An interview with
W.J. CODY

Conducted by Thomas Haigh
On
3 and 4 August, 2004
Glen Ellyn, Illinois

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.

ABSTRACT

W.J. Cody discusses his career as a mathematical software specialist within Argonne National Laboratory, paying particular attention to his creation of the FUNPACK, , (and others). Cody graduated from Elmhurst College in Illinois, before spending a year in Korea as a radio operator with the U.S. Air Force during the Korean War. A year after his return to America he entered the University of Oklahoma, where he obtained a Masters degree in Mathematics, eventually transferring to Northwestern from which he earned a Ph.D. Having previously spend two summer internships working with computers at Los Alamos, he returned to the National Laboratories with a job as an analyst in Argonne's Applied Mathematics division, where he remained until his retirement in 1991. Cody discusses his work and that of his colleagues, including Joe Cook, Larry Wos, Henry Thatcher and Burt Garbow. He pays particular attention to the development of numerical software libraries at Argonne, in which he played a considerable part both as program librarian and as a developer of mathematical software. He pioneered the testing and certification of special and elementary functions, pursuing this interest within ACM SIGNUM and through collaboration with Hirondo Kuki on the SHARE Numerical Analysis project. This interest was an importance influence on Argonne's NATS (National Activity to Test Software) project of the early 1970s, which produced the celebrated EISPACK system as well as Cody's own elementary functions package FUNPACK. He discusses the development, testing, distribution and certification procedures used on this project, and the relationship of Argonne's mathematical software to users and commercial libraries. Cody subsequently produced the special functions package SPECFUN. In the 1980s he worked on ELEFUNCT, an elementary functions package accompanied by a book and by a routine to automatically detect and exploit machine characteristics called MACHAR. Cody played an important role in the IEEE 754 working group that produced the standard for hardware floating point, and from 1981 to 1987 he chaired a committee producing a radix-independent version of the standard. Cody was also a longtime member of the IFIP 2.5 Working Group devoted to mathematical software.

[Start of Tape 1, Side A]

HAIGH: W.J. Cody interviewed by Tom Haigh. This is the beginning of Session One, being held on the third of August 2004, it's being held in W.J. Cody's house in Glen Ellyn, Illinois, outside Chicago. W.J. Cody is also known as Jim. So, Jim thanks you very much for agreeing to take part in this interview.

CODY: My pleasure.

HAIGH: And I wonder if you could begin by talking in general terms about your upbringing and how this may have led to the emergence of an interest in science and mathematics?

CODY: Well, I was raised locally, in Elmhurst, which is about fifteen miles east of Chicago, just a few miles from here. It was the depression years, we didn't have much, and I think science and mathematics were probably the furthest thing from my mind in those days. I can remember, when I finally got into high school, discovering a couple of popular books on mathematics that peaked my interest, "Mathematics and the Imagination," things like that. I took mathematics in high school, I took a college prep course although I wasn't entirely convinced I was ever going to get to college. I really wasn't a very good student in junior high and it wasn't until my junior or senior year in high school that I think I finally figured it out and learned how to study. I did manage to get a scholarship to Elmhurst College. That's probably the only way I ever would have gotten to college. In those days Elmhurst was a fairly small school. I think the total enrollment was about six hundred at the time I went. They didn't have a mathematics major but they had chemistry major. So I majored in chemistry with a minor in mathematics. As it turned out the degree that I took was worded as a double degree, Bachelor of Science in Mathematics and Chemistry, but it was the bare minimum at a college level.

HAIGH: And what kind of institution was Elmhurst College?

CODY: Well it was church affiliated. They had a very strong program in pre The(ology). They had a very strong program in pre med, and other than that it was just a bachelor of arts school, general studies, no graduate degrees offered at all. As far as I can tell, that's still the thrust of the college today, although it's much larger and they do have a lot of evening classes. They service the local community.

HAIGH: And at the time that you decided to focus on chemistry, how did you see your future developing?

CODY: I didn't have the foggiest notion. It was just something to do. You had to declare a major, and that was it, I really enjoyed the mathematics much more. We had to take a certain amount of physics but our physics teacher was a graduate student at Purdue, who came up and taught a couple of hours a week, and he really wasn't much into physics. So I didn't have any physics background. I had fairly decent chemistry background. I fortunately had a very enthusiastic mathematics teacher, staff of one, and I took every math course they offered and had a ball. But it was all pure mathematics there was no numerical analysis involved.

HAIGH: And during these years were you even aware that such a thing as a computer existed?

CODY: They didn't exist in those days. This was, I graduated in 1951, I guess there were a few in the secret laboratories here and there, but no, computers didn't exist.

HAIGH: Well, there were, there were maybe a dozen stored program digital computers in the world –

CODY: Something like that –

HAIGH: The ENIAC and the EDSAC and some commercial prototypes and so on. There was one popular book on the subject called "Giant Brains", and a few of the people I've talked to have mentioned seeing that, but there was not a lot of public exposure to computers. Now are there any ways in which you think this upbringing and early education have shaped the way that you've approached the rest of your career?

CODY: Oh, I'm sure that there are ways; I couldn't point to them. I think it's just a general attitude and an inquisitiveness, desire to get things correct, that's probably the most important influences. A little bit of ingenuity, you've got to think.

HAIGH: And then when you graduated from college, what came next?

CODY: Well the Korean War broke out in my senior year, actually in my junior year I guess. It looked as if I were going to be drafted, so I enlisted in the Illinois Air National Guard and that bought enough time that I could graduate. I was activated in September of 1951, that was just a couple of months after I had graduated. I served for a very short time in this country and then spent a year in Korea. I was with the U.S, Air Force stationed about three miles north of Kimpo Air Force Base and about five miles behind the lines. Fortunately we had a Turkish outfit between us and the Chinese, and the Chinese didn't want anything to do with the Turkish group, so it was a very quiet place, if you can call a war zone quiet. I served a year, came back to this country and got a discharge.

HAIGH: So what work were you personally doing in Korea?

CODY: I was a radio operator. I worked the, I was in the radar group, we controlled the fighter missions out of Kimpo, in fact out of the western half of Korea, that went up towards MIG alley. For a while I operated the direction finder in the radar hut. This was an old Navy piece of equipment that supposedly allowed you to pinpoint the direction of an aircraft transmission. I also monitored the emergency channel to alert the radar controllers if there was an aircraft in trouble. I was crew chief on that for a while. After about five or six months of that, it was six hours on and twelve hours off, and I got kind of loopy, so then I went back and handled the radio shack, got out of the radar operations part of it, and just did the day to day operations of the radio communications.

HAIGH: And did that give you any kind of interest in those areas of technology?

CODY: Well, we had hands on training to be radio operators and I learned Morse code. Part of the training, in this country at least, was to get your first ham license, so I had the novice ham license. I never pursued it any further.

HAIGH: And what came after your discharge?

CODY: I got married almost immediately. We had the GI Bill and I wanted to go back to school. I had been accepted at the University of Michigan before I went into service. But we made several decisions at that point. One, I felt that we really should work to try to get some money to go with the G.I. Bill, and so we agreed that we would work for a year, my wife and I, a year or perhaps two years. Towards the end of the first year we discovered that we really weren't saving that much, we might just as well go on and go to school and have it over with. And then the question was: should I go to the University of Michigan, which was a pretty high powered institution, recognizing that I had come out of what I considered to be a fairly weak undergraduate background, and not having done any mathematics for two or three years, or should I be a little more cautious. And so we wrote around, and I got a graduate assistantship at the University of Oklahoma, which I did not consider to be in the same category mathematically as the University of Michigan by any means, but was probably a spot where I could survive.

HAIGH: Where had you been working between the Army and grad school?

CODY: Oh, I spent a little bit of time with Motorola as a technician in the lab helping them pot filters, electronic filters, electric filters. And I hired on at U.S. Gypsum Company as a chemist in their research labs. That quickly convinced me that I was no longer interested in chemistry, and it made it a lot easier to go back to school.

HAIGH: Was there anything other than the graduate assistantship that attracted you to Oklahoma?

CODY: No. I knew very little about the field of mathematics. As I say I hadn't done any reading, or hadn't been active in the field for several years, it just seemed like it was the best opportunity for me to succeed.

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

CODY: [Laughter] Ah yes, the living conditions were horrible. We had a little building all to ourselves that measured sixteen feet square. There were four rooms in that building, counting the bath. When we measured it off in our apartment up here, it fit very nicely in the middle of the living room. These were prefabs, they had been built to house veterans from the Second World War, and they were still in use. I don't know if they exist today or not, but there were an awful lot of us ex-GIs in those buildings.

HAIGH: Were the buildings associated with the University?

CODY: Yes, it was university housing. We moved up as the years went by, we got in to better housing, and the graduate assistantships paid a little bit more. Jo got a job, so we could afford to upgrade a little. The rest of the housing was very nice but that first year was some experience.

HAIGH: And how did you find Oklahoma?

CODY: A strange place, strange place. They were in the midst of a drought so we had our daily dust storms. It supposedly was a legally dry state, and yet the newspaper every day carried the price lists for the liquor that was available in neighboring states and the list of the Oklahoma citizens who had Federal liquor licenses. So there was the catalog and the list of bootleggers. [Laughter] It was great. I don't drink, but I thought that's real hypocrisy. They had a gubernatorial election while I was there. I don't remember who the governor was, but the lieutenant governor was a fellow named Cowboy Pink Williams and his only claim to fame was that he had been arrested and convicted and served some time for sending obscene postcards about Dwight Eisenhower though the mail during a previous presidential election. The people down there in general were fairly ignorant, I thought. Don't get me wrong, there were a lot of people who were very intelligent, very nice people, but there were also a lot of uneducated people.

HAIGH: And how was the mathematics graduate program?

CODY: Classical mathematics. The department was divided in two, as there were those people who were still doing mathematics from the nineteenth century, classical mathematics, and there were those people who were trying to do modern mathematics: topology, differential geometry, that sort of thing. But it was all pure mathematics, there was no numerical analysis offered anywhere in the department. Graduate assistants taught incoming freshmen. Once you got your masters degree you were allowed to teach more advanced courses, but mostly you taught the remedial courses, and these included courses for kids coming out of one-room schools. So they weren't very high-powered courses at times.

HAIGH: Did you come into contact with a computer during your time there?

