is as something that specifies exactly what's going to be done. I think that's what mathematicians really believe is an algorithm, but they don't like to think that a FORTRAN program is an algorithm, because somehow they don't want to have the expression of what's going to be done be FORTRAN. But FORTRAN is just another language for saying what you're going to do…. Well it wasn't well defined in the beginning, that's true, but then that's true about all languages. English is not a very good language for describing algorithms either because you don't know exactly what you're saying all the time.

HAIGH: Right. So how you saw it, if you had to put them in some language then you might as well make it FORTRAN.

RICE: Or something that's mechanical, something that you can actually compute with some machine runs. If you want them to be useful it should be a machine, not a Turing machine, which can run but doesn't run very well. I think, I didn't have any preference for FORTRAN, per se, but the opportunities in those days were pretty small for what languages you could use. Algol was very well defined but these Algol systems didn't work very well, they weren't very widely available and FORTRAN was what everybody was using, but, you know, there are other languages like PL/1 that were popular for a little while, Pascal, and such things.

[End of Tape 1, Side B] [Start of Tape 2, Side A]

HAIGH: Okay, so I think you've said then at this point that *Communications* was becoming more open to publication of algorithms in FORTRAN. And you've talked about the distinction between an algorithm and piece of software. So can you from that to talk about the need for the *Journal*. It would seem that if Communications is becoming more open that might actually reduce the need for separate journals.

RICE: Well I think that mathematical software somehow has a niche between theory and practice where algorithms can thrive. Most of the people who do mathematical software and numerical software, scientific software, have a pretty good idea of mathematics even if they're a mechanical engineer, or that kind of thing. They have an idea of what their theoretical calculations are all about, yet they also want to actually get answers. It's not so much true with people who are in systems, for example, it's harder for them to deal with something that's more abstract, of trying to prove the quality of a system even though they talk about that kind of thing. You're in a framework where you can talk more easily about whether you're getting the right answers or not out of a calculation of cosine x or for solving linear system equations, than you can about something like managing the load on a computer system, where you're not so sure of what it is you're trying to achieve anyway. So it is in an area where it's easier to do something that's both academic and useful than many other areas of computing. You're probably aware that the computing community has its own division between the theoreticians and the practitioners, and a lot of professors don't want to see the theory guys because they're so busy making things work that they don't want any analysis, and then the analytical guys say "you're not doing any analysis, you don't know what you're doing." Our department has these kinds of fights all the time, and somehow mathematical software is an easier place to have both things, because the people see both sides by their backgrounds and nature.

HAIGH: Right. So having the *Transactions* as a separate venue makes it easier to make those kinds of contacts with application areas?

RICE: Right.

HAIGH: And would some of the readers of *Transactions* be, say, practicing physicists or engineers who need to work with software and would go there and read new methods and that kind of thing?

RICE: Yes, I think that certainly those kind of people do come to the *Transactions*, and sometimes even submit papers to it, because they've seen this way of doing things and they can take their own application and abstract out of it a kernel problem that looks like it would go in a math software journal.

HAIGH: Would you ever get the feedback from people who, perhaps with your own work or others who'd published in the Journal, from people who tried to put the methods into practice and would have some feedback or some comments?

RICE: Well there's a formal mechanism for doing that in the Journal and there have been some very expressive feedback appearing. Never mind saying, "This line is wrong,"

but saying, “This convergent criteria is not very good and you should try that convergent criteria and it works a lot better,” that kind of thing. So some of the comments that have come in have been fairly thorough analysis of what’s good and bad about the algorithms.

HAIGH: So that relationship between research and application has been quite fruitful?

RICE: It’s been quite fruitful, but it’s still not as fruitful as I would have hoped. I mentioned this experience with this little algorithm for finding polynomial roots from IBM that went on and on. You ask yourself, how could that happen, really there are a lot of people who just accept math software, so that when it says “this is the answer” that’s it. There’s not as much attention to quality as you would hope even in places that you would think would be the bastion of quality. Take MATLAB, for example, the polynomial root finder in MATLAB, if you take a slightly nonstandard problem, is a random number generator. Actually Cleve Moler knows how to fix it, because I’ve written him a letter to him pointing this out. But, if you take five points and interpolate that by a polynomial of degree five and the five points you’re interpolating are 1, 3, 6, 9 and 8, you get the right answer to sixteen digits. If the five points you’re taking are at 10001,10003, 10005, 10008, and 10011, mathematically exactly the same problem, you get random numbers out, just because in the algorithm they don’t shift the coordinate systems of polynomials and MATLAB is too busy doing other things. That was just like IBM, they were too busy doing other things to fix that polynomial root finder. Who knows what’s wrong with it but it’s got a flaw, so it’s very hard to get errors out of software.

HAIGH: And in general with people writing their own custom programs, did you find that there are some disciplines or areas where this kind of feedback between practitioners and researchers was more successful?

RICE: I can’t say that I find any. People who have a foot in engineering fields are much more amenable to discussing feedback and knowing that there’s a difference between proving a theorem and making something work, and that you often have to twiddle with programs, and that you have to have some heuristics that you can’t quite formalize to tell you whether or not you’ve got the right answers. You get comfortable with the answers, you know, but you can’t prove that that was the right thing to do because you can’t solve all the problems exactly, you can’t figure out what’s going on.

HAIGH: Okay. So backtracking slightly to the formation of the *ACM Transactions on Mathematical Software*, Were there any other people who were particularly involved in promoting and shaping the journal?

RICE: Well of course Lloyd Fosdick was involved. I certainly talked to Stuart Lynn; he was chair of the publication boards at that time, he was in favor of the idea. He was not especially a math software type but he just thought that was a good idea for ACM. But most of the math software community thought that was a good idea. After all, many of them were academics, they were looking for academic respectability, which means, “I’ve got a place I can publish papers and this will help me.” It was polarized, the people who were too academic didn’t like the idea because it was applied, and the people who

were against academics in totality didn't want any journals, so there was those two polarized groups, and we were (almost as the title say) in the middle, "mathematical software," so we got attacked from both sides in some sense.

HAIGH: So you'd mentioned that the papers, to begin with, came from the conferences.

RICE: Right.

HAIGH: Now as those were published was there a stream of new material coming in for the early issues?

RICE: Having enough material was not a big problem, there were times when the backlog was slim, we were usually saved by the fact that the ACM publication's office was also somewhat inept, so we could easily be two issues behind, so [Laughter] that allowed us to manage, we never had a crisis of not having enough material, but had we been on time we might have.

HAIGH: And were there any kinds of issues that proved controversial, about what standards should be used to judge submissions, or about what was and what wasn't an area that the Journal should include?

RICE: There wasn't any, I think, focused area of that type. There were some people who were unwilling to admit that their submissions had any faults and that kind of thing, but any sort of consistent theme…. Probably the most long standing issue was the language issue about what languages should be involved, because there are some things that Language A just doesn't do well compared to Language B, and we had gone from being one hundred percent Algol to one hundred percent FORTRAN, so what are you doing with those things that FORTRAN doesn't do well? In principle we were always willing to accept alternatives but it wasn't easy to say what the alternatives are because you really didn't want to have assembly language for a particular machine, and until C came along you couldn't write programs in any decent programming language that dealt with the hardware. So there were people who would like to have published programs in the IBM assembly language, and then there was a little bit of the argument well assembly language is more efficient than FORTRAN, but we dealt with that through the Basic Linear Algebra Subroutines, the BLAS, where people had a set of FORTRAN library routines that had machine specific implementations, so they're very highly efficient. But that was all very friendly and no big controversy. There certainly were people who wanted their programs published that weren't. I can remember getting a submission from somebody who wanted to publish in Algol, so I ran his program through the local Algol compiler and there were more lines of error messages than there were of code, so I told him that he might try to get rid of the machine specific things out of his Algol, and he wrote back and said, "what's wrong with your Algol compiler?" [Laughter] So that was the end of that submission, and there were things of that type.

HAIGH: And you'd also implied that there was some kind of machine readable distribution going on.

RICE: Right. Lloyd Fosdick's strong intention in the whole business was that you would get cards or magnetic tapes and there was a big distribution service set up and it was rather successful. So you could just measure how many people were getting the algorithms that were published, and of course it varied widely between algorithms, but there are a lot of algorithms where many copies were ordered. People only had to pay the handling costs, and it was a pretty nominal fee to get a machine readable copy. Then we had microfiche versions, too, which people got for free. They weren't all that useful to actually run the program but some people could use them at least to see what the program did.

HAIGH: So for a program that would be too long to publish in print you could also receive a microfiche -

RICE: In fact, microfiche was put in to the back of the journal issues, and then you could get a magnetic tape or a punched card version, and then when the programs began to get long punched cards became cumbersome to ship around the world. That kind of thing has now been replaced by making it available on the Internet, and so you can get them that way.

HAIGH: In many areas of computing there would be a big difference between a piece of code that is a nice little implementation of an algorithm that shows that it works and it's reasonably efficient, and a production piece of software that would work nicely with real data and that real users would use on a day to day basis and be completely happy with. I suspect less so in scientific than in systems or business, but would it be true with the kinds of algorithms that were being published here that, for example, usability would be a factor?

RICE: Even in the first decade, you would see a range from programs that were primarily demonstrations that this kind of thing would work, to rather complicated programs that somebody had written to solve a certain realistic class of problems. The programs and algorithms have gotten longer and longer as people become more ambitious. I haven't actually studied it, but I would guess that the percentage of showcase type algorithms has gone down. A certain number of directions have played out. A good example that is adaptive quadrature, where the first algorithms were really quite small and really amazingly effective for being one or two page programs, compared to the sort of nonadaptive methods that people would have to use before. But then as you make the adaptive quadrature programs more robust and able to handle more difficulties they get bigger and bigger. In a way somehow it's gotten to the point that these algorithms work for everything except people who are trying to define a problem they can't solve, so that the problems they don't solve really are not of great interest, and so there's not as much activity there. A little bit like Gauss elimination, if you have a dense matrix it's hard to improve upon Gauss elimination too much.

HAIGH: And did the journal use standard peer reviewing methods?

RICE: Right.

HAIGH: Would there be some challenges in applying those to software? For example, if you had a much nicer implementation that didn't really do anything fundamental to the underlying algorithm, would that count as sufficiently novel to be published?

RICE: Well I don't recall an example of that kind of thing. Certainly reviewing algorithms was a challenge because you were asking a lot compared to many papers. I mean, sometimes people have to work hard to get them approved. Often you can read a paper in an hour or two and discover that this guy doesn't know what he's talking about, and that's the end of it. But, with an algorithms consisting of two or three thousand lines of code, you're never going to get anything done in just an hour. So it's harder for the algorithms editors to get good reviews and to get people to exercise things. Since the programs are more complicated, if you had a contrary referee out there he can always find something to complain about. Also, sometimes a referee would just insist that all kinds of little details be taken care of that were not fundamental in other people's minds.

HAIGH: I should think it would be around this time that people first started talking about structured programming and programming style and, you know, paying attention to a program as something that you would read. Was that something that was becoming a factor in reviewing or in setting standards?

RICE: Certainly readability has always been a desirable trait, and people would, you know, referees would say you know an algorithm should be well documented, the variable names should not just be X1, X2, X3, X4, and things of that type. There are some authors who have a completely distorted view of how people read programs and don't want to make those kinds of changes. Just like there are authors who can prove theorems but they can't write, and there are people who can write programs that work but they can't make them understandable. But I don't think that this is much different; it's just in a different context, so to speak.

HAIGH: Right. So you think that a reasonable person would know good, readable code when they saw it?

RICE: Yes. And of course the algorithms editor has a responsibility to actually make spot checks of the code right off the bat, for example, to see if it would compile. Like say it doesn't compile; you don't send it to the referee, you just send it back to the author. You would read through the code and if there are no comments in the code you're probably going to send it right back. Structured programming is a flavor of making the program understandable.

HAIGH: So, for example, the famous debates about GO TO, those didn't prove too significant here?

RICE: I don't think they were that significant. Dijkstra notwithstanding, there are times when exactly what you want to say is GO TO someplace, and you can actually create problems where it's very hard to make readable or efficient code without using GO TO as compared to what you can do with GO TO. FORTRAN was a language that required

more GO TOs than were natural, but to say that GO TOs are unnatural is pushing it too far.

HAIGH: So then this community of GO TO using FORTRAN programmers would be quite distinct from the more Algol inclined community of more theoretical computer scientists?

RICE: It could be. I think most of the people involved in the journal were not trying to force everybody out of FORTRAN. Algol is fine, FORTRAN is fine, and they both have their strengths and weaknesses, and, again, C's fine. If you're trying to make a device driver or some graphics on the screen in FORTRAN, you have terrible problems. I mean, it's possible, a Turing machine can do it, but it's sure not going to be easy, and there are just some languages that treat some problems better than others.

HAIGH: And how did the journal change over time?

RICE: Well the biggest change was that the small demo algorithms disappeared primarily, I think, mostly because all of those of interest were covered in some way. Programs got longer and longer, I don't know what the record is for programs, but, you know, there's now some really huge algorithms in there, things that you would never want to have the deck of cards on your desk for them. They had twenty, thirty thousand lines of code, fifty thousand, they very ambitious kinds of programs. I think that's the main thing, it's always been the algorithms' editors intention for publications to be utilitarian, you know, this program has to be useful to a reasonable set of people, and there are just bigger and bigger programs. That's made more and more problems with the refereeing, because it's harder to. Of course, one thing you can do is ask the authors to present evidence that their program works, since after all he's supposed to test it. Ask them: "What tests have you done and let's see the results of those." even though they were not going to be published.

HAIGH: Does ACM gain the copyright to published code?

RICE: Yes and no. They do have the copyright but they have a blanket policy that the code can be used with only reference to ACM provided it's not in some sort of publication item. So you cannot write, you know, *The Purdue Book of ACM Algorithms* and sell that king of thing.

HAIGH: But you're free to use them or redistribute them?

RICE: Right. So it's just to, it's really designed to prevent publishers from republishing the ACM algorithms and making a profit, you know, of course a lawyer wrote it so it's a couple of paragraphs long, but that's basically what it says. It's just to protect the print rights.

HAIGH: Do you see that this related at all to the more recent open source software movement?

RICE: Well I think it is in the sense that you might interpret what that policy is; if it's published in the *ACM Transactions,* it's open source. But you can't make a book. Open source, probably doesn't have that constraint on it either. You could make a book about open source Linux or something, but ACM only is looking to prevent the publication competitors.

HAIGH: And at least with some open source licenses, if you then took something it and improved it you would be obliged to release the improvements back into the community.

RICE: That's not involved here because, well I think, it would be harder for ACM to manage that kind of thing anyway. I would say half of the comments that appear in the Transactions are actually the kind of open source where somebody says you can improve this algorithm if you replace this code by that code. Those kinds of comments on algorithms are welcomed by the journal.

HAIGH: You were Editor in Chief from 1975 all the way up until 1993. During that period do you think there's anything that you personally did, other than the things you've discussed, to put your stamp on the Journal, things that perhaps another editor might have approached differently?