CODY: Yes and no. After I got my masters degree, that was the end of the summer of '56, I had a fellow graduate student who had been employed as a summer intern at Los Alamos. I got a job in the same group at Los Alamos that he was in, so I spent the summers of '57 and '58 at Los Alamos. There they had computers: IBM 704s, which were some of the first ones delivered I guess. And when I got back to the University of Oklahoma, after that first summer at Los Alamos, I discovered that there was a JOHNIAC type machine up at the north campus. The main campus of the University of Oklahoma was sort of south of town, and north of town there was an old Naval air base and the electrical engineering department had taken over a portion of that. They had put together this machine in one of the buildings up there. I never actually saw it. I never tried to use it. The second summer after I came back, 1958, as I was leaving Oklahoma to move on, there was an IBM 650, a drum based machine, in one of the rooms in the math building. I'm not quite sure if that was through the electrical engineering department or if it was through the mathematics department. I had no idea at all what they were going to do with it because, as I say, there was nobody on the faculty that I was aware of that knew anything at all about numerical analysis, let alone computing.

HAIGH: Right. So through 1958, when you left the mathematics department there, there was really no computing activity happening within it.

CODY: Not to my knowledge.

HAIGH: So what area did you focus on in your course work and research?

CODY: Real analysis. I had a major professor who was excellent, Casper Goffman. He was a very good teacher and I think most of his students had an enthusiasm for real analysis as a result of his teaching. I had visions of writing a thesis in that area, I don't know about what, but we never got that far.

HAIGH: So you stayed on for a couple of years after your masters' degree?

CODY: Yes, I started work on the Ph.D., I got the course work in, I passed the language exams, that sort of thing. About that time the warfare in the department between the traditionalists and the modernists, if you can call them that, erupted and Goffman ,T.K. Pan, who was the geometer (differential geometry primarily), and John Giever, who was a topologist became frustrated. Joe Foote had some experience in numerical analysis because he was doing, he had contracts with, I think, Moffett Air Force Base, and he was solving PDEs on Frieden desk calculators, having to do with parachute loadings. But anyway, the internal fighting in the department erupted and several of these more progressive professors left. That didn't leave anybody in the department for me to work with, so I worked out a deal where I would go to Northwestern University, and work as an instructor up there and try to get a thesis topic from somebody up there. Giever back at the University of Oklahoma would be the formal sponsor. About six months into my stay at Northwestern the head of the department from Oklahoma came up to a math meeting in Chicago, at the University of Chicago, and informed me that that arrangement was not satisfactory. I would have to return to the University of Oklahoma and write a thesis under one of the older people, which wasn't satisfactory to me. And so we agreed to part ways.

HAIGH: And you have mentioned that you spent the summers of 1957 and 1958 at Los Alamos. How did that come to be?

CODY: Well I enjoyed it very much. I was part of a very small group. There may have been half a dozen physicists. There were two of us that I would consider to be mathematicians, John Thomas and myself, and maybe half a dozen secretaries or people to handle the Frieden desk calculators. The group leader was the guy who pushed the button on the A bomb tests for Los Alamos, that was Hermann Hoerlen. So he was in the field a great deal, either at Bikini or out in Nevada, and the physicists, most of them, were back and forth. My job was to take the data that had been sent in and run the computer programs that processed this data, I have no idea what was in the programs. I was a gofer.

HAIGH: So you were working as a computer operator?

CODY: In a sense. You understand that in those days we didn't have operating systems. You took your deck of cards and you went down and hopefully the individual responsible for the computing room had the computer turned on, and you fed the cards in and waited for the results. Sometimes they came out on computer cards, I think most of the time they did. And the machines, my goodness, we had I think three 704s. If I remember right, there was an 8K and two 16K machines. Those were words not bytes, but even so you had to really be clever about programming, so everything was done in machine language. The first Fortran compiler came through, if I remember right it was my second year there, so that would have been '58, and I can remember playing with that and thinking, "Wow, what a wonderful idea that was." But it was still hands on, and I had top priority on that machine, those test results were important. I could get on that machine any time I wanted as long as it was two AM.

HAIGH: So the computer was like a piece of lab equipment that you could sign up for time on it and then have an allocated shift.

CODY: Yes.

HAIGH: So there wasn't a team of specialized operators who would take the cards from you and put them into the machine?

CODY: No, no. It was hands on. That may have had something to do with security; I have no idea.

HAIGH: I believe the arrangement was relatively common in scientific sites…

CODY: Probably was.

HAIGH: …but pretty much unknown at data processing installations. So when you first saw the computer, and then started playing with it, did you quickly realize that computing was an area that you would want to spend your career in?

CODY: I don't think so, not initially. I was a little concerned that my education was in pure mathematics, and without a Ph.D. there was very little I could do. With a Ph.D., I probably would have wound up at some university somewhere teaching courses. But the experience at Los Alamos showed me that there were places, very interesting places to work, where you had freedom to think, good equipment, and you could learn something. So I think in that sense the experience at Los Alamos was very positive.

HAIGH: Did any contacts that you made there continue into your later career?

CODY: No, unfortunately not. When I decided I was going to have to go back to work after the year up at Northwestern, I contacted Los Alamos to see if there was a position available there. This was the year that the test ban went in, and so they were scrambling to retain the scientists that they had, with their experience, and they were shifting them around internally so there was no external hiring at all. I tried several places. I tried General Electric up in New York, Schenectady, and actually got an offer from them to work in their, I guess they were designing reactors, I'm not quite sure what they were doing. And I got an offer from Argonne, and the tie in with the Atomic Energy Commission, and the National Labs. I accepted the Argonne offer even though it was less money than up at Schenectady.

HAIGH: What were your initial responsibilities at Argonne?

CODY: I've never quite figured that out. They put me in a little office, it was a converted broom closet, there wasn't even a window in it. In fact I had to wait almost a week for them to get it ready for me, and they essentially said find something to work on and read. And so I spent quite a bit of time reading physics, reading mathematics, trying to get acquainted with the equipment they had. I don't think that was a very productive time, for Argonne or for me.

HAIGH: Were you part of a group?

CODY: Oh yes, the applied mathematics division at that time consisted of three levels of people. There were the analysts whose job it was to take problems posed by scientific members of other divisions, and work on them. There were programmers whose job it was to take the designs, if you wish, that the analysts produced and write programs. And then there were the computer operators, the people that were responsible for keeping the machines running. And I was one of the analysts.

HAIGH: But this group was based around the computer center?

CODY: It was based around the computing center.

HAIGH: So did the applied mathematics group have any research program of its own at that point?

CODY: Yes, people were bootlegging their own research. There was no direct funding from Washington for research initially, and so they would try to make enough off of the jobs that they were doing for other divisions so that they could spend some time on the things that were of interest to them. It wasn't until I'd been there three or four years, maybe even more, when we got our first research budget, per se, and what a relief that was. In some ways, because that also meant that you had to yearly justify your existence to the people in Washington.

HAIGH: And how would you describe the general atmosphere at the lab during the late 50s and early 60s?

CODY: Oh, I think the morale was great. There were a lot of interesting things going on computing wise. We had two machines, we had an old analog computer and that group was quite active, and then we had a machine that had been built at Argonne. Argonne built a number of early computers. The Oracle went down to Oak Ridge, and the one we had at the time I got there was one called George. It actually had a floating-point unit in it, home designed. It was a paper tape machine. You had to punch the programs into paper tape. About the time I got there they got a 704. Wow, that was living.

HAIGH: So what kinds of problems were you working on as an analyst?

CODY: Well I teamed up with a physicist, who actually hired into our division. That was Ken Smith, he came from Great Britain, and he was interested in scattering problems, positrons and electrons on all sorts of different atoms. And so I did a lot of the computational design for the things that he was doing. I did a lot of programming; he did expansions in Bessel functions and I had to figure out how to calculate those, that sort of thing. And I never knew what a Bessel function was before I got to Argonne.

HAIGH: Right. So was this the period that you were first getting into applied mathematics?

CODY: Yes, yes.

HAIGH: Was it an uphill struggle?

CODY: Oh boy was it, I'll say. I sometimes despaired that they would ever keep me. Yeah, yeah, it was a chore. I was actually the second professional hire that the division director had made. The previous division director had committed suicide and one of the young physicists was appointed division director, that was Bill Miller. There were a lot of people in the division at that time that I think were a little skeptical that he could pull it off. He hired a programmer, Burt Garbow, first, and I was the first one that he hired into the consultation section. And I've got to say looking back, Bill Miller had a vision, and he pulled some things off that in hindsight I'm amazed that he could do, and the division prospered under him. In fact he made such a reputation there that he went west to Stanford. I think initially he was head of the computer lab at Stanford and then he became a provost, so he moved up the ladder rather quickly.

HAIGH: And you had said that you were the first person hired as an analyst. Were there other analysts there who transferred or were you to begin with the entire group?

CODY: Oh, no, no. I said originally that there was a group of analysts. There probably were eight or ten of us all told. Some of them had been there almost since the founding of the laboratory. We had some interesting characters, that's for sure.

HAIGH: Anyone you want to talk about?

CODY: Well, let's see. Joe Cook, I think he's still down there in one capacity or another. A very independent thinker, a brilliant mathematical physicist, he went his own way, did his own thing. To give you an example, the story I got was that he was a crew member on a bomber in the second world war that got shot down, and so he spent, two years in a prisoner of war camp, something like that. When the war was over he was liberated and he was supposed to be discharged but it was spring and he wanted to go the University of Chicago. Now what was he going to do during the summer? So he got in line with his papers, he got in line and then he just walked, he spent the summer on the beach collecting his pay as a GI, and along towards the end of August he reappeared, stepped back in line, got his discharge. You know that takes a certain… I suppose he figured what were they going to do to him, he'd already spent two years as a prisoner of war, couldn't be any worse if they caught him. And then another was Larry Wos. I'm not sure if he's still down there or not. He was blind. Graduate of the University of Illinois, I think possibly Chicago, but I think he was from Illinois. Logician and brilliant, you can't imagine the things this guy could do, not being able to see at all. Computers? No problem. He had a Braille printer, he could touch type, put his input in, marvelous, marvelous individual.

HAIGH: And who was in charge of the computer group at that time?

CODY: You know I don't think we had a separate head, I think they reported directly to the division director.

HAIGH: The applied mathematics division?

CODY: Yes, yes, Bill Miller.

HAIGH: After you arrived in '59, through the early and mid 60s, was there a change in the kind of work that you were doing?

CODY: Yeah, I got interested in some things. Don't ask me how, I honestly don't know, I rack my brains on what led to what was going on. But I do remember that I was going to work on calculating Schrödinger wave equations. These were of interest to the physicists and it was something that I felt there wasn't a great deal of competition. There weren't too many people doing this. I was really into it and just getting to the point where I felt maybe I could start playing with it and I discovered that a fellow at the University of Chicago, who I knew, already had a program running and had been publishing his results for a year, and so that was it. There was no way I was going to catch up with him. And so I started playing around, and about that time we got a new computer. I'd had some problems with some computations on that 704, and I traced them back to the elementary function library. So I wrote a few replacement routines for that, and about that time we got the CDC 3600, and I ran some tests on the function routines there and they were terrible, and so I started working at replacing those and I guess that's where I kind of got involved in the functions, the approximations, that sort of thing.

HAIGH: And that must have been what gave rise to your 1964 paper in *Communications of the ACM* about double precision square root for the CDC 3600 ("Double precision square root for the CDC-3600," Comm. ACM 7, 1964, pp. 715-718).

CODY: Yes.

HAIGH: So as you started to dig down inside these functions work did you find that that was something that you enjoyed doing?

CODY: Oh, absolutely, absolutely.

HAIGH: What did you like about it?

CODY: Well, it seemed to me that in many cases the programs that had been written were based on classical mathematics, power series expansions, whatever, and it was sort of blind faith that if you programmed this up it would work. It didn't work. And so the question was why didn't it work? One thing led to another, I found. I started looking at methods for generating approximations, things that could replace the power series expansions or continued fractions. I started looking at more detail, what happens when the arguments get a little dicey, when they get large, when they get too small? What are the safeguards built in? What happens if you give a square root routine a negative number for example? As I got into it I realized a lot of it had to do with the design of the arithmetic engine in the computer itself. That what you did on one machine would not work on another machine, you had to know intimately what was going on inside the machine, you had to begin to think like the machine, and the designs of the machine meant you had to think differently as you moved from one machine to another. So it was preciseness, attention to detail, a little bit of ingenuity in trying to anticipate what might go wrong and building some safeguards in. Of course this was not immediate, it was over a number of years that these ideas began to take shape, but yeah, interesting times.

HAIGH: And I see that from 1965 onward you had quite a stream of papers on Chebyshev approximations.

CODY: Yes. I had, in the process of looking at ways to get better elementary functions I had come across a number of algorithms for generating Chebyshev approximations. As a matter of fact, when I first started working in that area I discovered a little belatedly that we had one member of our division who was an expert on that, that was Hans Maehly, who had come over from Princeton. I say I discovered belatedly, because he was only there about six months and died of a heart attack in the parking lot, so I knew him but I didn't know what he was working on, and he didn't know what I was working on.

So I inherited many of his papers, he had done a number of papers on early efforts of approximations, and I read a number of other papers and gradually put together some programs of my own that worked quite well, I think. I even discovered ways of handling particularly dicey situations. I had a couple of very successful computer programs there. The Chebyshev approximations, the ones that were published, were primarily functions of mathematical physics, things that were of interest to scientists in other divisions in the laboratory. We either did not have programs for those or the ones we had were very poor, so this was a contribution every bit as important, I think, as the elementary functions to the computation community at the laboratory.

[Start of Tape 2, Side B]

HAIGH: So you've said that the working in this area was directly related to some of the problems at the lab. Do all these various papers center on one or two main innovations or all they all chipping away on different and unconnected problems?

CODY: I think for the most part, the early ones at least, are unconnected. I attacked targets of opportunity, it was always my feeling that in an area of mathematics the people that got in first got all of the easy stuff done, and made a name for themselves. The people that came along later worked very hard at refining things, and here was an area where nobody was working. So there wasn't a great deal of competition. It was something that even somebody with my meager background could contribute to. So I attacked targets of opportunity. I talked to scientists from other divisions and they'd say, "Gee, you know I need a Dirac integral", well what the heck is a Dirac integral, go look it up and figure out a way to compute it. "I need Bessel functions," "I need an error function," "I need whatever." It wasn't so much that I was providing software for their specific need, although it did fill a need, as it was that, (A) I was getting a publication, which was important, and (B) those same approximations, computational methods, whatever, were more widely useful, so that people at Jet Propulsion Laboratory or wherever would make use of them.

HAIGH: Now why had this area been neglected? Was it because you needed a big computer?

CODY: I think nobody just got into it; I have no idea. You certainly needed a decent computer, you needed decent algorithms to generate the approximations, you needed to be careful in what you were doing. There were a number of other people that had written algorithms similar to what I was using but they never exploited them that I could see. On the 3600, we had Fortran supported single precision and double precision, but if I wanted to generate approximations that were good enough to give me almost full accuracy in double precision then I had to work in a higher precision yet, so I had to write an arithmetic package that would do triple precision.

HAIGH: So that ties in with your developing interest in software and elementary functions?

CODY: Yes.

HAIGH: And I see that a number of these papers were written with H.C. Thatcher, Jr., -

CODY: Yes.

HAIGH: Who is that?

CODY: Henry Thatcher. He was a chemist but he wasn't in the Chemistry Division, he was in the Reactor Engineering Division, and he was sort of their resident mathematician. He knew a lot of the old timers in the computational field, he had spent some time at the Bureau of Standards, he had some time at Wright Patterson Air Force base in Ohio, and he knew a lot of these people. He knew what the problems were, he knew what functions they needed, he was a good one to bounce ideas off of. We coauthored quite a number of papers. He moved on, he went to Notre Dame initially, and when their computing science department folded he went down to the University of Kentucky and then he retired, he was at Santa Cruz. And about three years ago I lost contact with him, I don't know what happened. My attempts to reach him, we communicated several times a year, my attempts to reach him failed, so something happened to him but I don't what.

HAIGH: Now, according to your resume, in 1963 you became group leader in numerical methods, did that reflect any actual change?

CODY: No, I think it was more a recognition that the work on the elementary functions was important. Rather than my having to do all of the programming, they supplied me with some programmers who were assigned strictly to this. At about the same time I took over responsibility for the library, so we checked the routines that were coming into the library being submitted. A lot of them were in areas where we had no expertise, but we could at least verify that the documentation was correct, and that they really did load and run.

HAIGH: So the numerical methods group was you and some programmers?

CODY: Yes. Nancy Clark, Ken Hillstrom, I don't know, you'll see some names on some of those papers. We had some summer interns from time to time and I tried to give them projects where they could wind up with a paper at the end of the summer, and so you see some of their names on the papers as well.

You know they say that a lot of times the great discoveries in mathematics are made about the same time by two or three different individuals, it's just the understanding of the problem. The necessary groundwork is such that suddenly everything gels. One of the big problems was exponentiation, calculating x raised to the y power, where x and y are both real variables, and a fellow at the University of Chicago, Hirondo Kuki, was doing much the same sort of work I was doing but he was working on IBM machines. And he and I collaborated, talked, visited back and forth, and we recognized that there was this problem with the exponentiation function. We both knew what the problem was, where the pitfall was, but we couldn't quite figure how to get around it. I can remember one day sitting up in my office, Nancy Clark was working for me at the time, and she called and said "I want to talk to you, I think I know how to solve the exponentiation problem." I said "I want to talk to you because I think I know how to do it too." And so, it turned out that we had come up with basically the same solution, so I called Kuki, and he said "I know how to solve…." The three of us had come up with essentially the same solution. I haven't followed mathematics since I retired, so I don't know what the situation is now, but I know that at the time I retired the algorithm that we proposed was the one that was being used most of the time.

HAIGH: Does it have a name?

CODY: No, it has no name, I'm afraid not. It's just, it's one of those things where, if you pay attention to the details and know how the computer arithmetic works, you can do it. And that's what we did.

HAIGH: And you took on the role of program librarian in 1965?

CODY: Yeah, that was, it's just keeping track of things, it's a clerical position.

HAIGH: Before that had there been an official program library?

CODY: Yes, but it was not very formal, people submitted a card deck and a write-up and that was it. It was available if anybody wanted it.

HAIGH: And would these routines be coming from people who, from physicists who just solved a particular problem or –

CODY: Anybody, anybody. I think most of them were general-purpose routines rather than being specific to solving the scattering equations and whatever, but, yeah, anybody who came up with a general purpose routine would drop it in the library.

HAIGH: How would you characterize the state of the library when you took it over?

CODY: Not very good. Quality was down, documentation was sometimes good, sometimes bad. There was just the informal repository.

HAIGH: Would it consist physically of a bunch of punched cards in file cabinets?

CODY: Yes, with write-ups.

HAIGH: So, if you wanted to use a subroutine from this thing you would physically take those cards and feed them in along with your own program?

CODY: No, I think you asked for a copy of the deck, and you received a copy of the deck, and you would do whatever you wanted with it.

HAIGH: Had all the resources in the library come from inside Argonne or were some of them exchanged from other places?

CODY: Now that's an interesting question. Most of them, I'd say almost all of them, came from within Argonne. Initially, there was not that much sharing of programs unless… no, it just didn't happen.

HAIGH: So, you think each of the other computer centers and national labs and universities would have had their own unique library?

CODY: Yes, yes.

HAIGH: So, when you took the job over what kind of efforts did you make to improve the library?

CODY: Well, as I recall we insisted that there be a demonstration deck provided so that we could run the program and verify that it actually ran and the results that it printed out corresponded to what the demonstration deck, or the documentation, said they should. That's rather lame, but believe it or not that was a big upgrade, and a lot of people objected to even doing that much.

HAIGH: Were there any standards in terms of documentation?

CODY: The barest outline, it should discuss the following things.

HAIGH: And how widely used do you think the routines in the library were?

CODY: Initially I would say that the matrix routines and things like that were probably pretty widely used, and as a matter of fact, that portion of the library was in decent shape.

Burt Garbow was primarily responsible for the linear algebra. I think as time progressed they became more widely used. As we changed equipment, and had to move the library from one machine to another, it got better.

HAIGH: You had mentioned that there were programming and analysis groups on staff. Were most of the scientific programs written by these groups within the Applied Mathematics Division or were they written by scientists in other parts of the lab?

CODY: Both, both. If scientists in another division wanted assistance with a computation they would hire a programmer from our group. Eventually, as you know, as we moved on and computers became a little bit more widely available, they would hire their own programmers. It got to the point where it was difficult for the mathematics division to maintain a programming staff. Eventually there was a split: there was a research group that consisted of the researchers, and a few programmers stayed with the research group, and we had our own computing equipment. And then there was a computing center that was what you think of now as a computing center. They had programmers for hire and if those programmers couldn't find enough work in other divisions then they were out of a job.