RICE: I can't think of too many things, In fact I'm trying to think of anything that I said, you know, we're going to go this way and not that way. I've always been a person that's utilitarian oriented so I've never been one to say well, you know, those kind of applications are not to be considered. If it fits with any kind of plausible idea of mathematical software we would consider it. I don't know how far away it would get before we would not do that. Well, you know, if somebody submitted an algorithm for a Turing machine I probably would not support publishing that, or something like that that was really not ever going to be computational, something that never actually executed in the real world. But I don't visualize myself as having said, "Let's try to do that" or "Let's try to do this" about something that was unusual and didn't seem to be a natural flow of things.

HAIGH: Okay. So let's jump back slightly in time now and talk about SIGNUM. Which you have said was really the beginning of your substantial involvement with the ACM. So how did you first hear about it? Were you there at the very early meetings?

RICE: I don't think I was there at the very first one, but I was involved fairly soon. I don't have a clear recollection, but I would say that during the first year or two I had become aware of it; SIGNUM would meet at places where I would be, so I was happy to participate in those meetings.

HAIGH: And other than Joe Traub do you recall any other individuals being particularly active in the early years?

RICE: Yes, Tom Hull was a very regular participant in SIGNUM. He was at Toronto, and worked on ordinary differential equations and he was also active in math software in general. Lloyd Fosdick was also a regular participant.

HAIGH: And what kinds of things would go on at SIGNUM meetings?

RICE: They would organize mostly special topics meetings. They wouldn't have a general meeting of SIGNUM in the sense that the math society would have a meeting and every kind of mathematician would show up. They would organize a meeting on software for solving partial differential equations, or software for special functions, pretty focused kinds of meetings. And they could have fairly successful meetings, you know, fifty, a hundred people would show up.

HAIGH: And would those be freestanding meetings or part of general ACM meetings?

RICE: Mostly free standing, but at ACM meetings they might have an evening session, or something like that, where the people would get together without a big program or presentations and such. And SIGNUM for a while sponsored an award and invited talks at the ACM meetings. You can see the plaque there; I was one of the speakers once, the George E. Forsythe Memorial Lecturer, that's at one of the ACM meetings. They did that for ten, fifteen years, but SIGNUM really fell into inactivity over the years and faded away, but it was really very good for the numerical mathematics community to have these kinds of things. You were on the program at the ACM national meeting in the same way as the Turing award lecturer was, and so on.

HAIGH: I understand that SIGNUM grew quite rapidly in the late 1960s. Now it also seems from what you've said that at, prior to that you, and probably other people with the interests, were more at home in SIAM.

RICE: Well I think there was a little bit of this awareness that SIAM was not as inviting to people who were really up to their eyebrows in computation. It was more learned people who would prove the convergence of some method for solving partial differential equations, not people who would make some method work well. Mathematical software was attractive to a lot of people but not to SIAM. There were probably three; this is just a guess, meetings that were joint ACM SIGNUM-SIAM meetings, to try to create a formal collaborative program between the two groups. So these would occur at some larger meeting, and it was one of the things that everybody that was at those meetings was for it, but it never really happened.

HAIGH: Do you have an idea of why it didn't happen?

RICE: Well if I had to say I think the larger organizations themselves were not particularly interested in it, and those two groups didn't want to try to form their own independent united organization. That's my guess as to what happened. But there was a lot of overlap in membership between those two groups and a lot of common interest.

HAIGH: Can you give an example for this time period, say late 60s early 70s, of the kind of paper that perhaps you yourself, or someone with similar interests, might want to give that SIAM at their meeting would have thought "this isn't really what we do," and perhaps ACM would have been more inviting to. What kinds of topics would fall into that category?

RICE: Well let's just look, we're talking about when 1975, 1980 period?

HAIGH: Maybe slightly earlier, although later is valuable too. But one of the things I was kind of interested in was how SIGNUM gathered so much steam. Joe Traub, looking at the newsletter, said that the membership pretty quickly increased between one and two thousand people. At this point membership was free so it didn't cost anything to be in SIGNUM, so the number of those who were really interested was much smaller. It still seems though that it's an area that attracted a lot of interest quite quickly, presumably because somehow SIGNUM let people do things or talk to people with similar interests, in a way that before then was not as well served either by SIAM or by the larger ACM.

RICE: Well the kind of paper, this in 1978, that SIAM would have trouble with would be "Evaluation of Numerical Methods of Elliptic Partial Differential Equation" [E.N. Houstis, R.E. Lynch, and J.R. Rice. Evaluation of numerical methods for elliptic partial differential equations. *J. Comp Physics*, 27, (1978), 323-350]. It was a paper that tried to take a serious but experimental view of how best to solve elliptic partial differential equations. There's Galerkin, finite difference, and various methods, so which one actually worked the best? You have some sort of view of what's best, not formulated in an absolutely precise mathematical sense, but in terms of accuracy and efficiency and so forth. Then you do some experiments and you report on them. That's the kind of things that wouldn't get very far in SIAM. So that is a very interesting question.

HAIGH: So it comes back to experiments versus theorem proving?

RICE: Yes, right. I think so.

HAIGH: So from 1970 to 1973 you were on the SIGNUM Board of Directors, and then 1977 to 1979 you served as Chair.

RICE: Right.

HAIGH: Is there anything you particularly remember from those experiences?

RICE: I don't have any complaints about those kinds of things. I think the people involved were energetic. We organized various conferences and workshops that were successful. I was quite pleased with the way things were going in SIGNUM in those days. The newsletter was reasonable. Newsletters always have their problems, they're not quite respectable academic publications so there's not as much motivation to publish in newsletters, but we were getting enough that they were interesting to read and not too thick. I thought things were going very well. I didn't ever think of SIGNUM as becoming a society like SIAM is, never mind ACM and I thought it was doing well in those days. And maybe five, eight years later I think it went down in membership and the level of activity went down first, and it was possibly that the people who became involved with SIGNUM simply didn't have the ability, or the desire, or both, to organize meetings that attracted people to SIGNUM on special topics, like mathematical software libraries. There were quite a few meetings that were quite successful I thought in the sense that fifty, one hundred people were present, with lot of interesting interactions. They stopped having those kinds of things, and the newsletter got thinner and less frequent.

HAIGH: Do you think then that bad management would be the main type of error? Or was there anything changing in terms of competition from other organizations, or increased specialization in research interests, or anything like that?

RICE: I'd hate to accuse them of bad management, but I certainly think that that was probably part of it. I think that the failure to have an ongoing set of meetings was not due to the fact that there weren't any topics available, but people just didn't want to spend the energy to organize them, that's my guess. That was my impression at the time.

HAIGH: And during the time that you were most actively involved with SIGNUM during the 1970s, how would you characterize the relationship with ACM headquarters and the organization as a whole?

RICE: I'd say they were pretty cordial. SIGNUM was not one of the bigger special interests group, but it was a viable size. It's difficult to say what you really think when you're being recorded but I'll say it anyway: by and large the staff and management, as opposed to the membership, at ACM headquarters ranged from barely competent to much worse. There are a few people there who know what they're doing, but I mean it was the exception. For example, the membership software for ACM got so bad that the ACM subcontracted the collection of dues to IEEE because they couldn't do it. Some people would get twenty-seven notices for their membership dues and other people wouldn't get a notice for five years and have to voluntarily send in the money. ACM publications had the same kind of people working for them, so there were a lot of problems with ACM in general. So that meant SIGNUM didn't get great support from ACM headquarters, but no one else was getting great support either.

[End of Tape 2, Side A] [Start of Tape 2, Side B]

HAIGH: So moving on then from SIGNUM to what it seems was the other large institutional and profession building activity you were involved with in these years: the series of mathematical software conferences.

RICE: Right.

HAIGH: Can you talk a little bit about the motivation behind these and how the first of them came to take place?

RICE: Well I'm trying to remember now just why we decided to have it, where the idea of having the first mathematical conference came from. I can't remember the deciding moment, maybe I should look in the book to see that.

HAIGH: I can read to you from your acknowledgements. You thank the organizing committee, Robert Ashenhurst, Charles Lawson, Stewart Lynn and Joseph Traub, for their advice and aid, and you thank ACM SIGNUM for sponsoring, and you thank the mathematics branch of the Office of Naval Research for funding.

RICE: Right. I recall that once the idea was put forth there were people who thought it was a good idea. I'm trying to think where the first support came from; it was probably

the Office of Naval Research. Now I can't remember the name of the head of the math computer science at the Office of Naval Research at that time. I'd have to dig up to see what his name is, but it might have been in a discussion with him because he was trying to promote academics, he was in the math branch, supporting applied type activities in academia. I saw him every once in a while, and we might well have been in some conversation where I said, "how about a conference on math software?" and he said, "okay I'll provide you basic funding." The conference was here at Purdue. It was not an expensive conference, you know, it didn't pay travel expenses for twenty-five people and such things as that, so it was a fairly small investment.

HAIGH: Had you been working with the Office of Naval Research previously?

RICE: Yes I had. When I was at General Motors I had a research grant from ONR, I think it was for $15,000, and the vice president of General Motors Research Lab was unhappy about it because he said it cost GM more than $15,000 to process all of this kind of thing because they didn't have Government research grants. But my department head liked the idea and he came back and said it's something that GM should do; it gives you visibility, and so on. So that didn't bother me so much, but I remember that it was unusual because the Government normally does not make research grants to people working at places such as GM. So, anyway, I had some contact at the ONR in 1962 or 1963, before I got that grant. So I'd had contact off and on with the ONR, sometimes as an advisor on proposals and those kinds of things. I did see them, not on a regular basis, in the sense that it was timed, but it was frequent enough.

HAIGH: Do you know if they would have been supporting any of the other pieces of work that were discussed at the conference? I guess I can check that by looking at the other acknowledgements.

RICE: It's possible, but I'd have to check. The papers in the proceedings list support from the AEC (6), NSF(4), NASA(1) and ONR(1).

HAIGH: In the preface you include the comment that there's this challenge "which is sociological in nature, in that mathematical software is not yet a viable scientific community or entity."

RICE: Right. I think that was certainly true at that time.

HAIGH: And did you see the conference itself as the beginning of the process that could –

RICE: Well I think so, of course I'm biased. I sensed a substantial amount of enthusiasm for the topic, and there were probably fifty or sixty people present. The company IMSL was just starting up in Texas to do mathematical software, the Numerical Algorithms Group in England was just getting off the ground, and things at that time were going on. Lloyd Fosdick was already interested in the idea of making ACM algorithms a more viable and useful kind of thing, and it just seemed to be a number of people who had to come to a similar conclusion, and that turned out to be mathematical software.

HAIGH: So is there anything that you particularly recall about the meeting?

RICE: Well not just that there were the particular papers, but I thought that there was a lot of common interests surfaced there. It brought together people who didn't usually speak to each other.

HAIGH: There are 35 distinct contributors listed in the proceedings. Do you know if there were more papers presented than made their way into the book?

RICE: Probably not. It wasn't a long conference. I think it was one of these things everybody who wanted to talk was given a chance to.

HAIGH: Actually there's another quote from your own paper in this volume. So on page 38, you talk a little about the ACM and the work that SIGNUM's doing and then you say, "These activities do not form a solid support for mathematical software and they should probably be described as sporadic. The primary deficiency, e.g. SIGNUM's committee on subroutines certification, has been the lack of a person or persons with both technical and organizational ability who will stake his professional reputation on his ability to organize and accomplish something significant in the field of mathematical software. Until such people appear, one can expect the professional society efforts to continue to be sporadic."

RICE: I think that's what was happening at those. There were very few people who visualized themselves as being in mathematical software. They would be interested in approximation theory so they would work a little while on programs to compute sine and cosine, and be interested in something else, and they'd work a little bit on this, but there were very few people who saw mathematical software as their domain of professional activity.

HAIGH: And you think that during the 70s, following this first meeting, that was beginning to change?

RICE: Right, I think that meeting really did bring people together who didn't know each other very well, many of them, who had some sort of interest in the field. The 1972 meeting in Colorado, I think, involved many of the same people. It was not a direct follow-on, but there was certainly a large overlap in connection, and then that meeting led to the formation of the *Transactions on Mathematical Software*.

HAIGH: So the second meeting was held in Colorado?

RICE: No, that was not a math software meeting. It was the NSF Workshop on Mathematical Software, in Granby, Colorado, 1972.

HAIGH: Okay. That's the workshop the proceedings of which turned out to be the first volume of *Transactions*?

RICE: No.

HAIGH: No? I saw these books, there's Mathematical Software 1, and there's Mathematical Software 3….

RICE: Okay. There was a meeting, there was this fictitious meeting, and it actually took place, where the proceedings were the first issues of the Transactions on Mathematical Software. That was Mathematical Software 2.

HAIGH: And that's why it isn't a book?

RICE: It's not a book.

HAIGH: Were you the main organizer of the first conference?

RICE: The one here that's called *Mathematical Software*? Yes, right, I was the main organizer of that. And I was the main organizer of the conference that doesn't have a proceedings, but I was not the main organizer of the one in Granby, Colorado. The main organizers there were the people at Argonne.

HAIGH: Did you already know all the people that you invited?

RICE: Not well at least, there certainly were people at the first mathematical software meeting that I really didn't know personally.

HAIGH: And would it have been the first time that all of those people had met each other?

RICE: I think so. I think it was of that type that a lot of these people didn't know each other. I'm not sure who knew what though, but, you know, there were people there who were not connected together beforehand.

HAIGH: And are you aware of any lasting relationships or collaborations, your own or other people's, which came out of these meetings?

RICE: Well I think, for example, that even though I knew about what was going on in Argonne in mathematical software, I didn't know the people as well as I did afterwards. I certainly didn't know the people who were starting up this company, IMSL. Basically I met them at that conference, they were not well known in the academic community, and I think there were a number of people like that. And there were people who came like Lothar Collatz , a Professor of Applied Mathematics at Hamburg , who wrote a very famous book on functional analysis, so a somewhat theoretical person on the surface but underneath he was actually very interested in things like mathematical software. He came to this meeting, and I don't think he presented a paper but he might have, but I remember that he and I had a very fruitful conversation one night after the meeting was over that led to unrelated things happening later. If I'd made a list of people who were interested in mathematical software then I would not have mentioned his name at that time, but with the publicity, he decided to come and see what was going on.

HAIGH: Do you enjoy editing, or putting together books of this kind?

RICE: Well, I don't think I enjoy it but I've done enough of it that there must be something about it that doesn't repel me anyway. But I don't look forward to that kind of thing and say, "Oh boy, here's another one".

HAIGH: As per the quotes that I just read to you, it seems that over the years you've published a string of these pieces addressing issues such as "what is mathematical software and what should we be doing with it" and "what is numerical analysis and what does it have to do with computer science." Those things are more thoughtful articles, which are great for the historian because they really try to step back and sum things up a bit. Do you enjoy writing those kinds of articles, and do you get strong reactions from them ever?

RICE: Oh, sometimes, but I like that kind of thing. I like history and such, even though I'm not an expert in history. I certainly like to see how the world works, and in my own domain I think about why people do the things they do, and how things develop, and so forth. So I'm certainly interested in the sociology of the academic world, or the lack thereof.

HAIGH: Have you ever written anything that turned out to be controversial in one of those?