HAIGH: And would that split have taken place in the 1970s?

CODY: I'd say the late 70s.

HAIGH: Do you remember if it was easy to persuade the end users, who were writing their own programs, to use library routines? Was this something they welcomed or was it something they tended not to -

CODY: Well the routines became available much more easily as time progressed, as our technology advanced, and, yeah, they used routines. EISPACK, for example, became *the* standard. Anybody that wrote their own linear algebra programs elsewhere in the laboratory was wasting their time in our opinion, and I think eventually they realized that.

HAIGH: How about other areas? You mentioned that the linear algebra routines were among the best.

CODY: Yes.

HAIGH: What was the state of the library in other areas?

CODY: Sporadic. The special functions, largely through the efforts of the people I headed, had gotten a lot better. You know, it's difficult right now to look back and see very many areas where you can write general-purpose routines. I guess optimization, we had a good group of optimization people starting about 1970, and they wrote some very good codes for optimization. But partial differential equations, things like that, the programs pretty well had to be hand tailored to the application. There may be some plug-in pieces you can prepare but I don't think there are a great many.

HAIGH: Now were you aware at this point of the SHARE library?

CODY: Yes.

HAIGH: I know that was for IBM. Your main computer by this point was a CDC?

CODY: We had a CDC 3600; we eventually got an IBM 360. For a while they coexisted and then the 3600 disappeared, much to my chagrin.

HAIGH: You though it was a better machine?

CODY: Oh, absolutely, no question. The Atomic Energy Commission or Department of Energy, or whatever it was as time progressed, would make these bulk purchases of equipment. There was a quota system, they bought so many CDCs, they bought so many IBMs, and they had to be distributed between the National Labs. Well the people at the weapons labs got the CDC equipment. I don't know how we ever got that 3600 through, but we did, and it was people at the other labs that got the IBM equipment. Frankly the IBM machines, from an arithmetic viewpoint, were inferior. We got involved with SHARE, Kuki and I collaborated on some things there. He'd been bugging SHARE for quite a while. In fact, he was so disgusted with some of the characteristics of the IBM machine that he began to lobby for changes in the design. Largely because of that lobbying effort (there were others that jumped on the bandwagon, but largely because of Kuki) IBM agreed to make a field change, and they did change the equipment. And the engineers that I talked to at IBM said that that was the last time they would ever do something like that, make the field change. But even then the arithmetic design of the machine was flawed; it just wasn't good.

HAIGH: Was it worse than the 704 had been?

CODY: Oh, yes. The 704 was a binary machine. The 360 was a base 16 machine, which meant that… well I don't know if I can explain it in layman's terms, but it did such strange things as, for example, on certain quantities if you multiplied by one you automatically threw away a digit, and that's not very good.

HAIGH: No. Now I know the SHARE library was originally established during the 1950s, so by this point it must have had quite a lot of stuff in it.

CODY: Yes.

HAIGH: Were there any routines there that you felt were of such quality as to be usable at Argonne?

CODY: Well we had the SHARE library available. You understand that by this time I was doing research work and was not doing computations for anybody, so I guess I can't answer that question. I don't know if they were using them or not. I do know that we urged them to use routines from our library as opposed to SHARE. There was also a second library that was available at that time, what the heck was it called, Scientific Subroutine Package –

HAIGH: Yes, IBM SSP.

CODY: Yes. The fellow responsible for that was Ed Battiste, and he was a fellow student at the University of Oklahoma, he entered about the same I did and he moved on earlier than I did, he went on to South Carolina, Duke, someplace down there, and got a Ph.D. in statistics. But he was responsible for that Scientific Subroutine library and the routines in there were again based on classical mathematics. There was no finesse, or very little finesse at least. I can't speak to the statistics; that was his specialty. The advantage of the Scientific Subroutine library was, (A) it was available, (B) if there were no other routines around to do the particular computation you wanted and it was in there, it was convenient. Disadvantage, it wasn't a very good library. Ed left IBM and founded IMSL, well you've talked to Tom Aird so you know much of the history of that venture, and when he left IMSL he moved back to Raleigh Durham area and founded a company called C. Abaci, which is a poor man's IMSL. He tried to approach it a little bit differently, obviously. Ed passed away about two years ago, three years ago, something like that. So, yeah, there were those two libraries, they were not very good. We urged people to make use of the routines in our library if at all possible.

HAIGH: Do you know where in IBM SSP came from?

CODY: No, I do not.

HAIGH: So from the user's point of view it just turned up one day, and then was announced?

CODY: Yeah, yeah.

HAIGH: And can you talk a little bit more about what was wrong with it and if there are any areas of strength you can remember within it or any benefits?

CODY: Well, I tested it against the routines I was writing for the special functions, and mine ran rings around it. It was all written in Fortran and you could look at it, and it was just a straight forward implementation of standard expansions, standard approaches. There may have been some routines from time to time that were good, I don't know, but in general the reputation was poor.

HAIGH: Was that because of inefficiency or problems with accuracy?

CODY: Both, both. And if you got in trouble it didn't tell you. There were no checks and balances in the routines.

HAIGH: So, sometimes it would experience mathematical difficulty and just give you an answer, and you wouldn't know there was a problem?

CODY: Yeah.

HAIGH: So, back to SHARE. Now I know from your resume that you were involved with the SHARE Numerical Analysis project.

CODY: Yes.

HAIGH: On its advisory committee or its board or whatever it had. So can you talk about that project and your role within it?

CODY: You know I don't remember much about that.

HAIGH: I understand that Kuki was the driving force behind its establishment.

CODY: Yes. Yeah, that was basically his baby. I draw a blank. I know I served there but I don't remember what we did.

HAIGH: And around the same time you were also getting involved with the ACM SIGNUM group.

CODY: Yes.

HAIGH: Do you remember how you came to be involved with that?

CODY: Joe Traub founded the group. Here's the first SIGNUM newsletter for example. I'm not going to give that up. I've got a complete collection up until about the time I retired. Joe Traub founded that group. I think it was an awareness on the part of SIAM that subroutine libraries were becoming important, that there were some problems with the software that was out there. I don't think Joe had any clear cut idea as to what was going to happen here, but it clearly was an area that needed somebody to talk about it and see what could be done. That's essentially what we were doing. We sponsored a number of symposia, inviting speakers in to talk on various aspects of software problems. These were usually given in conjunction with SIAM meetings. I think maybe even an ACM meeting or two.

HAIGH: Aside from Traub, do you remember any other people being particularly active in its early days?

CODY: I'd have to look at the newsletter. No, right off hand, Traub was the one.

HAIGH: Now I know at that point he had been involved in a numerical analysis library project at Bell Labs.

CODY: Yes.

HAIGH: Do you remember anything about that project? How people were thinking about it at that time?

CODY: Well, I think there were probably two or three different places that had cropped up with decent libraries. Bell was certainly one of them. The group down at Sandia Corporation had gotten their act together, I think a little late in the game, but they were beginning to work on a library down there. Sandia, Los Alamos.

HAIGH: Who was involved with that project?

CODY: Oh, boy. Bob Huddleston, is a name that comes to mind. Wayne Fullerton, Don Amos, that's three of them at least.

HAIGH: And did they ever produce a package or a library within or attached to it?

CODY: Yes, yes. Don Amos produced a collection of special function routines. Don Amos paid attention to accuracy and tried to write very good programs, a lot of them based on his own algorithms. But they were not transportable, they were designed for CDC equipment and he made no apologies for that.

Wayne Fullerton produced a collection very similar in concept to the Scientific Subroutine Package except it was strictly special functions. Highly portable because by this time we were worried about portability, but it was not terribly accurate, it was not terribly efficient, it didn't have the safeguards built in. But he had some things in that library that were not available anywhere else.

HAIGH: And do you know roughly what years those would have been released?

CODY: I gave you that survey paper, I think it probably is in there. Amos Lib is, Don Amos(?) –

HAIGH: 1977, "Amos Lib, A Special Function Library" Report 771390, Sandia.

CODY: That wasn't the completed project by any means but that was his first, and Fullerton's ought to be in there too. (Note added by WJC: I have found no citable reference for Fullerton's FNLIB. An updated version, SFUN is documented in: *SFUN/LIBRARY User's Manual*, IMSL Inc., Houston, TX 1984.))

HAIGH: Yeah, so those were a little later than the other things we've been talking about.

CODY: Oh yes, yes.

HAIGH: Your resume also says that from 1966 to 1968, within SIGNUM, you were chair of a subroutine certification group. Do you remember anything about what that group was trying to do?

CODY: I think it was inactive, I don't think we did anything. Certainly nothing that stands out in my mind.

HAIGH: Okay. I would imagine from the title it was something parallel with Kuki's attempts with the SHARE library to see which routines were any good and test them.

CODY: I draw a blank.

HAIGH: Now it does seem in this period your interest in testing different packages and trying to do that more rigorously and more systematically was something that was developing.

CODY: Yes.

HAIGH: Can you remember how that interest developed through the 60s and into the 70s?

CODY: Well I'm not quite sure what you're asking. As I was writing my own software, I was trying to demonstrate the quality in comparison to the software from other libraries. I think it was just a natural outgrowth of the work I was doing.

HAIGH: Sure. Well your second publication was on the CDC functions, I think square root. I think it's your paper in Rice's first mathematical software volume.

CODY: Sort of a survey of the elementary functions, yes.

HAIGH: Yes. And you had a number of papers along the way testing particular functions.

CODY: Well that was an outgrowth of the work we did at Argonne trying to certify the libraries that were presented to us by the computer manufacturers. We took a very careful look at the 704 library, found it wanting. We took a very careful look at the CDC 3600 library, found it wanting. We had developed, by the time the 3600, the 360 came along, we had developed a routine, a pattern for how we would go about looking at these elementary function libraries. The idea was to pinpoint the weaknesses and replace the routines that were weak with ones that we had written. We felt that the reports that we produced were of wider interest than just the local audience at Argonne, and so people using a 360 wanted to know what the library was like.

HAIGH: And can you think of any cases where changes were made to those libraries in general as a result of these testings?

CODY: I have no idea at all what the manufacturers did. Once we made the changes ourselves we were no longer interested in the manufacturer-supplied library. We now had in place a combination of their library and ours, usually it was mostly ours. We had certified it, we knew what it was, how it behaved, why bother looking at their latest effort.

HAIGH: So once you got it working you'd just stick with it, even if they subsequently supplied some updates or patches?

CODY: Yes. I think in a few cases they actually took what we supplied, because what we did was in the public domain.

HAIGH: So were the improvements and corrections going to the other National Labs and places like that?

CODY: Oh yes, available to whoever wanted them.

HAIGH: Some of the interviewees actually have mentioned something called the Argonne Code Center which was distributing code.

CODY: Yes. Let's see if I can explain that. That was in place when I joined the laboratory and it was still in place when I left. The Argonne Code Center was a repository for nuclear codes, codes that had been written to perform nuclear computations, design of reactors, whatever.

HAIGH: And code in this context basically means program.

CODY: Program, the complete program. Margaret Butler was the one who founded and ran that organization, it was attached to the Applied Mathematics Division but it was a separate entity, received funding directly from Washington, it was its own little domain. It had nothing to do with the codes that we're talking about. If Los Alamos had a non-classified program that did some sort of a computation that might be of use in general in the nuclear community it went to the Argonne code center. They had very strict rules on testing and acceptance and distribution, they couldn't distribute many of these things overseas for example because of classification problems, that sort of thing.

HAIGH: So that group was largely separate from the people developing mathematical routines.

CODY: That was completely separate, yes.

HAIGH: That's good to know. I know some of the other people I spoke to weren't quite sure where it had come from.

CODY: It was ancient.

HAIGH: Now how about the other groups and workshops that you were involved with. My impression is that a community started to form around mathematical software at the end of the 60s and start of the 70s, particularly with John Rice's series of conferences.

CODY: Yes, I think he brought some focus to it that, some cohesion if you wish, that had been lacking before.

HAIGH: So can you describe what was happening at that time and your personal involvement?

CODY: Well I think life was just going on as normal, that was about the time that Kuki and Clark and I came up with this algorithm for the exponential function, the exponentiation. John asked quite a number of speakers. I think you've seen the proceedings. Quite a list of speakers, trying to get a survey really of what the field was like at that time, what conditions were, what was lacking. I think he did a very good job, that particularly meeting was an exciting one, I think that's one of the best I've ever attended.

HAIGH: Why?

CODY: There were people there that I knew about perhaps but had never met. It was an opportunity to talk with people, to learn what their perceived problems were, what their perceived solutions were, it was an opportunity to spread the word about some of the things we were doing, Kuki, it was a very good meeting, very productive.

HAIGH: Did you form any relationships at the conference that proved significant later on?

CODY: Oh, I knew a lot of the people that I would work with or communicate with. I don't remember anybody in particular that I first met there that….

HAIGH: Did the conference change anything about how you thought of your own research interests?

CODY: No. I think it changed how the world, how the rest of the mathematical community, thought about what we were doing.

HAIGH: Looking at your resume, I see that up until about this time, 1971, you had really only one publication that was primarily about software, the CDC square root that we mentioned. I see lots about the Chebyshev approximations. Then it seems that there was a shift around about this time, and that following that more and more of your publications were about software.

CODY: Yes, some of them were philosophical, as you've discovered, explaining what we were doing. Not the technical details of how it was done necessarily but explaining why it was necessary and what we accomplished or hoped to accomplish, why it should be done, why this was a good thing. I think we were trying to gain support. There's one here that talks about "Why the Fuss…" ("Mathematical software - why the fuss?," by invitation, SIAM 1976 National Meeting, Chicago, Ill., June 18, 1976) and as I recall was given at a SIAM meeting in Chicago. There were people in that audience who had never heard anything about this sort of thing, it was a complete revelation to them. After I retired I was a keynote speaker at a computer arithmetic conference over in Grenoble in 1991, and I outlined there the problems that we were having in the early days with computer arithmetic, how we had to turn hand springs to do the simplest computations. The people sitting in that audience were mostly engineers responsible for designing computer arithmetic units (a few mathematicians) and this was a complete revelation to them too; they had never heard this stuff before. So I think that's what we were trying to do, during this period, was alert people to problems. This discussion of software for the elementary functions was a warning to the people at that conference, in Purdue, that by golly if you're relying on those manufacturer provided libraries you're in trouble. They're this bad, they really are this bad, in picking out examples.

HAIGH: Do you think that was a message that they took notice of?

CODY: I think that message had an impact. I think a lot of the people there had never thought about these things before. John Rice had, obviously.

HAIGH: So did you have a sense that mathematical software was becoming a more respectable area to do research and publish about?

CODY: Yes, yes. That software conference added legitimacy to the effort, if it needed it, national legitimacy. And of course the *Transactions on Mathematical Software* were a direct outcome of that.

HAIGH: Yes. Now within the lab did that make mathematical software more of a respectable research area?

CODY: I don't think so, I don't think so. We had already gained a great deal of respect and support in the laboratory, the lab directors, the funding agencies, funding supervisor in Washington, they were all on our side, we'd convinced them.

[Start of Tape 2, Side A]

HAIGH: So were there any other people who you think played a prominent or important part in the mathematical software community during the late 60s and the 70s?

CODY: George Forsythe, who headed up the computer lab at Stanford. And Vel Kahan. Vel was one of those individuals who had his finger in everything, I don't think he ever wrote mathematical software, per se, but he certainly had opinions. Both Hirondo Kuki and I batted ideas off of him from time to time. He was quite a valuable source of information. Very volatile, opinionated.

HAIGH: Did he participate formally in groups such as SIGNUM or SHARE?

CODY: Not to my knowledge, I don't think so. He certainly attended many of the meetings. He attended all of the SIAM meetings, he attended all the important meetings, but I don't think. He was not a person to get involved in committees or anything, that's why I was so surprised when later on he became involved in that IEEE computer arithmetic effort.

HAIGH: We'll talk about that later. Now you've discussed a little the impact of Rice's series of mathematical software meetings. Are there any other important meetings you recall from the 1970s? Perhaps the meeting on the portability of numerical software, which I know you contributed to.

CODY: If it's the same one I'm thinking of, it's the one that Wayne Cowell organized, was held over here at Oak Brook. Yeah, that was a good forum. It was not as open as the one that Rice hosted but it was a good forum. There were important people there, we got an opportunity to bat ideas off one another. Brian Ford was there, Ed Battiste from IMSL, we ironed out a few things there.

HAIGH: So, do you think that meeting had a significant impact on the direction that portability efforts took?

CODY: Well, I think the portability became more important after that. Now did it directly influence that or not, I don't know. Both IMSL and NAG were rather closed mouth about what they were thinking and their approaches on things so it's hard to say what was going on there ahead of time. I think from a recent conversation I had with Brian Ford, as he summarized it at that time, he was interested in portability at any cost, and I was interested in accuracy at any cost. He says as it turns out we were both right.

HAIGH: Any other important meetings in the 70s?

CODY: Just the ones down at Purdue, the ones that John Rice hosted. Yeah, I think those were the most important, and that first one was by far the most critical one.

HAIGH: Now a moment ago you mentioned the *ACM Transactions on Mathematical Software*.

CODY: Yes.

HAIGH: Were you involved with that?

CODY: Yes. John Rice became editor in chief. He proposed the journal I think to ACM, he became editor in chief, and he gathered together an editorial panel, associate and assistant editors, whatever we were called. I served on that panel I think for the first ten years of the magazine. It was the usual responsibilities, you received manuscripts from Rice and found referees for them. In the case of this journal, many of the manuscripts involved computer programs of one sort or another, mathematical software. So we tried to find somebody who would test it and banter back and forth about the quality of the software as well as the importance, as well as the text material.

HAIGH: And that kind of testing would be the same process that you had been doing at Argonne to see if routines were of sufficient quality to be added to the collection?

CODY: Probably a little more vigorous. You know it varied with the referee you found. Some of them really dug into things and others sort of gave it a wave, but, yeah, we tried to make sure that the software was worthy of publication.

HAIGH: And how similar would you say that is to the peer review process that a scientific article would go through?

CODY: Very similar, very similar. Editorially it was almost the same. You found two or three referees and tried to correlate what they said and then deal with the author.

HAIGH: How was it that you think *Transactions on Mathematical Software* could do that, had not adequately been done by existing outlets, such as *Communications of the ACM*?

CODY: Well I think it largely took over the computational software from the Communications, there was very little after that went to the Communications. It certainly pinpointed that this was mathematical software as opposed to algorithms for calculating the date of Easter, or whatever. Another thing that happened was that we accepted algorithms written in languages other than Algol. Most of the early stuff was very restrictive. I think I may have been the first one to publish an algorithm in Fortran in that journal. Kuki and I published some algorithms written in Fortran. It opened the gates.

HAIGH: And you were also involved in the IFIP 2.5 Working Group.

CODY: Yes.

HAIGH: Were you one of the original members?

CODY: No, that was a problem. This was the, I want to say that Jim Pool was the one that got that going, but it may have been Brian Ford, between them anyway. IFIP demanded that there be representation from a variety of countries, variety of organizations within a country. We already had Jim Poole as one of the founders, Brian Smith, was on the, one of the original members. Bo Einarsson, I think. Hans Stetter, there were a variety, but already there were two people there from Argonne. They agreed that I should be there but IFIP would object. So, I understood, that's fine. I did join them later.

HAIGH: So you had to wait for a while?

CODY: I had to wait.

HAIGH: Did one of the other Argonne people rotate out and –

CODY: I don't remember exactly how that happened, I know that Pool moved on and that may have been the opening, it may be that they just decided that it was time regardless, I don't remember.

HAIGH: So when you did join, what year was that, and what kind of work was the group undertaking at that point?

CODY: Well, what year was it? I have to look at my vita. 1978. The group at that time and I believe even since then, this is a general list of activities for the group: they held meetings in which there is a session among the members and they discuss the every day work affairs of the committee, that is, where they will meet next, what prospective members should be invited, they vote on the people that have been invited and are willing to serve, they organize topical meetings, every three years or so they have a meeting in which the public is invited and they have a list of invited speakers and people can also submit papers if they want. They encouraged participation in activities, for example I was in a sense the liaison between WG 2.5 and the floating point arithmetic efforts. Not as WG 2.5, but they encouraged their members to get involved in the ADA standardization effort. We had one member, Brian Smith, who was involved in the Fortran standards effort, so there were reports and discussions of the activities of these various groups and suggestions for what the representative might take back to that group. Very good, very good forum.

HAIGH: Did the group's international composition lead to making any new kinds of relationships?

CODY: Oh, I met people there that I never would have met otherwise, yes, absolutely.