RICE: Controversial. I can't think of something that has created a flap that I'm aware of. I'm sure that there are some people who don't like what I say, because I'm either too applied or too academic. I have written opinion type pieces off and on, but I don't think I've jumped in and started a fray or jumped into the middle of one particularly.

HAIGH: All right. So returning to the conferences there was a series of these three mathematical software meetings. Were there any further meetings after that?

RICE: Actually, the IFIP working group had a meeting on that topic.

HAIGH: Is this IFIP Working Group 2.5? –

RICE: Right.

HAIGH: 1977.

RICE: There was a SIGNUM conference on Mathematical Software in 1977 (Albuquerque) and an IFIP WG 2.5 conference "Performance Evaluation of Numerical Software" in 1978 (Austria). that was in the spirit but without the name.

HAIGH: That's right. That's under "professional" on your resume, and then you, oh then also from 1975 to 1978, Chairman, COSERS Panel on Numerical Computing. I've actually had other people talk about their report, but I haven't read it yet I'm not entirely sure what it was.

RICE: COSERS was some sort of government operation, or program, to look at a variety of computational areas and have an evaluation of the state of the art, and of what

the future research projects were. I think Bruce Arden was the editor of one of those books, I don't think I have the actual book here, it's probably at home. I'm the principle author of about a hundred page article in it. It was supposed to summarize the past, present and future of various areas of computational science, that's how I would describe it in a fan mail.

HAIGH: Are you aware of any impact that it had?

RICE: I don't know if it had a lot of impact. I think the people involved were fairly satisfied with the activity I was involved with. The other groups did the same thing, they had a small group, who did things and then they would have a meeting at, for example, the ACM National Meeting, and they would have an open forum to discuss present ideas about what was important and unimportant and ask the audience for on the spot or later feedback. There were, I think, maybe ten subject areas that the panel was supposed to address, of which five actually finished, that's my recollection. The book has about five or seven hundred pages from those five or six areas, so, which was a definite subset of those that they were intending to cover because it was a lot of work to do. I remember one meeting of that panel in San Francisco, there were probably two hundred, two hundred fifty people in the audience, listening to the presentations and discussions of some big operations.

HAIGH: So the impact would have been more indirect then?

RICE: Yes.

HAIGH: Alright. So, to return to where you were, there were these mathematical software symposiums in 1970, 1974, 1977. You were saying there was also some IFIP activity in your area.

RICE: Right. They had the 1978 conference mentioned above and a conference which had the topic but not the phrase "mathematical software" in its title. It was later in Sophia-Antipolis, France, in 1985 with Problem Solving Environments in the title.

HAIGH: You remember what happened at the meeting?

RICE: Well the IFIP group meets every year, and has maybe fifteen to thirty people who are all members of the group present. Every third year, roughly, they have a working conference, and there are all kinds of topics, but numerical software is the title of the groups, so it's someplace in that area. I was chair on one on numerical software for solving partial differential equations, and they've had them on linear algebra, they've had them on problem solving environments. And they once had one that used the phrase "mathematical software" in the title but if you're a working group on numerical software a lot of your meetings are going to be mathematical software. But in my mind it's just one in a stream of numerical software meetings so it doesn't stand out to me as being special like the ones that I organized in the 70s.

HAIGH: After the 1977 meeting, were there further meetings that you didn't personally organize?

RICE: Not with that title. Again, many of these IFIP working group meetings can be construed as being on mathematical software. A complete record of their meetings is given on their web site: http://web.telia.com/~u13107484/wg25/index.html

HAIGH: So this group of people from software companies, user areas, and researchers, came together for the Mathematical Software Symposium and the other meetings. Once they began to form this community, what happened next? Did it discover other meetings and other places to gather?

RICE: The IFIP working group was formed in maybe 1972 or 1973. Five people, or so, decided to put it together and they got IFIP to approve it and they had a series of working conferences. Starting in about1977, and then every three or four years after that, they had another one.

HAIGH: Would that include people from software companies or was it more academic?

RICE: It was more academic but there were definitely software companies. But being an IFIP group, the membership included so many people from this country, that country, and another country. Then when you have working conferences all kinds of people can attend, but it definitely has to be a spread across the globe kind of activity.

HAIGH: So did the community keep its coherence then? After 1977, were there still the same opportunities for people to meet and exchange ideas?

RICE: A lot of the people that are still active in the IFIP group are people who were in these math software communities, like Lloyd Fosdick. He's retired now and stopped coming to those meetings, but that was only in the last three or four years, and he was in it all through his working career. And Brian Ford from NAG (Numerical Algorithms Group) and Oxford University in England is another example of a person who was all the way through all of these things.

HAIGH: So then the IFIP group really became the main vehicle for the community?

RICE: In a way, right. Its shortcomings as a main vehicle are that it's got this international constraint on membership and that it doesn't have a journal, or similar thing.

HAIGH: Are there any other groups or journals that would play an important role in the 80s and 90s?

RICE: Well there are a lot of software oriented journals now, engineering software, all kinds of software journals. Which I think is a reasonable thing because they want to focus on software for particular disciplines.

HAIGH: So the application oriented aspects of it would now be covered more in more specialized places?

RICE: Right. *Advances in Engineering Software* is a journal, I just got it in the mail a couple days ago, just before I left town. There are several such journals, with slightly different titles, that are, you know, focused on special disciplines.

HAIGH: Well unless you have anything else to say about the conferences and their impact then this might be a good place to stop the first part of the interview.

RICE: Okay. And just for your records tonight I'll print it out, because there's a a web page that has all these conferences of the IFIP working group: http://web.telia.com/~u13107484/wg25/index.html

[End of first session, and of Tape 2, Side B]

[State of Tape 3, Side A]

HAIGH: The interview is now recommencing for a second session, held a day after the first part, on Thursday, the 25th of March, 2004.

So in the first part of the interview we talked about your early life, career, undergraduate and graduate situation, and your time as a researcher with General Motors. We also talked about SIGNUM, about the *ACM Transactions on Mathematical Software* and its origins, and about mathematical software conferences. So if we can now pick up your personal story around about the time of your departure from General Motors and your move to your first academic faculty job, indeed your only academic faculty job, at Purdue University.

RICE: Okay. So why I came here is because I realized that even though it was a very comfortable situation at General Motors, in the sense that I had no duties and complete freedom, I also realized that it was in some sense a dead end position. I would never get anywhere at General Motors if I didn't actually start working for General Motors. So I decided to look for academic positions. The financial situation in the universities had improved greatly in the few years that I was at General Motors, so salaries were now much more competitive than when I first graduated from Caltech.

I started looking at jobs, interviewed at several universities and had various opportunities. I was intrigued by two things at Purdue. One, they had started a computer science department, basically the only really operating computer science department in the country at that time. Two, they had a strong group in applied mathematics in the math department. I was jointly between math and computer science, so that was a good environment for me. I came here.

I continued my research in approximation theory as the main activity, but I also became involved in computing more heavily. I was teaching courses in computing and we were basically inventing the courses as you were going along because there were no textbooks, it was a very early days of the field. And that got me more interested in computing, but I didn't really change my research activities that much. My research was primarily a mathematical analysis type program and with occasional aside into actual computing applications. For probably ten years, twelve years, something like that, I stayed in approximation theory. But I gradually got less and less enthused about approximation theory, I think when I entered the field it was really very dynamic, all kinds of things to do that nobody had been looking at. For many years after, what was it, close to twenty years of work, people were counting the number of angels that dance on the head of a pin as opposed to solving real problems, I thought. I got less and less enthused about it so I basically, in the late 70s, just decided to quit doing approximation theory.

My interest switched to solving partial differential equations. It seems quite different, though of course it's in the same general field, but underneath it's actually a very similar field to approximation theory. I realized that all the machinery that I had been using in

approximation theory applied also to solving partial differential equations, and so I began working more seriously in partial differential equations. I become more experimentally oriented, software oriented, when I made that move, too. The thing that really cemented going into software was the ELLPACK project which started probably in 1976. There were two or three groups around the country who were seriously looking into the question of how numerical methods and numerical software actually works in solving partial differential equations: what's the fastest way to solving linear systems, what's the best way to discretize the systems, and so forth. The University of Texas was in the linear system solving business, and people at Yale were in exploring discretization, Galerkin and collocation and such methods, different ways to approximate the derivatives.

There was a meeting at ICASE; I believe it was in 1976. ICASE is the Institute for Computer Applications in Science and Engineering, it's a NASA research center in Virginia, and for many years it had a small but very active operation in scientific computing. I would guess there were twelve, fifteen positions there, not all of them were filled by permanent people. ICASE had this meeting, with about fifteen people, I would say, representing Texas, Purdue, Yale, and ICASE. We just discussed how we could collaborate to carry out this study. Some people wanted to do one thing while other people wanted to do other things, but they clearly all were in the framework of trying to figure out what methods actually worked well for partial differential equations. So I proposed that we actually create an environment where the people interested in linear equations could stick in their linear equation solvers, and the people interested in discretizations, whose output is linear systems, could create that kind of thing. We could then have some test cases, and we could actually do some systematic evaluation of how various methods worked on various problems, depending on the size of the problem, the type of the problem, and so forth. That was the source of the ELLPACK project.

A few years earlier a colleague, Houstis, and I had been doing a lot of experimentation with the efficiency of various ways of solving partial differential equations and we had discovered a clerical, but nevertheless serious, problem. Mainly that we would write a program, we'd solve a problem this way and we'd have a bunch of output and we'd do it again, and by that time we had stacks of outputs sitting in our offices and we had to try to remember what had been done where. Inevitably these environments for making the experiments gradually changed accidentally as you went from problem to problem, so it wasn't that easy to know whether I really solved exactly the same problem in this run as I solved in that run. So the ELLPACK idea was also to manage this operation, so that you would write down your problem as some high level statement you could actually read a page or two, to say in human terms this is what we're going to do. Then you would connect to big programs that would do that kind of thing. That was first idea of having a problem solving environment for partial differential equations, even though the terminology hadn't yet been invented, but it was a computational management project from the beginning, to allow collaboration of various groups.

HAIGH: Now I have here your article from *Mathematical Software 3*, discussing ELLPACK [J.R. Rice. ELLPACK: A research tool for elliptic partial differential equations software. In *Mathematical Software III*, (J.R. Rice, ed.), Academic Press, (1977), 319-342], and in there you say that in the summer of 1975 Garrett Birkhoff

started discussing the possibility. So would he have been someone involved with the NASA group that you mentioned?

RICE: I don't think so, but Birkhoff was a close friend. I've mentioned Bob Lynch who was here at Purdue and a collaborator; he's a student of Birkhoff's –

HAIGH: Actually the other person you mentioned there, I think, is James Ortega.

RICE: Ortega, that's the ICASE meeting that we're talking about. He was the director of ICASE for a number of years. And it actually lists the people. There was Martin Schultz from Yale, Roland Sweet who was linear equation solver, and Richard Varga who was a PDE man in general. Not all of these people actually participated in ELLPACK, but they all discussed it and there weren't any real objections to it from anyone, but there were some people who were just listeners or commentators. I hadn't actually remembered that Birkhoff had started discussing this idea, but it's certainly possible because he was a very ingenious guy who saw all kinds of things, and liked to keep his finger into the partial differential equations game, even though he had lots of other things he did.

HAIGH: Now you mentioned that solving partial differential equations had become your new main research area.

RICE: Right.

HAIGH: How well understood and developed were methods in that field when you moved into it in the early 1970s?

RICE: I would say it was reasonably mature and very active, much larger than approximation theory field, with many different kinds of players. A lot of engineers were solving partial differential equations, and some of them inventing new methods on the fly. For example, the finite element method was invented by engineers who didn't have any real insight into mathematics; it just looked like a nice thing to do. Fifteen years later mathematicians figured out what they were doing, from a mathematical point of view. There was a lot of activity at that time, but there was also a lot of theorem proving about various methods to solve partial differential equations, how to take the equations and replace them by a linear system. The idea was to get a linear system that was small and easy to solve, and that gives high accuracy in the solution. Finite element methods are one way to do that, finite difference methods are another way to do that. At that time those methods were seen as very, very different. We now realize they're just variations, if you stand at the right place, of the same way of doing something, but that wasn't very well understood at that time. And most of the methods were relatively crude, even in the 70s, compared to what people *can* do now. Not what *is* being done, the engineering community still is very much using simple minded methods that are very inefficient and have very low accuracy, still the most common situation in new applications. But the people who have looked at the area now realize that what's being done in general is not very efficient and there's really many orders of magnitude improvement in efficiency possible which people are not exploiting.

HAIGH: Who were the intended users of ELLPACK?

RICE: Well in the beginning it was actually the researchers that were doing this kind of work. The people who were at this little meeting were samples of that kind of person, so, you know, there's two or three of the people, were really into how to solve linear systems of equations, and with iterative methods or direct methods, and how to make it efficient and accurate, and so forth. And then some of the people were into discretizing the partial differential equations using various Galerkin, collocation and finite difference methods, and those people tended not to be that interested, in the sense of their own research, how to solve the equations once they were created. So the objective really was to try to marry these two groups in a seamless manner because you really can't tell how good a method is for solving a problem if you don't do both of the things (discretize and solve the equations)in a coordinated way , and then measure the accuracy and the effort to get results that are really "down to the bottom line".

HAIGH: So from the point of view of a researcher, the benefit would be that you would take this integrated framework with all the pieces and then you could pretty much swap in a new version of one piece and see how well it worked in your system?

RICE: Right. And you would have a standard set of test problems. A very common idea in mathematical software for difficult areas to describe, like curve fitting or optimization, is to have sets of test problems that people use to see how these methods do on those problems. They then become standardized test points to evaluate new ideas and improvements on old ideas. So that was part of this. The intention early on was to get some standard problems and make them automatically available.

HAIGH: Were the methods themselves that were built into the first version of ELLPACK different from those that would have been propagating widely through other libraries and standard routines?

RICE: Not really, because the methods were methods that some participant in that group was developing. If there hadn't been an ELLPACK they would have still written papers about them. There were some methods that were new in the sense that everybody has their latest version of their method, but we also had people put in, let's say, older, more standard methods, that weren't novel in any way, just as benchmarks, and with the idea of getting a good view of what the field was like.

HAIGH: So in the paper I'm seeing some block diagrams of ELLPACK here with the different modules feeding into each other, which depicts that structure. To what extent was this a new idea on the technical level? Were there any new techniques that had to be developed to implement it?

RICE: Well the first version of ELLPACK was a fairly simple minded kind of thing. It was a crude preprocessor for FORTRAN so that you had a standard set of routines and you didn't have to remember everything. If you wanted to get a certain problem you could just call PDE Problem 13, and it would connect it up to you. But it was really a simple minded high level programming system, without a great deal of programming

language support. You could put comments in a FORTRAN program and those comments would be expanded to FORTRAN code that got Problem 13 for you, that type of thing. So it was not a respectable programming language where you could say "here's the syntax." It was just an ad hoc thing that you had to learn how to do, but it was very good clerical help.

HAIGH: Were there any earlier projects that had made a similar kind of effort to provide application-specific high level tools that were an influence?