HAIGH: And did that lead to tangible things in terms of collaboration or spread of ideas?

CODY: Spread of ideas certainly. Some very strong arguments in a few cases but, yeah, it was a good forum for exchanging ideas.

HAIGH: Were there any areas where you think the group was particularly successful in having an impact?

CODY: Well I think most of their symposia have been successful. They offered resolutions of support for things like the IEEE and the Fortran standards. I think they had a voice in the international community in this area, and they're known.

HAIGH: Now you've mentioned your role as informal liaison with the floating point standardization efforts. Was there anything else that you were particularly personally involved with?

CODY: I helped out with the ADA effort. They were interested in the elementary functions in ADA, and there were two of us I think, maybe three, that served on that standards committee. No, I think those probably were the areas where I was most active. I'm not a good organizer of meetings, so I stayed out of the symposia end of it.

HAIGH: And are there any other groups, activities, developments that you think were important in the emergence of the numerical and mathematical software community through the 70s and early 80s?

CODY: Well, there were a number of speaker programs that were sponsored by SIAM or the ACM. Some of us had an opportunity to make the tour to local universities, sometimes not so local universities, and spread the word on what we were doing, what was important in our field.

HAIGH: So, is there anything you particularly remember from that?

CODY: Well I can remember visiting a number of small schools such as Ripon in upper Wisconsin, and so forth. I found it stimulating. It was interesting to talk to students and to faculty alike, many of the faculty members, especially a small academic community, are unaware of what's going on in the outside world. They may know their specialty but they don't know what's, you know, what's really going on in computing or something, and so I found that the opportunity to sit down, to give a talk and then sit down and talk one on one with people was stimulating. I always enjoyed teaching, and that's the closest I could come to it I think.

HAIGH: So did you every regret not having got your Ph.D. and taken an academic direction?

CODY: That's probably the best thing that ever happened to me. If I had gotten a Ph.D. I'd have been a staid old traditional mathematician teaching somewhere. Man, look at all the excitement I had. I may have misjudged things but I think I made a contribution, and that's a very satisfying feeling. I'm sure the field has grown, I'm sure the things that I did are now historical rather than frontline, but that's the way things progress and, wow, what a ride.

HAIGH: W.J. Cody interviewed by Tom Haigh, this is the beginning of Session 2. It's now the fourth of August 2004, and as with the first session this interview is taking place in Glen Ellyn, Illinois.

So, in the first part of the interview we covered your early career and your general involvement with the mathematical software community. What we didn't talk about at all were the Argonne series of mathematical software PACKs and the projects that gave rise to them. So if we can now turn to that, I wonder if you could talk about the origins of the mathematical software libraries that were issued to the public from Argonne.

CODY: You're speaking of EISPACK and FUNPACK?

HAIGH: Yes.

CODY: Okay. Well, this was the NATS project. Initially NATS stood for National Science Foundation, Argonne, Texas, Stanford, which were the institutions involved. It also stood for the National Activity to Test Software, which is the public title that we gave this. You understand that Argonne had a long history in linear algebra. For years we had had visits from Jim Wilkinson during the summer, Garrett Birkhoff from Harvard had been a frequent visitor, Wallace Givens was the division director, so we had strong history in linear algebra. And when we began to think about the possibility of a coordinated effort to produce software packages it was natural, I think, to concentrate on linear algebra. Not only the local strength at Argonne, but the field had matured to the point where the algorithms were well understood. There was not so much a question of development of new methods as there was the question of proper packaging, testing, and distribution, and that's what we wanted to concentrate on. It would not do, however, to concentrate on only one package. We wanted to diversify a little bit into an area where we felt we had strength, but, more importantly, where the problems were a little bit different, and that's where the special functions came in, that was my strong point.

The linear algebra we could envision as being a set of routines that could be moved among machines with very little modification. There might have to be some adaptations because some machines worked naturally in single precision and some in double precision, but there were no inherent problems with word length of machines or that sort of thing. Whereas, with the special functions, if you were going to have accuracy the programs had to be highly tuned to the individual machines. So we selected three types of mainframe computers, CDC, Univac and IBM. Each of those machines was represented at at least one of the installations that was involved. We set out to produce highly accurate, not transportable, these thing would not move from one machine line to another, but highly accurate machine dependant programs for special functions. I helped with the EISPACK package, I was just a slave labor on that one, but the FUNPACK was essentially mine.

The problems, as I way, were different. For one thing, we wanted to produce test software, a way of checking that the implementation on a particular machine was accurate. So I had to devise a way of writing data tapes on the equipment we had available at Argonne that could be read on CDC machines, IBM machines, and Univac equipment. The word lengths were different, the representation of data in the native format on the machine was different. This was a very difficult problem to solve, but we did it. So, for example, when we sent a program up to the University of Wisconsin to be tested on the Univac equipment up there, we sent the program, we sent a driver program, and we sent a data tape that consisted of a binary representation of arguments, machine arguments, and correctly rounded function values corresponding to that machine argument. The people at Wisconsin then would run the tests, and send us back a summary of the test results, and from that summary of test results we tried to debug the program. And it took a little bit of ingenuity, but I think we were successful.

For example, one of the test results that we got back was rather mysterious. It simply indicated large errors in an area where we expected to be almost exact. In going over the code the only plausible explanation I could come up with was that the Fortran compiler on that particular machine ignored the grouping of arguments by parenthesis, so that it simply threw away parenthesis and algebraically combined what was in the expression coming, up with a much simpler representation. Of course, that was the wrong thing to do. It not only violated the Fortran standard but it screwed up the accuracy of the program. We confided our findings to the people at Wisconsin, they did a little bit of testing, and lo and behold, that's what the compiler did. And I think that's rather impressive to be able to debug a compiler that we've never seen from a distance based on the test that we had performed. I later met an individual at a meeting, an IEEE sponsored meeting down in New Orleans, the day after Mardi Gras I remember (for some reason the math people and a lot of professional people in general tried to get the cheapest accommodations they could and that seemed to be in places like Atlantic City in the middle of winter, New Orleans the day after Mardi Gras when most of the chefs were laid out recovering from all the revelry, no restaurants open). At any rate, when I approached this individual and talked with him about his compiler he was very proud of the fact that his "optimizing" compiler was able to do this, and I told him that, (A) it was a violation of the Fortran standards, and (B) it certainly screwed up the work that we were trying to do. I don't think that impressed him. I doubt that he changed things.

But that meant that we had to go back to the drawing board and break up the particular Fortran expression into two expressions that were separated by enough intervening computations to foil what the compiler writer had done. The result was a very limited set of functions, maybe a dozen, but a very limited set of functions that were highly optimized for individual machines, individual architectures. We were successful, I think, in the sense that installations with those particular machines used that library. It also convinced me that it was an awful lot of work to do to gain that limited success. That there must be a better way, a compromise, so that we could make available, perhaps a larger collection of programs, but over a wider variety of machines. Clearly we would have to sacrifice that last bit of accuracy to get the portability but I hoped that it was not necessary to sacrifice the overall robustness of the programs, and that's what led to the SPECFUN effort.

HAIGH: So, before you move onto that I have some follow-up questions.

CODY: Alright.

HAIGH: So you had mentioned that this was part of the NATS project.

CODY: Yes.

HAIGH: Now who came up with the idea for that project?

CODY: Oh, boy, that's a good question. Wayne Cowell, I think. If I remember correctly, Wayne Cowell was the primary mover. This occurred at a point in time where funding was becoming a problem and we had begun to look for other sources of funding for the research work going on at the laboratory. The National Science Foundation seemed like the logical place to go, and there was a lot of discussion internal to the laboratory, internal to the division, as to whether or not it was proper to accept funding from another Government agency for work that was being done for, I don't remember what we were at that time, AEC, DOE, something in between. The compromise was to include people from Stanford, University of Texas and thus make it a broader project, one involving institutions that the National Science Foundation traditionally did fund. And that's where Cleve Moler came in, he was at Stanford, and Ikebe at the University of Texas. Both had been at Argonne either as visitors or as summer interns, so we were familiar with both of them.

HAIGH: And at that point what was your actual job title and role?

CODY: I don't know what the official job title was….

HAIGH: Actually, I can probably see that on your vita… at that point you would have been an associate mathematician.

CODY: Probably.

HAIGH: But I think you had implied earlier that you had switched from your original role of analysis and support of users into more of a research position.

CODY: Yes, well that occurred throughout our section I think. The division had split, or was about to split, into a pure research division with independent research funds from Washington, and the second half of the division became a support group. Supporting of other divisions within the laboratory as well as supporting us, but they ran the computing center, they provided programmers and that sort of thing.

HAIGH: Now was FUNPACK an explicit part of the original proposal to the NSF?

CODY: Yes, yes.

HAIGH: So the proposal was for the two packages FUNPACK and EISPACK?

CODY: Yes. And as I say, the reason for the inclusion of FUNPACK was that the problems were entirely different from those posed by the linear algebra.

HAIGH: Now talking to some of the EISPACK people I got the impression that the testing of software, which was the official justification for the project and the grant, they viewed as in some ways as more of an excuse to get the funding.

CODY: I think that may have been true.

HAIGH: You, yourself, had been involved in testing and certifying software for some time. So, from your point of view was research into testing methods something more substantial, or just an excuse to get the money?

CODY: You know I never really thought much about it. I think Wayne Cowell handled the political end. I saw this as an opportunity to do something that I wanted to try, but would find it difficult to do with just funding from the laboratory. So, to me it was an opportunity to expand, to bring more people into the act. I realized that the driving force behind the NATS project was the linear algebra, and rightfully so. I was sort of an add-on, it was convenient for them to have me, and boy it was sure convenient to have them.

HAIGH: But do you think that what you had already been doing on testing and certifying software played a part in inspiring the broader project?

CODY: Oh, yes, absolutely.

HAIGH: And you had been the main person at Argonne involved in this effort to certify functions?

CODY: I think I was the first one, I think it was catching. There were other people that were becoming interested, certainly not to the point that I was. We had demonstrated an expertise that made the proposal to the National Science Foundation a valid proposal, and that's important.

HAIGH: And how do you think, how significant do you think Jim Pool's role as acting director and then as assistant director was?

CODY: Very influential. We had been very fortunate at the laboratory in general, with having division directors that, I don't quite know how to describe them, ambitious certainly, but visionary. They were willing to take, take chances, they were willing to approach Washington with new ideas and try to sell them and they were fairly successful at that I think. There were glitches, there were times when Washington simply came back and said, "look our budget has been cut, your budget is cut." Those periods of time were traumatic at the laboratory, but through it all the software work managed to survive and to thrive, and I think it was largely because the division directors, starting with Bill Miller and going right on, had the foresight to champion this cause.

HAIGH: And you had mentioned that you did some grunt work for the EISPACK project. What did that involve?

CODY: Well, let's see, if I remember correctly there was a crisis that came up. I can't exactly remember what the problem was, but we had to reproduce a large quantity of source code. I had listings, Jim Boyle was trying to push his, what became the TAMPR program, so the work could be done automatically and I simply spent some time at the keyboard and keyed in a lot of this stuff again. It took several weeks but the source code that I keyed in was available before the stuff that came out of the TAMPR program. So, it was that sort of thing. When they needed an extra hand, somebody that knew what was going on and was reliable enough to get the job done, why I stepped in.

HAIGH: And by this point were you doing the developments on the time-sharing system?

CODY: You know I don't remember much about the time-sharing system. I don't think it was, boy I don't know how that fit in.

HAIGH: Okay. So moving now more directly to the FUNPACK project itself. I think you had implied that you would have liked, that you had had in mind to do something like this anyway but that it was only the arrival of the NATS project that gave you an opportunity to actually execute it.

CODY: Well not only execute it but it helped formulate what would have to be done. I was beginning to think along the lines of providing good software for other machines, but this was the opportunity to really dig in and get some support for doing that. And frankly, without the help of the people at various laboratories and university computing centers, that were involved in the NATS project, I probably wouldn't have learned enough about the arithmetic on the other machines to be able to do it. So, the interaction was extremely important.

HAIGH: Now in your 1975 article on FUNPACK in *Transactions on Mathematical Software*, you wrote that the original impulse for FUNPACK called for a Fortran translation of part of the package of subroutines available at Argonne National

Laboratory on the IBM 360 for use on other machines. ("The FUNPACK package of special function subroutines," TOMS 1, 1975, pp. 13-25.) So were the original versions of these routines already in Fortran?

CODY: Yes, yes it was. It was not the final version that went out, we did some things, but yes, the groundwork was there. It's the same with the linear algebra of course, they had existing programs that were being polished and revised with transportability in mind.

HAIGH: So, obviously the CDC and Univac versions had to be extensively reworked –

CODY: They had to be done from scratch, essentially.

HAIGH: How much work was done on the IBM version, compared with what you'd already got running at Argonne?

CODY: Well I think they were modified, I won't say extensively, but there were certainly significant modifications to incorporate the lessons we learned about making them reliable. The accuracy I think was largely there but we, the NATS project taught us some things, and we incorporated those lessons in our own library.

HAIGH: So by the NATS project in that context you mean EISPACK?

CODY: I mean the FUNPACK part of it.

HAIGH: Oh, I see. So the internal Argonne library was further developed as a result of the FUNPACK experience.

CODY: Yes.

HAIGH: Now were all the functions that finally made it into FUNPACK based on software that you already had working at Argonne?

CODY: I think most of them were, I won't swear that all of them were.

HAIGH: And tell me if any of them came from contributors outside the lab or primarily developed by people other than yourself.

CODY: No, I think most of these, if not all of them, were based on work that I had done. I'd have to look at the literature citations, but maybe the $Y\nu(x)$ function involved some work from others, elliptical integrals certainly were all mine, exponential integrals, Dawson's integral, the sine function. $Y\nu(x)$ is the only one that I can think of that possibly may have had some outside influence.

HAIGH: Now one of the things that you talk about in the articles you've written on the FUNPACK project is this idea of robustness as a desirable characteristic. Can you define what you meant by robustness?

CODY: Well the definition that we came up with I think varied from time to time. The idea was that when you ran a program it either gave you the correct answer, or something very close to the correct answer, or it told you that something was wrong. So it's reliability under not only normal use, but if somebody happens to give it a bad argument or misuses it, accidentally or deliberately, the routine does something beyond just returning an answer and not warning you that something is wrong. That's a broad definition.

HAIGH: And was this a concept that was well understood prior to the NATS project?

CODY: No, no. The term itself was coined in the NATS project.

HAIGH: And do you think that's a concept that was influential on later projects?

CODY: I would hope so. There was always a question with later "PACKs" produced elsewhere as to whether or not they truly understood what the word PACK meant to us. And robustness was one of the concerns.

HAIGH: You also talk about the division of the software into something that you call "packets," which as I understand it you came up with because compilers had some kind of problem with having a subroutine you could call in several ways. Is that the issue that you alluded to with the compiler incorrectly optimizing and messing things up?

CODY: Oh, boy. I do remember the term. Let's see if I can describe this properly. Sometimes you get a family of related functions for which there is a common computation, a core computation, and you derive the individual functions of the variations on it by very minor modifications. The error function for example, Dawson's integral, complimentary error function, that sort of thing. And so it seems natural to provide a computational core program, with surrounding function programs that rely on that core program to do what it's supposed to do. Then the individual function programs do what they have to do to convert that core computation into whatever it is they're doing. That package, of sometimes two, sometimes three or more, function programs with a computational core, is what I call the packet.

HAIGH: So it's really a way to work around the limitations of Fortran when it came to modularizing things?

CODY: Yeah, it was an attempt to modularize, exactly.

[Start of Tape 2, Side B]

HAIGH: Now as I understand it another thing that was distinctive about this project was the attempt to provide these self-contained tests, I think you called them test drivers.

CODY: Yes.

HAIGH: So that the users would just be able to feed in some tapes, compile the thing, feed this data into it and you would get useful feedback.

CODY: Yes.

HAIGH: Now was this motivated by disappointing experiences with more laborious kinds of testing?

CODY: The tests were a beginning. It was an attempt to provide the user with an indication that the program was working properly, and to give him an indication of the quality of the program, the accuracy. The resiliency, there were some tests there usually that involved improper arguments, that sort of thing. It was a recognition that most librarians when they receive software from outside, have no idea at all where to begin to determine whether this program is working. Typically when they began to look at function programs they would run in a few arguments and compare the results against tables. Well, the arguments are not represented exactly. You've got a conversion problem, the conversion routine in the program that reads the decimal representation and converts it into the internal representation. You don't have the argument you thought you had. There's a computation done, the result is put back out, you go through the conversion routine again, there's another error. You look at the table, the table isn't the same word length as the machine, you have another error. If it looks bad in some of the low decimal places what is the source of error? Is it the conversion of the input argument, is it the conversion of the output result, is it the comparison against the table? Who knows? What we were providing was a binary tape that contained exact arguments, correctly rounded results, you looked at them and gathered statistics and you say "okay on this set of test arguments the results are such and such, and that looks pretty good." It's far superior to what the librarian was likely to do himself but, as I say, it was not all-inclusive.

HAIGH: Was this idea of sending out these relatively extensive largely automatic test routines something that, as far as you are aware, had not been done on any earlier project?

CODY: Yes.

HAIGH: Do you know if that was influential?

CODY: I would like to think it was. I think, you know when programs were initially accepted for the library collection of *ACM Transactions*, you had to provide a driver. But those drivers were rudimentary; they didn't really do anything they just made sure that the program compiled.

HAIGH: So, this term "driver" was around before the project to describe, basically to describe a simple test routine that would call the functions and produce an output?

CODY: Absolutely. I would like to think that after the NATS project that the drivers became a little bit more sophisticated. At the same time as we moved on, when we get to talking about SPECFUN, it's clear that that type of driver is time consuming to produce. It's an extra mag tape to distribute so there's an extra problem there. It's really not the best way to do things, but it was the simple thing to do and it allowed us to pay attention to the details, the last bit accuracy, if you wish. But in the overall picture it is probably is not the best things to do, and so that's why we began to look for other ways to test function routines.

HAIGH: Yeah. In fact in your article in Wayne Cowell's book, *Sources and Development of Mathematical Software*, there's a nice statement that with the EISPACK project the group originally intended to supply field test sites with only source text and preliminary documentation and that they would do lots of testing on their own. And then you say, "Early experience with EISPACK however dashed these hopes, the volume of material to be tested overwhelmed most sites. Personnel assigned to testing often lacked numerical training necessary to produce independent tests or were busy with other responsibilities. In a few cases field testing was carried out as we hoped but most site reports would simply report the programs had compiled." ("Observations on the mathematical software effort," in Sources and Development of Mathematical Software, W. Cowell and C. Moler, eds., Prentice Hall, Englewood Cliffs, NJ, 1984, pp 1-19.)

CODY: Absolutely correct.

HAIGH: And was it that that motivated the production of these more elaborate drivers?

CODY: That certainly played a significant role.

HAIGH: So, you've mentioned that there were a few imitations, and the drivers didn't test everything, but you feel that overall they were a big improvement.

CODY: Absolutely. They were far better than what was likely to be done without them.

HAIGH: And were there any sites that did do more thorough testing on their own for FUNPACK that you worked with particularly closely?

CODY: Well I think the people at the University of Wisconsin were conscientious. What was the fellow's name up there, Wayne Wallce. And later on as these programs were distributed there were people who received them after the fact, not as part of the NATS project but because they were available, that took closer looks at them. I had some friends down at Sandia and Los Alamos and Livermore that sort of place.

HAIGH: In your TOMS paper you mention University of Texas was testing the CDC version.

CODY: Yes, that was Yasuhiko Ikebe.

HAIGH: In one of the papers you also talk about rewriting routines to reflect the coding standards that were derived for the EISPACK project, improving the program's style.

CODY: Okay, that was for our local library, I think. That's the sort of thing I was talking about where the experience, the NATS experience, led us to better programming practices. Certainly I think my own programming style improved after the NATS experience.

HAIGH: Actually I think in the book you say that it was only partially achieved because new sections were written in the new style but the old ones were not rewritten.

CODY: In some cases that's true, yes.

HAIGH: So, you said your programming style improved. Do you think that this idea of there being such a thing as good programming style, and agreement on some elements of it, were a new thing at this point that the project was helping people to understand?

CODY: You know that's difficult for me to assess. I can testify to what happened locally, at Argonne. It may be that many of these self disciplines had been imposed elsewhere without our knowledge, maybe even prior to the NATS project, but certainly it improved what we were doing at Argonne, the way we looked at code.

HAIGH: Was that partly because you were reading code more?