RICE: Well I would say at that time MATLAB was also coming out which has a much better understood and a much similar application area because it's matrix and vector analysis which is very standardized and it has a notation that's very widely known, very easy to represent in a computer environment. It's not so easy to do these things for partial differential equations because they have symbols that are not usual and if you put then into the typical computer characters they all look strange and awkward. It was just a much less developed area for implementing a higher level language.

HAIGH: So how did you overcome that problem?

RICE: Well we made this first test at ELLPACK and actually did some substantial use of it. One of the other things that was associated with ELLPACK was what is now called a data base system, which would store everything that anybody had done. So you had to have some record in this data base: we did this problem with these methods on this computer and it took this much time and this much memory, and so forth. So you had a systematic record of everything that was done. Then, after we'd made a first test, we realized that we really should make a programming language that knew about partial differential equations and so that's what the ELLPACK language was. So it was where you basically wrote in a sort of a cryptic way: here's the equation and here's the method and here's the grid and here's the solver. You could have a one page or two page program that would be expanded into twenty pages of FORTRAN that actually connected up to the solvers. So that's where we got into the programming language business. Not in the same sense as FORTRAN and C, but as a user oriented special purpose programming language for partial differential equations.

HAIGH: And were there any examples of such systems outside the area of mathematical software, perhaps in a simulation packages?

RICE: Well the statisticians have been driven in this direction, because most of the consumers of statistics don't know anything… you could almost put a period there. Social scientists, agriculture people, have to analyze data, they don't know computing, they have a weak mathematics background, so the statistics community was gradually driven into providing user friendly interfaces for statistics. I actually gave a talk at one of their meetings that built up to the point that what the statistic users want is a language that just has one statement in it: it's "analyze my data." That's basically all they want, they really don't know about all these statistical tests, they don't understand them, they just want somebody to reliably process their data and see if this is correlated with that, and, if so, how much. This is unlike the people solving partial differential equations.

Even the engineers, who are not particularly sophisticated in pure mathematics, know a lot about mathematic computations and they're not in the same boat as the statistical users.. So the statisticians certainly also have a history of doing something of that flavor and there is a very large statistical software operation in the world similar to for mathematical software. For example, *ACM Transactions on Mathematical Software* has a substantial amount of statistical algorithms and programs.

HAIGH: So attempts to do that kind of thing in statistics predated the ELLPACK idea?

RICE: I would say maybe a little bit, but in the middle 70s you couldn't find what you would call a respectable statistical system either. They were trying to. Spreadsheets were just invented and becoming usable at that time and spreadsheets actually arose out of people trying to do statistical systems in the beginning.

HAIGH: So it was an idea that had been in the air in a number of different fields but nobody had produced very much that worked?

RICE: I think the first really successful ones, in the sense they were widely adopted by users, were MATLAB because it had a very well understood application domain but a very broad one, and the statistical ones because they had these desperate user communities that just could not manage writing FORTRAN programs.

HAIGH: Now in the paper on page 321 you wrote, "One of its primary objectives is to test the concept of a modular approach using interchangeable software parts". What kind of steps did you have to take in order to make the different pieces fit together?

RICE: Well we had to do what was called structured programming, eventually component based programming. You really said, "We have this declaration of what these things do," and that interfaced with the outside world. This was not the kind of thing FORTRAN was designed to support, but languages that came along after that were of that flavor. And in fact, object oriented programming is exactly that. This department made a proposal to the NSF for a large project in object oriented programming before object oriented programming was an idea that was popular, and we got some super reviews and some terrible reviews, and there were senior people on CS that said "it will never work, blah, blah, blah," so we didn't get funded. Five years later object oriented programming, which is exactly what we had proposed to do, was the fashion in computer science. That was basically what these are: we defined PDE objects involved, defined solver objects of various flavors, some of them solve equations, and some of them discretize equations. It was very much that thing, but it was in an application domain, so we did it so didn't have to generalize it to the whole world.

HAIGH: Running alongside the kind of componentized technical architecture the paper describes a social system in which the ELLPACK organization was completely voluntary and contributors used this framework to prepare modules and submit them. It also says that you serve, this is Page 320, as "a coordinator and benevolent dictator." Had an arrangement like that been formalized before, do you know, on any projects?

RICE: I don't think so. I'm not aware of it. For example, in statistics these activities were primarily commercial, or pseudo-commercial. There were some companies that were created in university computing centers and then the people realized that they had a product that would sell, and they quit the university and went into business. The MATLAB project came out of the Department of Energy's interest in linear algebra software and, again, someone saw this was a viable commercial product and went that way. You've probably had lots of dealings with academics, and it's very hard to tell them what to do. Particularly, if you look at that list there's a lot of people there who consider themselves to be the number one math software person in the world, at least half of the people there. Nevertheless, you had to have some standards or you couldn't collaborate. So that's where the benevolent dictator was needed, and there were times when you have two different ideas how to do things, you had to choose. But it wasn't that common, because really most of the people did have a focused interest in one part of the problem and they really didn't care that much what people did on the other parts of the problem.

HAIGH: So the arrangement proved to be successful?

RICE: Yes, it was successful.

HAIGH: How many people on the original list finished up contributing methods?

RICE: Well let's see the list. You have to keep in mind that some of these people are wise men and not software producers, so Birkhoff, for example, who had great insight as to how the world works but, you know, to my knowledge never wrote a line of code. Ortega was a facilitator. The people who really contributed a lot were David Young, Martin Schultz we got stuff from them, we got Roland Sweet's linear solvers, we got some codes from Allan George at Waterloo, but he was not very active. Ortega didn't participate in the project and neither did Fix. Gene Golub didn't contribute anything beyond good advice to the project; he's a linear equation solver type. Varga didn't really participate. I think Widlund didn't like the idea of collaborating with other people; at least I never got that feeling that he was a real collaborating type, so he didn't. So we're talking about maybe half of the people, or a third of the people did something significant, and I'd say most active was David Young at the University of Texas, who was in iterative methods for partial differential equations.

HAIGH: Were there people who were not at the original meeting who finished up becoming important contributors?

RICE: Well many of those people were the leaders of a group of other people. David Young had collaborators, so what I had ascribed to David was actually done by Ph.D. students or assistant professors associated with him. I represented Purdue at that meeting, but many people from Purdue worked on various phases of that project. Besides graduate students don't have much choice what they work on, unlike professor type people.

HAIGH: There was also a list there of anticipated capabilities for the first version. So can you talk about which capabilities made it into the initial release, and perhaps, about the next couple of versions progressed?

RICE: Well let's see. There was operator approximation, in two dimensions; all of what was listed there got implemented. In three dimensions only the seven point star got implemented in a general sense. The 27 point star was only done for a very specialized problem. Lap lace's equation, which is a very interesting test case but it's a very simple case. The algebraic complications of a 27 point star were huge. Everyone who worked on it, including myself, and several graduate students, just drowned in the number of cases that there are on a 27 point star, and we had to actually figure them all out and write many lines of code for each one, and we never got that done. Special options: we did all of those things, the Poisson problem, constant coefficients, self-adjoint form. And direct elimination was very easy to do. We did do nested dissections. We did do fast methods. We did all of the iterative methods, SOR, etc. We had a conjugate gradient routine in there, I don't think it was as sophisticated as it could have been, but we had a respectable one. So the biggest downfall was in three dimensions.

There's a complexity that is not visible in that list of objections, and that is the geometric shape that the partial differential equation lives on. If it is in one dimension then life is easy, or in two dimensions you can have squares, and life is still pretty easy. Once you get circles and ellipses and very strange shapes, there is a geometric problem there that is not something that most people solving differential equations think much about, but that in fact is one of the most difficult parts of the problem. By now people are thinking about it, but in those days people did not see the difficulty of handling curved domains and complicated shapes, and then three dimensions is even worse. And the geometry was the toughest nut in that whole picture and it's not even mentioned there.

HAIGH: So some of those problems became much more apparent as you actually worked on producing it?

RICE: Right, yes.

HAIGH: Now what you suggest here is that you hope that a significant part of the capabilities would be implemented by late 1978, and then it's implied that there would be an ELLPACK 77 and ELLPACK 78, and presumably regular releases after that. Did that happen?

RICE: Most of those were released and there was sort of a final ELLPACK, I don't think it had a number on it. There's a book, it was 1985 before it actually came out. I remember when I was on sabbatical leave in 1981, I was working on final pieces of it, but it shows how long it takes to get books out. So this system called ELLPACK was completed in the early 80s, everything took longer than you'd expect, and it had lots of modules. You can see that there are the standard problems that came with it, so you didn't have to invent your own problems, and it had these fifty odd problems, and they had parameters so you could make the problems hard or easy by adjusting those parameters.

HAIGH: That was the first version? How many versions were there?

RICE: Well ELLPACK 77 and 78 were really thought of as intermediate versions, and then there was an ELLPACK, then this book documents ELLPACK, and that system was actually fairly widely distributed. It was a one hundred percent FORTRAN based system, so all you had to have was a FORTRAN compiler that worked. You would get a magnetic tape, in those days that's how you did things, and you installed it, and you could run ELLPACK. You didn't have to have a systems programmer or such things to help you.

HAIGH: Were there special challenges involved in making the code portable?

RICE: Well the answer's certainly yes. Were there challenges beyond what is normally done in mathematical software? Not too much, because the mathematical software community has always been focusing on portability and we had all of these various kinds of procedures and rules and practices to make FORTRAN portable. The only difficulty was to make the system itself portable. So we had a language processor that was written in FORTRAN because we didn't want to depend on anything else. It's not very common to write language processors in FORTRAN, but it's not that hard, so that was something that was unusual about it. We probably distributed four hundred, five hundred copies of ELLPACK to, can't remember the numbers now, about thirty different countries, it was a fairly widely distributed system. Some of the people you could tell used it, because they kept asking questions. Many of the questions were not related directly to ELLPACK but they were trying to solve hard partial differential equations, and they couldn't get ELLPACK to do it, but the problem wasn't with ELLPACK. It was they had a problem that was hard to solve then you have to have some ingenuity to figure out how to solve it, no matter what you were doing.

HAIGH: What was the balance, among of four or five hundred copies, between people who just had problems that they needed to solve and the original audience of mathematical researchers?

RICE: I would say it was primarily people who had problems to solve. Some place there is a list of who it was sent to and very often you see John Smith at XYZ Corporation, and that's all you know. But certainly it was not primarily the research community that got it; it was primarily people that had problems to solve.

HAIGH: So in the 1977 article you say, "It is plausible to expect if ELLPACK is successful then someone will use it as the basis for user oriented production quality software", and then you say, "This eventuality is seven years in the future". So by the final release of ELLPACK had it evolved to be that tool that you described there?

RICE: I think it was a reasonable approximation to it. It certainly had some shortcomings. The implementation being based on FORTRAN prevented us from doing some things that really should have been done, because in pure FORTRAN, you cannot really make a graphical user interface of any respectable behavior. You have to go into C in search of things like that, and we just made the decision for portability that we would stick to FORTRAN. So we had some shortcomings in user interface, and such things that we regretted, but we didn't see how to get around it in the FORTRAN environment.

HAIGH: So to use the system the user would write it in this ELLPACK language, then feed that into ELLPACK, and then compile some resulting FORTRAN code?

RICE: In fact the ELLPACK system did all of that for them. Unless they were sophisticated, they would never have even looked at the FORTRAN code that was generated. People who knew how ELLPACK worked would often run their problem through ELLPACK, generate FORTRAN code, and then go in and do things to that FORTRAN, to add things that weren't there already. But there were only of few people who could do that. Some very extensive experimental studies were made. In the late 1980s we had what, in those days, was called a mini super computer that ran for three months dedicated to solving partial differential equations to generate data to investigate some questions about how various methods did on various kinds of problems. We had to make it bullet proof because sometimes when you're setting a bunch of experiments something is not right and the system dies in the middle of the night, and you don't want to come in the next morning and find out it hasn't been running. So if it died it rebooted itself and went on to the next problem. We made some very large experimental studies trying to generate enough data to make plausible statements about how methods compare and so on.

HAIGH: So was efficiency of the code a major concern?

RICE: Certainly efficiency of the solvers themselves. The system measured various kinds of time, and people were concerned about how the ELLPACK system itself ran. But they would time it: how long it takes to generate these linear system equations, and how long it takes to solve a linear system equation. That kind of timing data came out and that was of primary interest for us to try to see how different methods actually behave in practice.

HAIGH: During this period of the late 1970s, and the early 1980s, you saw a vast proliferation of mini computers and then of personal computers. What kind of hardware would people have needed to run ELLPACK?

RICE: Well technically all they needed was a FORTRAN compiler. Solving partial differential equations in most cases is computer intensive, so you needed to have reasonably powerful computers to solve interesting problems. For example, this experiment that went on for three months with the mini supercomputer. If it was done today, you know, it might take three hours on a laptop, something like that. So computing was very underpowered then compared to now.

HAIGH: So would a practicing engineer have been able to solve a large number of useful problems with, say, a low end VAX?

RICE: Yes, that's what most engineers had, that kind of computing power. The combination of lack of good methods and the lack of powerful computers meant that engineers had to solve simplified versions of the problems they wanted to solve. If they had a 3-D problem maybe they couldn't solve that, so they took a cross section and solved a 2-D problem, and made some guess of what would happen if they expanded that

into 3-D. If you have cubes, that's not such a bad idea, but if you have some really complicated 3-D object that's not much help, but that's the kind of thing engineers were forced to do in those days.

HAIGH: Alright. Now how easy was it for an engineer, with say a reasonable mathematical background, but by no means a specialist, to figure out which of the methods in the pack they should be using for their particular problem?

RICE: Of course that was the primary motivation for the whole project, was to give answers to those questions. We find, and found then, too, that most engineers actually had learned some method that they liked and they were most likely to use that just because they knew it, no matter whether that was the best way to do things or not. They had the opportunity to explore other things, but again, most people who actually got the system were not interested in comparing different methods for solving partial differential equations, they were interested in solving a particular problem. They figured out how to write their problem down in the ELLPACK language, and they just had to name a method in order to get it applied, and then named whatever they were familiar with. If they got good results they were happy, only when it took too much time or something would they actually start studying, well maybe there's a better method.

HAIGH: So there was nothing in ELLPACK itself to help them pick the method?

RICE: No, No.

HAIGH: Was that something that you went into detail in the book on ELLPACK?

RICE: Well there was a lot of discussion of applications, and there was a lot of discussion of how to solve problems, to do tricks you might call it, to solve problems that ELLPACK really on the surface you might not think was able to solve. There was a lot of data presented on how various methods worked, there was a lot of graphical data, you know, here's a standard PDE problem on a rectangle and here's how nine methods work. It could be, if you're looking for four digits of accuracy there could be five orders of magnitude difference in computer time, so they had that graphical data trying to tell them that, you know, some methods were much faster than other methods.

HAIGH: So they can make a simplified version of the problem. Would all these methods, if you give them the same data and they eventually produce an answer, give you the same answer or do they involve approximations?

RICE: Well, all but the most trivial partial differential equations have to be approximated, and one of the big sources of error is in the very first step of reducing the differential equations to linear equations. Often that's the biggest source of error there. So there's a big difference in what you did in the way of discretizing things. Now there were some methods that were what we would call integrated methods, in the sense that you couldn't discretize and then order the equations and apply this solver. Instead, they did if all as a single package. There were times when there were two methods that actually had the same discretization inside them and they should, knock on wood, get the same answers because the only difference would be how they solved the resulting system in

linear equations. Generally that is primarily governed by either round off error, or if there's iterative methods, by when you stop the iteration. Then one method might decide to stop sooner than another and get a different amount of error.

HAIGH: So back to our hypothetical engineer who always uses the same method. So that would actually be more dangerous then because if they picked the wrong discretization technique then not only might the method take a very long time, but they might actually get an answer and it wouldn't be a good one?

RICE: In fact your first inclination as an engineer is to use very crude methods because the ones that are in the textbooks that you use when you first learn how to solve partial differential equations. Again, one of the motivations for the project was the awareness among people who had looked at it a long time that the easy simple methods were not very accurate, and you would find people who were trying to get accuracy. So suppose you have a rectangle, well they were willing to take ten thousand points on both sides and then they have a hundred million unknowns in that problem trying to get three digits of accuracy correct. Whereas, if they had approximated the partial differential equation by a better method then instead of a hundred million unknowns they could have gotten that accuracy with two thousand unknowns, and if you have two thousand unknowns you don't have to worry too much about what methods you use to solve them. Even in the 80s computers were fast enough to just go blindly ahead and get an answer in a reasonable amount of time.

HAIGH: So once the software was in use at these four or five hundred sites, you've mentioned that you got feedback from users asking for help. Now did you view this as a software product in the sense that bugs will be reported, requests from users will be made, there will be interim releases that fix things and improve things, or was it just something that you said this is finished?

RICE: Well, we kept perfecting the system for our own use, so bugs were being repaired, but we didn't have the same kind of mentality that NAG or IMSL had. We sold people a hundred thousand lines of code for $275, and just for that kind of price you're not going to get somebody who's going to actually provide consulting services and so forth. It wasn't part of the game; we didn't have the resources to do that.

HAIGH: Did you ever have cases where users sent in code, improvements to the code?

RICE: We did get some positive feedbacks of various kinds. Not much, but somebody would say, "I've discovered how to make this ELLPACK system solve a problem," a kind of problem which we hadn't thought about doing. We had people who reported bugs to us, and since we were using ELLPACK we were, of course, appreciative of that and fixing them. We didn't have any users that blindly bought the program and used it and then made some substantial contribution to it, the people who made contributions were people we were already dealing with in one way or another.

HAIGH: And did you view the code as being copyrighted?

RICE: It was copyrighted by Purdue. We had a substantial analysis of selling the code but there were, I believe, twenty-six or twenty-seven places that could claim ownership to some of the code in ELLPACK. We were convinced that it was not worthwhile trying to determine who owned what percentage of ELLPACK in the legal sense, because similar codes had been on the market selling instead of for $275, for $40,000 and so forth. We were convinced that if we took that route all of the money would go to lawyers in trying to figure out who owned it. No one who had contributed code to ELLPACK was particularly concerned about it just being distributed with a message saying that they were the ones that contributed it. So it read actually when you bought ELLPACK you didn't pay anything for the code, what you paid for was actually making the copies and mailing it to you.

HAIGH: So if someone had wanted to run off a copy of the tape for their friend in the neighboring computer center would they have been able to that?

RICE: Right, we didn't have any kind of monitoring or constraint at that time.

HAIGH: So practically they would have been able to do it, but would you have felt that that was something that they shouldn't have done?

RICE: Well, I don't know exactly, we never were aware of it happening. It probably did happen; I'm not sure what our reaction would have been.. I mean we would have been unhappy if somebody had gone into business with ELLPACK, I think. That's actually in a way why we had the copyright; the university said that would at least prevent somebody from making a business out of it.

HAIGH: So none of this code ever made its way into libraries, for example, IMSL, or any of its competitors?

RICE: Well I think some of the components actually came, maybe not directly out of IMSL, but there would be academic types who would contribute software to both places. Much of IMSL's and NAG's code comes from academic organizations, they would give it to IMSL to make a routine, and they would give it us, and so the almost identical code would be in two different places there.

HAIGH: So the author was free to contribute the code to other places, but you just wanted to make sure that nobody took the whole framework and reproduced the whole thing?

RICE: Right.

HAIGH: Can you talk a bit about competing systems that would function in the same general area?

RICE: Well there are engineering systems now which have finally become fairly sophisticated. Probably the first very successful partial differential equation solving system was NASTRAN. NAS is from NASA, I think it's NASa TRANnslator. Its user interface was that you wrote a main program in FORTRAN and called on a set of library

routines. I actually attempted to use NASTRAN long before I got interested in ELLPACK, when I worked at General Motors. These numbers are only approximations, but I got a one thousand page manual and was told that it would take me six months before I had a real hope of running my first test problem. Nevertheless, NASTRAN became a widely used system, and a company took it over from NASA and publicized it and made a substantial business out of it, because NASTRAN would solve a wide variety of engineering applications that people needed to solve. You may moan and groan about spending six months to make the first test turn, but that's a very small investment compared to writing the code yourself, and so engineers learned how to use NASTRAN. And then companies began to appear who made front ends for NASTRAN, and those companies were eventually quite successful. So you now have bridge designers who have a graphical user interface and they sketch out the bridge and put in some numbers, and so forth, and they say, you know, what load will it carry, and they may not have NASTRAN behind it, but they might well have NASTRAN underneath that. There were certainly systems that came out that had that flavor.

HAIGH: And is there the name of any individual particularly associated with the development of NASTRAN?

RICE: There probably is but I'm not aware of it, and I have no idea who was behind it, that is besides NASA.

HAIGH: So by the time you were developing ELLPACK, NASTRAN had already became a commercial product?

RICE: Right.

HAIGH: Now how about the IMSL and NAG products?

RICE: They were also well along as businesses, both of them were organized roughly in 1970, so by the time ELLPACK appeared they had been in business for ten years. Partial differential equations weren't the first thing they put into the library, nor the third, but they eventually got around to putting in some pieces of partial differential equations software. They would have linear equation solvers that were the type that people would use to solve the linear equations that came out of PDEs, or they might have a Fast Fourier Transform solver that solved Laplace's equation on a rectangle as a library routine. So they would have some. Neither one of them got in competing with NASTRAN, for example, because NASTRAN's claim to fame was that you really could, if you worked long enough, describe a pretty complicated bridge, and eventually come up with what the stresses and strains and loads were on that bridge. But you have to invest a lot of effort to (a) learn it, and then (b) to actually put the information into bridge design.

HAIGH: So in the mid-1980s, when the ELLPACK book was available there were a lot of people using it. If you've got a bunch of partial differential equations that needed solving, what were the strengths and weaknesses of ELLPACK versus the NAG and IMSL products?

RICE: Well ELLPACK was much more versatile in what it would deal with for a given level of effort because solving partial differential equations just wasn't NAG's and IMSL's focus. They weren't really trying to be especially versatile, and ELLPACK was also much easier to use than a library routine, because if you had a complicated problem you have to describe it in a FORTRAN program to use the library routine. ELLPACK, even though it didn't have a graphical user interface, had a reasonably efficient way to describe domains. You described a domain by having segments of the boundary written in parametric form, so for ordinary shapes, you could make twenty pieces and say, "this is straight line," "that's straight line," "that's semi-circle," and so forth, and fairly quickly describe a really complicated shape, in an hour or two unless you had to actually figure out for yourselves where the points were. But, if you actually knew the geometry well, you could get it in, and then ELLPACK would digest that information and use it to figure out where the domain was and where all of the grid lines intersected that boundary. All of that kind of thing was done, and that was not something that you would find in a NAG or IMSL library, you would have to write the FORTRAN code that did all of that.

HAIGH: So would the benefits for the library approach be that you can flexibly incorporate the routines into a larger system that you create?

RICE: That's one advantage, and you have much more control because you're working at a lower level. Again, most users of ELLPACK were not able to go inside of ELLPACK and massage the FORTRAN programs that ELLPACK generated. But the experts would actually do that, if you have some special kind of situation that ELLPACK couldn't do automatically, you got the closest you could in ELLPACK, and then you would take the code and you would change twenty-five lines down there, so it did what you wanted it to do. But you had to know what ELLPACK was doing, where it was generating those lines, and so on, but there were a number of people who could that.

HAIGH: By and large if you were and end user with a problem to solve you would have been much better off with ELLPACK?

RICE: Right.

HAIGH: I know during those years the phrase "fourth generation language" got thrown around to describe end-user development systems focused on particular application areas. Were you aware of the general wave of hype around the idea?

RICE: Yes, I was aware of that. Most people in the fourth generation language business would not think of ELLPACK as being a competitor because they were violently anti-FORTRAN, and since it was a FORTRAN based system they wouldn't consider it, really, as an example of such a higher level type language.

HAIGH: One last question before we move on. I see here that at least the original work on ELLPACK was supported by an NSF grant. What was the grant for, was it for research in software methods (like EISPACK), or was it actually because they wanted to help people solve their differential equations?

RICE: Well the proposal was fairly much describing how we want to evaluate how methods work and that we want to develop new methods. The ELLPACK project did generate new methods or, included the first real implementation of methods that in some abstract sense existed but nobody had ever tried them out. So it was a combination of trying new methods and doing experimental studies. The NSF grants had both these components.

HAIGH: And was it easy, at that point, to get grants to work on software as opposed to something that you can spin as more conceptual mathematical research?

RICE: Well it's easier to do theoretically oriented, numerical work and get support for it than more practical work. Yes, it's always been easier to get support for theory.

HAIGH: And that's stayed true through to the present?

RICE: I think in numerical computing, scientific computing, it's still easier. Of course there's parts of computer science where being anti-theoretical is the norm. We have guys on our faculty here, where if some candidate comes in and gives a talk and there's an equation in his talk here, it's down the tubes. Some faculty don't want to see any analytical methods in systems engineering, which there needs to be if people are going to figure out the performance of networks systems, and operating systems, and so forth. You have to have a little bit of modeling and analysis in there, but certainly we have a couple of professors who don't want to see it, that's all intuitive analysis. But in the scientific area, theoretical ideas are better than having a big, important, but messy, software idea.

HAIGH: Did that grant pay for the conference, did it cover any of the actual development work?

RICE: Which conference were you talking about?

HAIGH: The original two meeting with about a dozen people, when you came up with a plan to produce ELLPACK.

RICE: Well that was actually paid by NASA out of Jim Ortega's discretionary funds, just as a one time thing. So that, and Ortega, would help ELLPACK in some indirect ways because he liked the idea, such as inviting people to come down to ICASE and paying their way, but he wasn't in the business of giving grants to universities. So we got NSF grants to focus on various pieces of the ELLPACK project. And then, maybe it was 1982, there was a program in the NSF that lasted for five years to, give large grants to departments, and several million dollars to groups. So we had a project called Computing about Physical Objects that involved geometric modeling people, parallel computing people, and solving partial differential equations people. ELLPACK was a major component of that proposal and activity for five years.

HAIGH: Okay. Well that leads us nicely then into the follow on project then.

RICE: Right. During that time I became very interested in very sophisticated computational environment for applications involving geometry modeling and all of the

aspects of scientific applications. So it gradually led to the idea of this problem solving environment which was crystallized, I believe it was 1987 or 1986. There was an NSF workshop that crystallized this idea.

HAIGH: Can you describe the idea?

RICE: Well I think the idea is that it's a natural extension of what ELLPACK was trying to do. You have to have a natural way for people to state the problems they want to solve, and the attributes of those problems that they think are important, and then you need a system underneath that that knows how to connect that up with the software that will actually solve that problem. It's, in a way, like what Word does for typists, but fortunately for Word it has a much easier environment and there are simple ways to say "I want the font bigger," "I want it smaller," or "I want the margins moved." You talk in terms of the visible level of the problem and Word has somebody down there who knows how to make fonts bigger and smaller. When TeX first came out, you know, you had to describe how your X looked and everybody was all intrigued with that. I know of people here who didn't like the super script alpha on TeX and they went in and they'd write their own super script alpha and put it in their papers, and that kind of thing was what you could do in TeX. Well Word talks to secretaries pretty much in the secretarial language, and that's what you want to do in scientific applications too, and so that's the idea of problem solving environments. Communicate with people in their own language, as close as you can get. Like Word, you have to learn some new concepts to use Word but you don't have to go off in the abstract mathematics or engineering technology, or artistic things to get a lot of use out of it.

HAIGH: And your work in this area was this also focused on partial differential equation solving?

RICE: It was, because that was what I was familiar with. That's a rich area because it has all the ingredients of mathematical software, and there's geometry in there which is still a very difficult area for computation in spite of what you see in the movies. And just to give you an idea of how geometry is different from everything else in computing, you can, at the hardware level, add numbers, process a string of data and all of that kind of thing. But there is no small atomic geometric operation that corresponds to what people think about. You scratch on the blackboard there's a very small amount of information, really, when you make a diagram there. There's not any analog of that. If you put twenty-five numbers on the blackboard and add them up, the computer does that and it's the same thing. The huge geometry computers don't have any underlying geometry operations; they make thousands of points, and are very expensive and very hard to get going. So geometry that you see in the movies now is a testament to how much computing power there is in the world, because it's not that we understand how geometry works. We just absolutely don't have a clue, in my opinion. So anyway that's what we need, is a geometric engine to do it, to have real problem solving environments.

HAIGH: Okay. So, backtracking to the meeting. Who else was there and what kinds of things did you discuss?

RICE: The NSF meeting. There were probably thirty people there. We discussed this whole idea of trying to automate it and what areas were ripe for it. People like the MATLAB people had, you know, an area that was mature by that time and so really it was a successful point because you could just take the things out of the textbooks and put then in MATLAB and you got the answers out of them. So that was an example of the kind of thing. There were people from a variety of mathematical disciplines, and numerical software disciplines. There was a report that came out that was by Houstis and Gallopoulos and me. [E. Gallopoulos, E.N. Houstis, and J.R. Rice. Computer as thinker/doer: Problem solving environments for computational science. *IEEE Comp. Sci. Engr*., 1, (1994), 11-23]. It summarized the idea, and a version of that was in the IEEE journal that was just starting up in those days on *Computational Science and Engineering*. I believe the title has changed slightly, it's still CSE but it's *Computing in Science and Engineering* now, let's see here, it's on Volume 6. The journal had merged with another journal, that's why it's only on Volume 6. That title is "thinker" because I'm a strong proponent that the idea that you have to have some intelligence in what you're doing, again, another thing that's unpopular in this department, but I think that in spite of many decades of frustration AI is really going to succeed someday and it's beginning to do that. That's part of what one has to do make automated problem solving, and that's what you're trying to do, you're trying to automate problem solving. You have to have substantial intelligence in there to do that.

HAIGH: So were there any important ideas that you came away from the meeting with, or that you heard articulated there for the first time?

RICE: Well it's hard to say. Certainly I don't have a recollection of going away and saying well here's a great new idea. There was a lot of energetic discussion and so I think it focused on lots of things and crystallized the idea.

HAIGH: And then on the book part of your resume, I'm seeing that with E.N. Houstis and R. Vichnevetsky you edited three volumes, *Intelligent Mathematical Software Systems* in 1990, *Expert Systems for Scientific Computing* in 1992, and in the same year *Artificial Intelligence, Expert Systems and Symbolic Computing*. Can you talk about those books?

RICE: Well they're all trying to bring in the idea of how you would apply artificial intelligence in scientific computing. In a way, it naturally flows out of the ELLPACK environment. You really want to have this data that ELLPACK would generate for you about how methods work on various kinds of problems, brought to the surface and made available to a user. Well, there are a lot of different approaches to AI, and actually though they look very different, they're seen not to be that different in how they perform. The easiest one to understand is the following: say you have a new problem, that nobody's ever solved before, and you want to know which method to use. So you have some attributes of that problem, like it's in two dimension that's a curved domain, one of the coefficients is very big. You have some data like that, something a user would know, you take those attributes and you go into your data base, and you find the closest problem you can, however you measure close among problems and there are various ways to do that, to that problem and now then you see how the methods perform on that problem and for

the new problem you choose the best method on the closest problem. Now AI actually has a name for that method, and actually it's a very effective AI technique, it's simple minded but it's often as effective as neural networks and all those other kind of things. That's an example of AI.

HAIGH: That would be case based reasoning, wouldn't it?

RICE: Case based reasoning, right. But you can take the data that ELLPACK would generate and train a neural network on it, for example, and we've done experiments on how well various AI techniques work on the data the ELLPACK generates. That's why I've come to the conclusion that they perform similarly. At any particular point they're quite variable, but overall success rate one method seems to be as good as another and case based reasoning has the advantage that it's easy to do. Although actually in ELLPACK, and the next generation of ELLPACK, we didn't use case based reasoning because we had an AI guy that liked another way of doing things.

HAIGH: So this must have brought you into much closer contact with the AI research community?

RICE: Well you might think so. I'm not sure that people in the AI community would recognize me as having ever done anything there. I taught AI here a couple of times just so I could learn it, and followed it from afar. As you probably know, AI had a very disappointing, at least to most people, level of development for several decades, and it had a very bad reputation in this department here. A substantial group of faculty members will never approve of hiring a person in AI here, we no longer offer a course in AI, which is an absolutely crazy situation, but the anti-AI faction in this department is quite strong.

HAIGH: So Houstis and Vichnevetsky are they also specialists in scientific computation and numerical analysis or is their background in AI?

RICE: Houstis has a very similar background to me; he's my Ph.D. student, graduated in 1976, or something like that. Vichnevetsky is a more theoretical person, he's probably older than I am, and he's in scientific computing. He is the president of IMACS, that's the International Association for Mathematics and Computers in Simulation.

[End of Tape 3, Side B] [Start of Tape 4, Side A]

RICE: Vichnevetsky has been the man whose run that organization for twenty-five, thirty years. His background is engineering and scientific computing.

HAIGH: So this was very much a case of a push from the scientific computing community to investigate AI tools, rather than any AI people reaching over?

RICE: Right. We never got very many AI oriented people interested in what we were doing. If you look at those books most of the people were more like us, they were engineers who thought AI should be good for engineering, and they learned something about it and they had this application or that application. There was a much smaller set of people at AI that thought that engineering was a good place for them to work.

HAIGH: That's interesting. You've talked about these AI techniques to help the system choose which methods are appropriate. Are there any other important aspects that you would see going into making up a problem solving environment?

RICE: Well you do have to have this data that the AI works on, it has to be put in there in some way, sets of rules, cases, whatever kind of thing, and then you need the operations implemented again. So if you look at MATLAB, as an example you can understand pretty easily, in MATLAB when you solve a system of linear equations it basically has a solver that you name when you solve it. If it was a problem solving environment, you would describe the linear system of equations and just say "solve it" MATLAB would examine the linear system of equations in some way, and would know what kind of method would be best to solve it. Whether you use Gauss elimination, whether you should use a band solver or an iterative method, that's the kind of thing which I think would be practical to do now. You could be quite successful in that, because that problem's well enough understood, you could really do well.

HAIGH: That sounds to me very similar to what happened in database systems with the transitions to the relational model, a shift away specifying a detailed series of steps to a high level logical statement of what needs to be done.

RICE: I'm not in databases so I don't have a clear picture of that, but I'm generally familiar that they made this shift of paradigm in how they approached things.

HAIGH: And to what extent have these ideas now been fed into working or experimental versions of ELLPACK?

RICE: Well ELLPACK has in it a very large data collection management system and a learning system to generate information to choose methods. But ELLPACK is really somewhat dormant now, for various reasons. One is that my interests have changed. Also, the AI person that was leading this work has moved to another university. We had a couple of AI people involved in it at one time, and Houstis is only half time at Purdue, he's now half time in Greece. It's not an active project anymore.

HAIGH: So does that mean that the work on problem solving environments in general is more dormant, or are there other avenues in which you've been pursuing it?

RICE: The idea of problem solving environment is, I think, been widely adopted. Different people have a different view of exactly what a problem solving environment is. There have been a number of international conferences on problem solving environments; the Japanese are particularly enthusiastic about this approach to things.

HAIGH: So would some people's approaches to it put more stress on the user interface, rather than your emphasis on AI?

RICE: Well certainly, if you look at the original ELLPACK, the user interface design was pretty simplistic. After all, it was a line oriented user interface. The next generation, which was called Parallel ELLPACK and PDE Lab, had their versions of the same thing.

It had a graphical user interface, was not FORTRAN based, so it had fancy graphics and you could draw a domain on the screen that you wanted to use.

HAIGH: I'm seeing a publication on Parallel ELLPACK from 1992, so was that a development that was happening in the early 90s?

RICE: Right, that's when it started. And Parallel ELLPACK has had two thrusts. One is actually to put numerical methods in it that were parallel, and to use parallel computers. In the process the user interface was completely redone and so now Parallel ELLPACK has, I believe, a million and a half lines of code in it. So it was really a major operation of integrating very large software packages and it had FORTRAN, it had C, it had LISP, it had assembler, it had all kinds of languages there. Portability went out the window, and there was no idea of having other people use it. It was just going to run here at Purdue so we made it work on our environment, period.

HAIGH: And did this follow the original model with modules being contributed from many different places or were they all produced within Purdue?

RICE: We took over all the old modules that were the original ELLPACK. Then more modules were put in, they certainly came from a variety of places. Many were public domain parallel codes, because we could bring in modules more easily. So we would bring in interesting codes that we could get for nothing, so to speak. We wrote some ourselves that were parallel oriented, did a lot of work on trying to understand parallel computing.

HAIGH: And do you think you made any particularly important developments or impact from this project?

RICE: I don't think it's had a great deal of impact. It hasn't been widely used because it won't move. We actually made a network based version of it, so you could use it over the net, and we had a few people do that, but it was fairly cumbersome, and we didn't push it. But I think that's the wave of the future is not to take the software to the user but have the user come to the software. So you can make some big clunky system like Parallel ELLPACK work on a particular computing environment, and people will come to it, you can export a user interface that's well defined and easy to move around and it knows how to get home where the computing power is, and then that links into distributive computing at some time.

HAIGH: I saw a reference to that in Parallel ELLPACK on your website. On your résumé I'm seeing a number of publications to do with "//ELLPACK." Was that another version?

RICE: "//", that's parallel, that's just another notation for Parallel ELLPACK.

HAIGH: Okay. So the regular ELLPACK was finished in the mid 80s –

RICE: Right.

HAIGH: These other versions had been used only at Purdue?

RICE: Basically. A couple of students had got them up and running when they moved to other universities, but that's, you know, by their diligence, not by any effort on our part.

HAIGH: So these days, say from the mid 1990s, onwards, if you're a practicing engineer with some differential equations to solve what software would you use now?

RICE: I think you have to go to these commercially available systems, which are relatively expensive, and they don't have this flexibility of choice of methods and such. Most of them basically implement a method, maybe two methods, and they'll have a Fast Fourier solver because that's so good you can't resist it, and then they'll have a finite element solver, and that's it. They don't put their effort into making lots of numerical methods available.

HAIGH: So in terms of the mathematical sophistication, the software that engineers would actually be using has become less sophisticated?

RICE: It never got sophisticated. The last few years I worked with some people in mechanical engineering on computations on gas turbine simulation. It was a big time computational problem, I mean huge, and the focus on this was to figure out parallel computing and such things, not user interfaces. The numerical methods underlying these codes that people are using are so primitive, just incredibly primitive, and they don't seem to be worried about it. Of course in their calculations, if they're off by twenty-five or forty percent that's not too bad because they don't understand the physics very well inside gas turbines, and there's all kinds of key things that you need to know to solve the problem that they don't know. So they're very happy with really very crude methods that are very inaccurate. Their methods, no matter how fast they make the supercomputers, will never compute accurate answers to those problems if they never understand the physics. But that seems to be a lot of what's going on, and I think a lot of engineering applications are that way. I think even the people that are building bridges are using very primitive methods by and large, the finite element methods are not sophisticated, that's my opinion.

HAIGH: That's interesting compared with something Cleve Moler said, though obviously the areas that MATLAB's been strongest in are better understood. Basically, he said that when they produced EISPACK originally they were bringing new and improved methods that hadn't been widely used to practitioners. But then he also said that, in more recent years, the relationship between the software and the research community in numerical analysis had become much less important because the methods that were used in MATLAB, in his view, were meeting the needs of ordinary practitioners pretty well and didn't need a dramatic overhaul. Now it seems like you're saying that in this area there's still this enormous disconnection between what practitioners are using and work that's being done. I guess there are two halves to my question then. One, why is your area so different, and the second half to the question is, is it the case that researchers are continuing to come out with better methods or is just that

they came up with these great methods twenty or thirty years ago and people still aren't using them?

RICE: Well I would say that the main difference is that MATLAB is not being used for serious problems. Look at gas turbines. It is not unheard of for a gas turbine calculation to run for six months on the biggest computers around, and a week is not uncommon at all. They're using very crude methods. You're not going to run MATLAB, that's not a MATLAB application, and I think if you look at some of the frontiers of engineering computation there are people struggling with incredibly complicated problems. You know, Parallel ELLPACK isn't going to do it either, but it is frustrating to see that they could get the same inaccurate answers in six day instead of six months if they used better methods. Well to give you an idea, one of the things that is related to this is diesel engine operation. It's very poorly understood, and there are schemes where you can actually find out what's going on inside a diesel motor, and the best calculations that people can make have errors roughly at fifty percent of what goes on inside there. The diesel engines have transparent blocks so they can see inside, then they actually set off a diesel charge and the gas is burning and it all explodes. They make a calculation and they compare that with the pictures they have, fifty percent error is from the best calculations. And, furthermore, the next cycle the variations are huge, so it's very hard to know what's going on inside these things that are real important. They're not MATLAB applications, and, you know, MATLAB is a very expensive overhead in computing power unless you can use their built in modules. I mean the modules are fast but the rest of the code is incredibly slow, but it's convenient.

HAIGH: Okay. So then people with really big problems would be one group that could definitely benefit from better methods and efficient implementations. Now are researchers in this area continuing to produce new methods which if people with these kinds of diesel engine or gas turbine problems knew about them, would be directly useful, or is it an area that has lost steam as a topic of active research?

RICE: Well I guess; it's difficult to say what's going on. Take this guy I was working with in mechanical engineering, who worked for twenty years in the gas turbine industry. He's a distinguished professor and world famous guy, but he wasn't too interested in knowing that his methods were inaccurate, it just didn't seem to him to be the problem. But the day will come. There are problems that are better understood than gas turbines, because they don't even understand the physics at various key points in combustion. If you take another guy that I've talked to, a guy in nuclear engineering that's looking at flow in pipes, which is pretty chaotic looking, but it's actually a much better behaved problem so you don't have the difficulty that you don't understand the physics. It's just that, because of the bubbling and so forth, it's complicated but you know what's happening in principle. It's problems like that where you can calculate accurate answers for very complicated problems if you have enough computing power, but the people there are using the same crude methods that the gas turbine people are using.

HAIGH: Well I remember you said earlier that at one point you were losing interest, because you felt that the work on approximation had reached the "angels dancing on the

head of a pin" stage. Did work on partial differential equations solving reach the same point, or are there still big questions to be addressed in that?

RICE: Well, I don't visualize new numerical methods appearing. You can always be surprised, but I don't visualize it, the waterfront has really been worked pretty hard. The advances in this sort of mathematical technology will be in the handling geometric complexities; that's part of the difficulty in applications. You think of bubbling, well there are lots of bubbles and bubbles have funny shapes that oscillate; that's hard to approximate well, but it's fairly plausible that you can do it. It's not like the, when a diesel explosion takes place you don't even know what the chemistry is there. And, you know, the methods that are being used to approximate bubbles are very crude and there is the potential to put in spline surfaces which is the kind of thing that many of the people in the areas that are dealing with bubbles don't really know, and don't know what the movie producers know: that you can make good approximations for things that bubble along, and you could put that in and you could get really good results for those kind of calculations.

HAIGH: Let's rewind then back to some things that you said right at the beginning of this session. Now I should make clear for the transcript at this point that you've talked in an earlier interview with Bill Aspray, which hopefully will be getting transcribed as part of this project. It's the Charles Babbage Institute Oral History 320, conducted in 1997. In that you talked a lot about Purdue Department in general and its place in the development of computer science, so I will refer people who are interested in that particular topic to that transcript. Also, you have in here this book, "Studies in Computer Science. In Honor of Samuel D. Conte, John Rice and Richard A. DeMillo, 1994, and in there are two articles about the history of the Purdue department. In both of these, you very much try and give a very realistic kind of story stressing good and bad things together and speaking quite frankly. Those things aren't personal memoirs though, so what we could address is the relationship of these things to your personal career.

You've mentioned that one of the attractions of Purdue was the work that was being done there in applied mathematics. Can you talk about the specific people who attracted you and what if any collaborative work you finished up doing?

RICE: I was attracted to those people not because I wanted to work with them, but because they were senior leaders in the department who were applied mathematicians and you had the feeling that applied mathematics would be reasonably well respected in the math department. I don't know how familiar you are with the sociology of math departments, but applied mathematics around the country varies from being treated poorly to incredibly obnoxiously treated. So if you're interested in applied mathematics you have to be very cautious where you go, and so I came here because at that time the applied mathematicians were a significant proportion of the good people in the math department at Purdue, so I felt it would be safer.

Of course in the computer science department I didn't have any concerns about academia, but it's fair to say that applied mathematics is still in big trouble at Purdue University. In spite of the fact that over the years that they've had a number of really good applied

mathematicians, the majority of the pure mathematicians wonder why the University is wasting their money on these guys' salaries. I was fortunate that I came here as a full professor to start, so I didn't have to worry about promotions, but I've seen applied mathematicians get turned down for tenure just because there were people, people I know well, who just could not bring themselves to vote for an applied mathematician to get tenure. They were leaders against discrimination against women and minorities, but to discriminate against applied mathematicians that was no problem. So that was my main concern there, and I was aware already that applied mathematics had problems at different places in academia.

HAIGH: So is that why you were in, what was then very novel, the department of computer science, rather than sticking with the mathematics department?

RICE: Well I was actually paid by both departments. I was very active in both departments; I was a member of the math graduate committee for many years, taking an active role in the department, taught courses in math as well as in CS. I only stopped being active in the math department when I became head of the CS department, because it was sort of implausible that the head of another department would be active in math.

HAIGH: So then it was quite a gradual transition in terms of shifting your disciplinary identify from mathematics into computer science?

RICE: Right. Because my research for the first, you know, fifteen years out of school was primarily mathematics, and my mathematical credentials still exceed those of most professors in the math department, though I haven't written a math paper in fifteen or twenty years.

HAIGH: So when you first arrived did you feel fully at home in computer science?

RICE: Yes. Of course it was a very fluid situation, nobody really knew what computer science was. But I felt very comfortable with the situation and I didn't have to worry about problems of promotions and such, so I didn't have any personal difficulties.

HAIGH: Yes. And how about the transition from the research lab to the university, were there any aspects such as teaching or departmental politics that were a shock?

RICE: Not really. I of course expected it to be a big change so, you know, overall was happy with the change. I mean it was nice at General Motors to have forty hours a week to spend on your own research, no students, no nothing to distract you. But certainly in the beginning, when I was teaching all these computer science courses, I learned a lot about computer science which I didn't know.

HAIGH: So what were you teaching?

RICE: Basically half of the courses were in math, things like graduate courses in approximation theory and mathematical analysis and things of that type. Then I would teach numerical analysis courses, linear algebra, and numerical linear algebra. For about five years I taught the first course in computer science, wrote a book in that area which

was quite successful for a while. It was the first introductory computer science book that wasn't just how to program an X.

HAIGH: And that came out of a course that you had been teaching here?

RICE: Here, right.

HAIGH: Actually I want to talk a little about your books. So that was your second book. [J.K. Rice and J.R. Rice. *Introduction to Computer Science*. Holt, Rinehart, and Winston, (1969)]. Now the first one was the "Approximation of Functions, Volume 1, Linear Theory" [J.R. Rice. *The Approximation of Functions, Vol. 1: Linear Theory*. Addison Wesley, Reading, MA, (1964).] What kind of book was that, was that a textbook?

RICE: That was a graduate math book, right. In fact I used it as a text when I taught approximation theory as a graduate course in the math department, here.

HAIGH: How did it differ from books that had been available previously?

RICE: Well there had not been a very recent book on approximation theory, I think the preceding book was probably in Russian by Achieser in 1952 or 1954, something like that, so it was ten years earlier. There weren't many books in approximation theory then. There was one that came out in Romania at a similar time, a very nice book.

It was not terribly innovative for somebody in approximation theory who was up on what was going on in approximation theory. It was a compendium of basic stuff from classical approximation theory, and a few newer topics.

It was something less than a research monograph. That was in two volumes, the second volume was really a research monograph book. [J.R. Rice. *The Approximation of Functions, Vol. II: Nonlinear and Multivariate Theory*. Addison Wesley, (1969)].

HAIGH: Now were those books widely used?

RICE: Well for approximation theory they were. Approximation theory was not a very big area in mathematics, but they were very widely used and known in the approximation theory community.

HAIGH: So then your first three books really were all textbooks. You've got the 1969 *Approximation of Functions*, the 1973 *Introduction to Computer Science*, then you're one of many authors on *Computer Approximations*, and you're the sole author of *Approximation of Functions, Volume 2*. So over that five year period you had three main books, all textbooks. That's a very worthy thing, but I understand that within computer science it's considered a lower status activity than producing the same quantity of peer reviewed papers.

RICE: I guess I was in a position of not being very concerned about my career development, because I was publishing more ordinary papers than other people were

anyway. So when I got interested in this introduction to computer science, we had a big fight in the department over how to redo the department's introductory course and I won the fight by volunteering to teach the course, [Laugher] and actually the other author there is my father who helped me with the book. He didn't have much input into the, you know, intellectual content, he didn't know that much about computing, but he helped write it. [J.K. Rice and J.R. Rice. *Introduction to Computer Science*. Holt, Rinehart, and Winston, (1969)].

HAIGH: So if you won the fight, and that's reflected in the textbook, what kind of issues were you fighting over and what was the result?

RICE: Well the previous course the textbook was a FORTRAN manual, McCracken's I think, just how to program in FORTRAN. In my view was there was more to computer science than how to program in FORTRAN. So if you look in that book you'll see that there's things like algorithms are discussed, and data structures, and various things of that type, and there's not a great emphasis on learning how to program in any language. The bulk of the book does not have a programming language in it. There are two appendixes, one's FORTRAN, and one's Algol. The thing professors are always talking about you should be teaching principles instead of skills, so that was an attempt to do that. But then I wrote all of these books which are variations on computing with FORTRAN and BASIC and PL/1, are all basically the same book with just different programming languages in them. Computers: Their Impact and Use with BASIC, with FORTRAN, with PL/1.

HAIGH: So what was in the *Computers: Their Impact and Use* books [starting with R.E. Lynch and J.R. Rice. *Computers: Their Impact and Use - with Basic Programming*. Holt, Rinehart, and Winston, (1975)] that was different from the *Introduction to Computing* books?

RICE: It was basically for general non technical students, so it talked about the social impact of computing, and the history of computing, and very minimal programming capability. A course that we taught at Purdue for liberal arts and such people was the audience. The *Introduction to Computing with FORTRAN*, which came out a number of years later [R.E. Lynch and J.R. Rice. *Computers: Their Impact and Use - with Fortran Programming*. Holt, Rinehart, and Winston, (1977)], was again, a first course in computing for students, which recognized the fact that even though in principle you want to teach principles, in practice the students want to learn how to program. So you have to sneak the principles in while you're talking about programming. That was a different approach to things.

HAIGH: What kinds of motivations and satisfaction did you get from all these books, because obviously this must be something you enjoy doing.

RICE: Yes, I like writing books. I mean books you don't have any referees you have to fight with. I've put chapters in books which I thought, well, probably not very many people are interested in this but I think this is neat and I want to write it down. You can put it in a book; you'd never get it in a journal. You have complete editorial control, and

you can do what you want and if people like it they can say so, and if they don't like it they can ignore it.

HAIGH: And that introduction to computer science textbook, the original version, 1969, some translations, was that widely adopted in other departments?

RICE: It was. It was a very good seller for a few years, until other people found out what I was finding out, is that students don't want to learn principles, they want to learn how to program.

HAIGH: Other than this stress on principles, would you say that there's been anything that's distinctive about your teaching style or philosophy?

RICE: Any things? Well I guess I've tried to choose some middle ground, between being very mathematical and theoretical and completely intuitive. So I like to have a combination, and most of the courses I teach are of that type, where you prove some theorems and learn various definitions and why this theorem is being proved, what it's good for. I think it's some middle ground between those two. There are a lot of people in numerical computation who teach a course in mathematics, and the textbooks are of that nature. They're just the mathematics of numerical computing; they're not telling you anything else about numerical computing. I wrote a book on mathematical software: how do you design a library routine, why is it a problem to design library routines, why are subroutine libraries so hard to understand, issues like that. So my style, compared to the average, is maybe more down to earth. It's easier for me to do that, because I don't have to prove that I know how to prove theorems, because I've already proved so many that I can actually outnumber almost everyone else in that department. So I don't have to prove myself as I'm a theoretician.

HAIGH: Would you say that over time the relationship of computer science students to mathematics has changed?

RICE: Oh, yes. In the beginning computing came out of engineering applications and scientific applications, so everybody was an engineer or physicist or mathematician. Systems programming was nonexistent practically, as systems were very simple things. Now they're plenty of people who get a Ph.D. in computer science who really don't understand a lot about sophomore level mathematics, they don't see why they need to know how to integrate or differentiate or whatever. So it's mathematics, and the things that are mathematically sophisticated, they tend not to want to learn about. The analysis of the behavior of networks, for example, is a very sophisticated mathematical problem, it's not integrating integrals or such things, but it's a hard problem that requires very serious mathematical analysis. A lot of people just want to do it on intuition.

HAIGH: Is scientific computation still part of the core curriculum at Purdue?

RICE: Well, due to the political situation in the department it's almost fair to say we don't have a core curriculum anymore. If you had to have a core curriculum, scientific computing might have one course out of ten, or something like that. But I've mentioned already that artificial intelligence, which is a central, in my opinion, part of computer

science is just missing. The AI course in this department has probably been taught twice in the last ten years. To me it's ridiculous, but it's not my fight.

HAIGH: And would engineering students take your courses, or would they get the same material covered in their own departments?

RICE: Courses I teach tend to have substantial number of engineering students. Maybe a third engineering (and in engineering I would throw in the physics students and such, there's not very many of those), a third math, and a third CS. That would be the typical breakdown of the course I teach.

HAIGH: Actually I have one question related to the material you've written about the history of the Purdue department. You talk about a string of crises and the departure of a large number of the senior and more promising faculty, some of this during the period you were serving as department head. Did you ever feel tempted to leave?

RICE: Not really. I was very comfortable, you know. Not only in the department, my family liked it here, I liked living in a small town. I've lived in placed like Detroit and Washington and Los Angeles, and I didn't care for those places. So I wasn't that tempted to leave, I never actively looked for a job. Well, in retrospect I probably should not have become department head, I guess I didn't dislike it too much, but I resigned once after about five years, and I was talked into staying on. But department head positions in universities are really losers for faculty. Unless you want to become the president of the university, then the first step is to be a department head, but I never had any ambitions to become an administrator in the university, so I certainly wasted a lot of energy being a department head. You know, somebody has to do it, but I should have let somebody else do it.

[End of Tape 4, Side A] [Start of Tape 4, Side B]

HAIGH: One other thing that must have been important to you, which you didn't have in your lab, would be the relationship with graduate students.

RICE: Certainly that's a good thing about universities. I had a very stimulating environment at General Motors, even though it was a very small group. I mentioned Carl deBoor, he was there, I mean the guy's incredibly stimulating to work with. Robert Lynch, who came on the faculty a year after I did, he was also a close friend. General Motors had consultants who came there fairly often and they had Garrett Birkhoff, who was at one time one of the most famous mathematicians in the country, he's very interested in applied mathematics among other things. And they had a consultant who they just hired for me from Harvard, Walsh, who was the leading American figure in approximation theory. He would come for a couple of weeks a year, so it was a very academically stimulating environment for me. I had all this freedom, and Carl was interested, he came there he didn't have much. He was so young and really only had one year of graduate school, but he became interested in approximation theory and that was his whole career. So we would talk about that a lot and had a lot of interactions, it was a

very stimulating environment, unlike what most people would have found at General Motors.

HAIGH: So I've got a list here of your Ph.D. students, and I recognize some of those names from people who've done significant things in mathematical software. Could you perhaps talk more about your approach as an advisor, or any work with your students that you're particularly proud of.

RICE: I really liked the thesis of John Hoff who sort of disappeared into the woodwork; he still actually lives in the San Francisco area. He was a professor but he never did anything after his Ph.D. thesis. Hermann Burchard is Carl deBoor's brother-in-law and by far the smartest student I've ever met, and who has been a complete failure as a professor, he's a professor at Oklahoma State University now. He's an incredible guy; I only talked to him twice about his thesis. The second time, the only thing I said you have to stop now, you can't afford the typist, you have to graduate. He had this inferiority complex, which is why he never succeeded as a professor, he wouldn't publish because he was afraid it wasn't good enough.

But people who have made more impact are [Tom] Aird who was a principal person at IMSL for many years, he's retired now. Houstis who is a professor here and is now part time in Greece. [Ronald F.] Boisvert is a very accomplished guy; he's the head of Applied Mathematics at the National Institute of Science and Technology. And Scott McFaddin has done well; he's more talented in administration than he is in research. He's a respectable researcher, but he could be a dean someplace, some day. Cal Ribbens is doing very well at Virginia Tech. And then some people have literally disappeared, Huerta is from Chile; he went back to Chile a couple of years after his Ph.D. and I've lost contact with him.

But none of my students have been what I would call world famous people. Larry Symes was from Canada, he went to the University of Saskatchewan, became department head, vice provost, and now he's the advisor to the Province of Saskatchewan's governor on technical matters or something of that type. I'm not exactly sure what his title is, but he's moved out of technical work. But I think students are nice to have. There are some people who try to live through their students. I don't really do that. I think, give them good advice, and make them work hard on their thesis like my professor made me work. I know that many of them felt I was unfriendly in insisting that they capitalized all the words that should be capitalized and a few things like that, but I do it any way.

HAIGH: So it's interesting, in your first interview you had described as wasted, the four months that you spent improving your thesis, but it seems like from your subsequent enthusiasm for writing and –

RICE: Well I won't say I wasted it, because my style of writing mathematics was really pretty dismal, in retrospect, and it was good for me. But that's all I did and from my point of view you're saying gee, I'm spending four months sitting here learning how to write English. I'm not doing any mathematics; I'm just spinning my wheels, so at the time I didn't appreciate so much the training I was getting.

HAIGH: Now through your career have you continued to write code personally?

RICE: I have, I still program some but not very often, but I wrote large amounts of code up into the 90s, I mean really large programs, but I gradually got out of writing code in the 90s.

HAIGH: And would you say that it's something that you particularly enjoyed doing?

RICE: I liked programming. Yes, it's very frustrating, but, you know, there's a real challenge to certain problems. I spent an incredible amount of effort on trying to make a smart program that integrates a function from a to b, a very classical problem. I spent probably two, three years working fairly regularly on a program to do that that. I don't think I wrote any papers about that but I found that a very challenging problem to try to get some sort of intelligence into that program, so it wouldn't do stupid things. Because there's some model you have of what the function is doing, and how it can misbehave and you try to figure out ways to find that out. Now if I write programs, I write in MATLAB, or I use what's called three address code, which is assembly language programming that's good for any machine, so you can also write it in MATLAB format. So when I do things for computer security I write it in three address code because most computer security actually operates in a machine language code on the various PCs, and I'm not interested in learning another assembly language program.

HAIGH: You talked about the relationship of the software packages that you worked on to artificial intelligence. Did you ever feel a relationship between this kind of work on the structuring of projects and components and research that was going on at that time in software engineering?

RICE: I was an observer of software engineering for a long time and I have a much different view of that than most people. It's at an even lower level in the hierarchy than numerical computation in computer science departments. My reaction to that is, "if you think it's so easy, try it." It's actually an extremely difficult subject to understand. I don't claim to have reached much understanding, beyond the fact that I know it's very difficult to understand, and all these people are looking down their noses at it. I'd view it as like their trying to judge Chinese literature and they don't know Chinese, so they say it's just squiggles. It is a very intriguing subject but I have no idea how it's going to turn out.

HAIGH: Did you ever try and generalize any of the lessons from the project and publish them as software engineering?

RICE: I never wrote papers for software engineering journals. I was very unfriendly to a guy who was up for his Ph.D. final exam, and I asked him some questions, because in fact he didn't know what he was talking about. So he ended up not getting a degree and hated me forever for that, but that forced me actually to look into it. That was maybe in the early 70s or something like that. We had a software engineering research center here for a number of years. I was an active observer of it. I never participated in the projects but I've always been interested in that subject and trying to understand why it is so hard to program, why it is so hard to understand programs, why some people are good at it,

and why some people are bad at it, and all of these conundrums that exist about software engineering.

HAIGH: Can you think of any examples where things that you've learned from the software engineering community have fed back into your work on software packages?

RICE: Well I certainly looked at the software engineering community from the point of view of how to design user interfaces and those kinds of things. I don't think I learned a whole lot about that. I looked into the software engineering approach to correctness of codes, testing codes, how many errors per thousand lines and all that kind of thing. That's a problem in numerical software that is around in software in general, this phenomenon that you can never get all the bugs out of a code. It seems counterintuitive but the evidence is almost overwhelming that it's true. I've looked at a number of things in software engineering and the reason I like the problem solving environment is because I've come to the conclusion that the way you make advances in software productivity is to raise the level of languages. FORTRAN is despised universally now, almost. It was the greatest advance that's ever been made in programming languages. You just had to be there at the time it appeared, it was like magic, and nothing that's come along since has been even a tenth, or a hundredth, as advanced as FORTRAN was when it appeared. It was really an incredible impact on the computing world.

HAIGH: So one thing I know that you've been active with in more recent years is the Computing Research Association. And I believe you've also been giving a bit more thought to the bigger picture, you know, defining the discipline, and the policy related issued. So can you talk about how your interest and participation in those kinds of things has developed?

RICE: Well it took me quite a long time to really understand that computer science as the new kid on the block, not just in Purdue University, but in the scientific community itself, was getting the short end of the stick from Federal funding and all of these kinds of things. And part of the problem was that the discipline was disorganized because it was so new it didn't have the long established Nobel Prize winner types that would lead the charge on Washington to get funding. There was very little information about what was going on as a discipline. There was a computer science board, which Sam Conte was very active on, so indirectly I was aware of what they were doing but I was never active on the computer science board. But it evolved in the Computing Research Association as it tried to change from a volunteer organization to an organization that actually had a budget and lobbyists and an executive director and all that kind of thing. And I always thought that was an important thing for computing to do to get ahead in the world. We still have a very long way to go compared to the well established disciplines like physics which live in a whole new world of funding compared to computer science. I mean the kind of thing that I find revolting is that every professor of physics in Purdue University gets their summer salary paid whether or not they have any research grant or whether they've had a research grant in the last fifteen or twenty years, whereas in the CS department: no external support, no summer salary, period. They have four buildings over there for a hundred students and we have one building for a thousand students, and it's this distribution of resources that is very frustrating.

HAIGH: It's been seven years since the interview with Bill Aspray. I know you have a forthcoming article related to the history of the department, in *IEEE Annals of the History of Computing*. Does that bring the story up to date, or is there anything important you think you should say about what's happened since the last interview?

RICE: Well, the allocation of resources became a real issue when I was department head. I became department head because of a crisis with the resources, a lot of people left. For a while the department was getting more resources, nothing on the scale of the Physics Department but still more. And then that tapered off in the early 90s, when we had a new dean and he actually constricted the number of faculty in the department, we lost five people over a period of years. I had positions, people would quit, and he wouldn't let me fill them. It became a lot of contention again between him and me over this kind of thing, and his view of the world was that Computer Science was getting what it deserved which was very little. I found that not to be very satisfactory. So in the history of this new paper which is coming out in the IEEE Annals of the History of Computing, this spring I believe, the history is brought up, it says from 1962 to 2000 in the title, there's very little discussion beyond maybe 1998.

HAIGH: Did the internet boom of the late 90s produce another crisis in student enrollment?

RICE: It did. We had a rebound in enrollment. I mean, they actually went down after the 1983 bloom, although never back to what it was. But it came back up and we had the same kind of thing where graduate courses have 110 students and then that kind of thing, which our dean thought was fine. Never mind that our school has professors who are teaching courses (you almost think I'm kidding, but I checked it all out) with one student and three professors. They take turns dividing up this big chore [Laughter]. The Dean thought that was fine, that was the way the world ought to work, and then the Physics Department had enough space that they converted three thousand square feet to make a museum mostly from the old junk they had in the building, I think. But they didn't have any space they could give to the CS department for labs or student offices or anything. They had probably another ten thousand square feet that was literally vacant, physics wouldn't give it up. Those kinds of things were frustrating.

HAIGH: And that's pretty much how the situation remains today?

RICE: There is some change. We have a new president. We had an administrative culture before which was primarily "don't rock the boat". The new president has in the first couple of years retired or replaced every vice president and most deans. That can't do anything but improve this university enormously. He has really gotten rid of this idea that, well you couldn't get ahead administratively unless you said I'm going to join the clique that's running the university and the main thing is to keep that clique alive, don't change anything, don't create any waves. That's why Physics couldn't give space to CS: it creates a wave, makes a precedent; we don't want to do that.

HAIGH: So you have some optimism there. So returning to the earlier topic, you have mentioned that you've been drawn to involvement with the CRA largely out of

perception that computer science wasn't receiving a fair share of funding, that it didn't have the kind of profile it needed. What kinds of methods did you use to try and address this problem?

RICE: Well, we hired an executive director; we spent money on getting lobbyist type people who work at getting the community through a regular newsletter to raise awareness. It's a very slow process, but progress is being made. The computing budget has improved compared relative to others in the NSF, if not in Purdue, but it's still a very, what should I say, unbalanced situation. Of course the physicists say well we won World War II what more do you want, and there's something to that, but some of these physics projects are incredible subsidies for the physics community. Like the fusion project. That sounds good, but I think in 1950 it was going to be twenty years till that succeeded, and I think that's still the estimated time till it succeeds. It's consuming several billion dollars a year and employing lots of physics people, and that money could be much better spent on computing where actually it's making a huge impact in the world.

HAIGH: And were there any of these initiatives that you were particularly involved with personally, in coming up with or pushing for?

RICE: Well at one time I spent a considerable amount of time at the NSF, talking to program managers trying to provide them with ammunition. Of course the program managers are looking for help. So I spent a lot of energy maybe from the 70s and 80s, and into the 90s, traveling to Washington, being on panels and all that kind of thing, trying to give them ideas and ammunition, but it's a tough life.

HAIGH: Are there any other important groups or panels that you've been involved with?

RICE: I'm trying to think. I am a member of the National Academy but not much happens there. I can't think of any professional organizations that I'm particularly involved with. I'm a member, you know, of Sigma Xi, and some things like that, but I don't do much with them.

HAIGH: So since you've stepped at Editor in Chief of the ACM Transactions you've been on a number of editorial boards. Any of those have a particular impact, other than the normal work of reviewing articles?

RICE: I actually spent some energy on *Computing in Science & Engineering*, which started out as *IEEE Computational Science and Engineering*. For a while I was fairly active in writing some articles for them, one and two page opinion articles and so forth. They've merged with a physics journal and changed their way of doing things and I'm no longer active as an editor of that operation, although I sometimes handle papers.

HAIGH: So what did you hope that this journal would be doing that other journals weren't already doing as well?

RICE: Actually there was a nationwide thrust to start interdisciplinary educational programs in computational science and engineering, and I started one here at Purdue, which was a great success for the students –

HAIGH: Your resume says you were organizer of this program from 1992 to 1994.

RICE: Right. I was the organizer of it, and then somebody else took over as the actual head of it. I spent two years trying to get this thing funded internally but again the structure at Purdue was frozen. We had a program that had one hundred graduate students, probably twenty affiliated faculty, and the most money we ever got out of the administration was $25,000 a year. The physics department with one hundred undergraduates was getting ten million dollars a year to run their operation, but Purdue just didn't have any way for something to change. I had people who were providing money out of their own personal research grants to run this program. The students, a lot of engineering, physics, and so forth students, need to learn about computing and this was a way they could have a Masters or Ph.D. program and a combination of computing in their discipline. It was a very important kind of thing to have happen.

HAIGH: So then the journal was designed to do the same kind of promoting?

RICE: Right. Promote that kind of thing on a national basis.

HAIGH: So if it's now merged with the physics publication, which would represent a shift towards a more specific disciplinary identity?

RICE: Well both of these journals were not financial successes; one was from the American Physical Society, and one from IEEE. They had similar goals. The actual title of the journal I think is the one that the physicists were using, or very close to it, *Computing in Science and Engineering*. So they were very similar in what they were covering, and it made good sense to merge them. I can't say that the Journal has changed that much.

HAIGH: I don't see you on the editorial boards of any SIAM journals. Is there a reason for that?

RICE: I think SIAM doesn't care for me.

HAIGH: Oh.

RICE: Because I took that journal to ACM. I'm confident that there are people in SIAM hierarchy who still remember that.

HAIGH: That's interesting. So then the field remains quite polarized.

RICE: I've heard people quote other people in SIAM saying, "He's an ACM person."

HAIGH: That's interesting. Alright then, let's turn to your current research projects. You've obviously spoken a great deal about ELLPACK and the related projects, and you

spoke about the work in problem solving environments that came out of that. I know that you have a number of other current interests including computer security.

RICE: Right. And currently I'm spending almost all of my time on computer security. A little bit on parallel computing, but my real interest is computer security. And actually I'd spent six months in the Army Security Agency, when I was very young, so I had an interest in security long ago. The Army Security Agency was folded into the National Security Agency when NSA was created, but I never did anything with it until maybe four years ago. [Mikhail J.] Atallah, who is a theoretician, got interested in a problem of what's called secure outsourcing. So you want somebody to solve a problem for you, but you don't want them to know what the problem is that they're solving. That is basically the idea: how do you disguise the problem, so they do all the work? You get what they produce something that you can trivially massage and be the answer to your problem. That's a very nice mathematical problem. He had done a little bit of work with that with a couple of other people, and then he asked me about it and then we got much more involved in it. There's actually a paper that has other authors. Our patent has other authors on it but they didn't do any work, besides Mike and I along those lines. One thing led to another and then he had a student. I'm not exactly sure who had the ideas, but between them they had a bunch of ideas of making software secure, and so we just said, "well let's take these two ideas and put them together and make a company and see what happens."

HAIGH: So that's given rise to a company?

RICE: Right. And so far we've raised eleven million dollars in venture capital funding, and we have real employees and offices in San Francisco and West Lafayette and Washington, cross your fingers, no profits but –

HAIGH: You have a product?

RICE: Yes, it's a software tamper proofing system, the words probably don't mean much to most people, but the idea is to make it so that programs can't be hacked. How would you make a program so nobody can hack it? Well at first glance it doesn't even seem plausible, but what you do is you have to put something in that program that not only solves the problem, but checks whether somebody's trying to hack the program. That's the underlying idea.

HAIGH: And is the mathematics of that at all related to your previous interests?

RICE: Not really. It is more like crypto analysis. It's a rather mathematical thing, but it's not related to scientific software very much.

HAIGH: So this would be related to the same kind of techniques that would be used for something like public key encryption?

RICE: That's the kind of thing, right.

HAIGH: That's very interesting. So suddenly after spending so long in the university environment and not commercializing ELLPACK, you're being thrust into this kind of startup environment. So what's you personal role within the company?

RICE: Well I'm technically a consultant, and I work something close to half time for the company. I don't have any eight-to-five type involvement, but the last six or eight months I've been heavily involved in organizing and creating patents and patent applications, generating ideas to patent this, ways to protect software. This technology that we have is a different approach than anybody else has, so it's important to cover it by patents before other people figure out what's going on.

HAIGH: I see you have two on your resume. And will there be some more?

RICE: Actually, I think they just brought in the new version of my resume which has, I think, five applications now, and we're going to make more.

HAIGH: And you're enjoying this?

RICE: I like it, yes; it's a change of pace. Yes, I'm very much interested in it, very satisfied with it. Cross my fingers I don't spend all this energy and the company goes bust it looks good. For example, one of the persons on the Board of Directors is the head of the National Security Agency, another one is the ex head of DARPA, we have a president whose business it is to start businesses, and he knows how to get these people. He gets a good idea and something going. He's very well connected, has done it a number of times before. I've forgotten the exact number, but I think he started ten companies before he started this one, and I think only two of them flopped, something like that. One company he only was involved with for a year and a half and they sold it to Intel for ninety million dollars, so he pocketed a nice piece of change out of that. That's his business. He's a business major, but he knows computing on a superficial level, not a technical person.

HAIGH: So do you think that if the ownership to ELLPACK had been more clearcut and concentrated, that it could have been successful as a product back in the 80s?

RICE: It could have compete. Some companies are pretty big; MacNeal Schwendler is a hundred, two hundred million dollar a year company I think, that does interfaces for engineering applications. It's not something that I visualized as being easy to accomplish and it doesn't have the attraction of the current thing. Computer security is an area that hasn't received as much support as it should. You may not be aware that the Federal Government actively discouraged research in computer security for twenty or twenty-five years, so there's a vacuum there, and this idea is just a new way of doing things that we're exploiting here which has a lot of potential.

HAIGH: Okay. To wrap up; I'll ask two related questions, so the first one would be of all the things that you've been involved with over your career, what single thing do you think you'd be proudest of?

RICE: Proudest of? Well, I certainly am very pleased with the work I did on nonlinear approximation and developing a theory that explains a lot of the crazy things that people were observing in practice about nonlinear approximation. I'm certainly pleased with ELLPACK, even though it didn't have a big impact in the sense of a product. I think everyone now in this field is aware that you need to be serious about evaluating the effectiveness of numerical computations. That was its main thrust. It worked in the PDE area, but it certainly had an impact there.

There are other things that I'm very pleased with that are not very well known. Let me give you an example. Earlier I'd mentioned that the first mathematical software conference there was a German professor Collatz, I don't know if you've heard of him he's not in computing, he's been dead for a long time. We had a discussion one evening and came up with the conjecture that the work to solve a partial differential equation, numerically, is the same as the work to tabulate a closed form solution to the equation. That seems wildly impossible at first glance, it was so implausible that we never even published the conjecture any place, but it's one of the sources of my interest in how hard it is to solve partial differential equations. And at a conference last summer, I actually could give an outline of a proof that it's correct based on approximation theory in PDEs. Now the people in the audience were not in numerical computation, so I'm not sure what they thought of that thing. It requires you to take a new view to mathematics and what's important in mathematics. It was a heretical kind of approach to things but I think it's actually true. It's still actually not been published anywhere, that conjecture, but deBoor and I proved key results in about 1979 that were seventy-five percent of establishing that conjecture. It's the kind of thing that I'm pleased with and nobody even knows about it.

HAIGH: So the flip side of that: Is there anything that you particularly regret. Perhaps a choice you made, something that you spent too much time on, or that should have followed up on and didn't?

RICE: Well, if I was as smart then as I am now…. I was offered a job at Stanford the same time I was offered a job here. The situation at Stanford was pretty iffy at the time, it was a temporary appointment, but if I'd known the world as well then as I do now, I would have known that having the opportunity to become a professor at Stanford was worth a whole lot more than being a professor at Purdue. But I didn't want to move to a big city and the position there was on soft money, so I decided to come to Purdue. That certainly would have been a very big difference in my life, but I can't complain too much about Purdue. And when you're young you really don't appreciate the impact of the guys at Stanford versus guys at Purdue. I know guys at Stanford that don't do very much at all, and they're always getting grants just because they're professors at Stanford. They don't accomplish anything but second rate work.

HAIGH: Well thank you for taking part.

RICE: Okay, you're welcome.

[End of Tape 4, Side B. End of interview].