CODY: I think we became more aware of the organization of the code. I know in some of my later programs that I spent a great deal of time reorganizing, shifting things around, to try to make the flow straight forward from top to bottom, for example, rather than having to double back and do something. So it made it easier to read, it made it easier to understand, it made it easier to maintain. As I say, I'm sure these same lessons had been learned elsewhere. It's just that, you know, sometimes it takes time for things to sink in.

HAIGH: How was documentation handled?

CODY: Oh, boy…. Laboriously. Within the NATS project we wrote the documents and we massaged them over and over and over again trying to make sure that we used the right words in the right way, trying to make sure that everything was covered. It's like any editing job; it's particularly difficult. It's awfully difficult to edit your own work, but fortunately we had the services of a technical writer and that helped tremendously.

HAIGH: So, apart from your own work and the people that you've mentioned at Texas and Wisconsin, were there any other people actively involved with FUNPACK?

CODY: I think those were the three major installations, seems to me we sent it out to a few other places at the end just to make sure, but I don't honestly remember who they were. Those were the three primary installations involved.

HAIGH: Now one of the ideas that features prominently in articles on the NATS project is this idea of software that comes with a pledge of certification.

CODY: Yes. We did not guarantee that the software was flawless. I think anybody that would claim their software is flawless is lying. What we did guarantee was that if it malfunctioned we would look at it and fix it and make the correction available. We worried a great deal, there was a lot of discussion, internal discussion. I don't think it was documented anywhere, but we worried a great deal about what would happen if something that we had certified should malfunction in such a way that there was significant loss to somebody. How culpable were we, what were the potential penalties? And I don't think we ever came up with an answer. It's a copout, but we decided finally that because it was a Government organization the best we could do was to be clear up front that the software was not guaranteed to be perfect but we would examine problems and fix them, and essentially beyond that there was no guarantee.

HAIGH: So as I understand it the certification pledge was basically saying that you would support the software in terms of fixing problems that were reported. And was this an open-ended indefinite agreement to support it?

CODY: I think it was intended to be that way. By the time the NATS project had terminated we hadn't had any reports of problems at all, that I'm aware of, and so I think it just lapsed. Is it still supported? I don't know. [Laughter]

HAIGH: So it turned out to be rather hypothetical. There was no definitive budget or assigned people to do the job?

CODY: No, no. I'm proud to say that the job was done well in the first place.

HAIGH: And do you know if this kind of pledge to support had been made by any other projects or groups prior to this?

CODY: Not to my knowledge. That doesn't mean it wasn't, I'm just unaware of any.

HAIGH: Or for example, when people reported problems with SSP to IBM –

CODY: IBM laughed.

HAIGH: So IBM didn't view it as a supported piece of software in the sense that you were aiming to achieve?

CODY: They did originally, I think, but towards the end it was sort of given to you, you can use it if you want. They did issue revisions from time to time as I recall.

HAIGH: Now according to the article the Argonne Code Center was responsible for distributing the software.

CODY: I think that probably is correct. That was not its primary responsibility you understand, it was primarily for distribution of nuclear codes. But they were geared up to do it, and we could see a volume there that we, as individuals in the math division, simply couldn't handle. So, yes, we turned it over to them.

HAIGH: And do you know if they did a good job of distributing it?

CODY: I never heard any complaints.

HAIGH: Was it available to users of all kinds?

CODY: Absolutely. I'm not sure, there may have been some problems with Iron Curtain countries. I just don't know.

HAIGH: Do you know if there was any kind of copyright message included in the software?

CODY: Interesting question. Not to my knowledge. I don't think so.

HAIGH: And presumably users didn't have to agree to any kind of license terms.

CODY: No, no. Perhaps they had, you know, as I say, if distribution was prohibited in Iron Curtain countries there may have been a clause from the code center that they had to sign saying they wouldn't pass it on. I don't know.

HAIGH: So did this issue of intellectual property ever come up at all during the project?

CODY: No.

HAIGH: So as far you were concerned it was something that was completely in the public domain?

CODY: Absolutely. Almost all of the work that our division did was in the public domain.

HAIGH: Why was that?

CODY: We were funded by tax dollars.

HAIGH: So at that point your idea was that the best way to maximize public value was to give the software away?

CODY: Yes.

HAIGH: And nobody ever suggested that if you charged for it that you could recoup some of that public investment?

CODY: I'm sure the suggestion was made but it didn't fly.

HAIGH: No one would have taken it seriously?

CODY: No, I think the laboratory had a policy in general that prohibited such things. Later on, I'd be hard pressed to come up with a date but I would say in the early 80s, they set up a technology transfer center in which the laboratory helped coordinate the transfer of intellectual property developed at the laboratory to the private domain. A number of spin-off startup companies were formed in that way, but, unless they were classified, the developments at the laboratory were freely available.

HAIGH: You mentioned in your article in Cowell's edited book that the FUNPACK routines were modified and incorporated into the IMSL library.

CODY: Yes.

HAIGH: So there was not even an objection to commercial use of derivative code?

CODY: Absolutely not. If NAG wanted them they were available. You see IMSL and NAG were providing additional services. So they weren't necessarily selling the FUNPACK routines, what they were doing was packaging and selling maintenance and an overall package.

HAIGH: You have mentioned that you thought that the package was widely used among scientific users of IBM, CDC and Univac.

CODY: I'm convinced of that. I had friends in JPL which used Univac equipment, at the weapons laboratories which used CDC equipment, obviously a lot of people using IBM equipment that told me they were using these programs. They went overseas, they were used as CERN, there were some astronomy places, Pic de Midi and a few other places that astronomers were using them, so, yeah.

HAIGH: And did you ever get feedback from users?

CODY: Sometimes.

HAIGH: What kinds of things would they say?

CODY: They enjoyed having them; they worked. It was a pleasure to have programs that they didn't have to worry about.

HAIGH: And do you know if the code found its way into any other libraries aside from IMSL?

CODY: Well, as I say, it was incorporated into the libraries at places like JPL and it found its way into university and computing center libraries.

HAIGH: Now returning to the issue of portability. What was it about these kinds of functions that made it harder to write a portable version than for the higher-level things like linear algebra

CODY: Well, linear algebra codes deal with matrices that are generated by applications programs somewhere. They have to be able to handle those matrices, but there's no question as to last bit accuracy. It's just a question of doing the right thing, getting the most you can out of the matrices that you're given. Doing it in a numerically stable way. With special functions there's a one to one relationship between the input argument, or arguments, and the output, and there is a correct answer. So the real problem is to determine how close to that correct answer you are, to get as close as you can. Because different machines have different arithmetic systems, what works on a Univac machine, word length, expressions, number of terms in approximation and so forth, is not going to work on a CDC machine, is not going to work on an IBM machine. You've got to have something specifically tailored to the individual machine if you're going to come up with the best possible result. And, as I said, there's an awful lot of work that goes into that, and the question that arises after, what did we spend two years, three years, four years, the question that arises is: is that much effort worthwhile if you only get a handful of routines for three different machines. You know those machines will become obsolete in another year or two. Is that the right approach? I convinced myself that it could be done but I also convinced myself that there had to be a better way, that's where SPECFUN came.

HAIGH: So, just before you move onto the details of SPECFUN, did you realize that there were techniques that would have been available at the time of FUNPACK that could have made it more portable?

CODY: Yes.

HAIGH: Okay, so it wasn't just that machines got bigger and compilers got better and hardware got more standard?

CODY: No.

HAIGH: Okay. Let's move on to them.

CODY: The big difference between FUNPACK and SPECFUN is that SPECFUN was designed to be portable. Let's say, you've got a function computation that is based upon an approximation. In FUNPACK the coefficients were given in the native representation of the machine, binary, hexadecimal, whatever, where the built in representation of the significand and the exponent varies from one manufacture to another. In SPECFUN, instead of relying on those coefficients being given in that form, which is very exact but non-portable, we tried to devise ways where we could give the coefficients in a decimal representation, but in such a way that we would extract almost as much significance out of the coefficient as we would the other way.

For example, we set a maximum accuracy of, let's say, twenty decimal digits. Now if you put in a coefficient of twenty decimal digits, the machine compiler is going to lop that off and take whatever part it needs, and ignore the rest. In some cases, I don't remember if we did it universally, you break that coefficient in two. You can provide a higher order term, the most significant part, in a representation that is exact in binary or hexadecimal (and of course in decimal if there are any decimal machines out there) and then you can provide the rest of it. Now you've got twenty, twenty-two, significant figures total. But if you're clever in the way you use those two constants, you first of all convert the low order part, so you get that correct to within what the compiler thinks is rounding error, convert the high order part which is exact, and combine the two, and now you've got a full length term that is accurate to within rounding error. So it's that sort of thing. You sacrifice a little bit of the exact precise control that you had earlier. It's not an exact representation in the machine. I haven't done the conversion myself, I've relied on the compiler to do it, but I've tried to instruct that compiler. I've tried to give it instructions, so that when it finishes it comes up with something that is very, very close to what I intended. So we sacrifice a little bit.

HAIGH: And is that more portable because you're relying more on capabilities that are built into the compiler?

CODY: Yes, yes. Well, I'm using the same coefficients for all machines, I don't have to have a separate version for CDC or Univac or IBM or, you name it. So it's that sort of thing.

HAIGH: So, that's a different approach from the one that some of the libraries were taking during the 1970s in terms of automatically generating optimal code for different platforms based on some parameters that were created to represent machine characteristics?

CODY: Yes. Oh, we still had machine characteristics built into this. You had to know what the underflow threshold was, you had to know a lot of different things, and so those parameters were built in as well, or they were determined on the fly.

HAIGH: So, it was automatically detecting some of those things?

CODY: Yes.

HAIGH: Was that a novel idea?

CODY: Yes and no. The first programs that I'm aware of for detecting machine characteristics went back into the early days of the algorithm column in the CACM. They did the obvious things for trying to determine underflow threshold, overflow, that sort of thing. When I produced the book on mathematical software, which dealt with elementary functions, I had written a code called Machar, machine characteristics. It was an amalgamation of everything I could find in the literature, polished so that it would work on as many different machines as I could find. That little subroutine calculated all of the machine characteristics that I think I would ever need. For each of these libraries, it was the method for determining the characteristics of the machine. That was being revised right up until the time I retired. We built into it, for example, techniques based on the IEEE arithmetic, so that it would detect whether it was an IEEE machine, it would detect whether it was little-endian or big-endian, as I recall. There were a lot of things that program did.

HAIGH: I shall ask you more about the book in a minute. So, the techniques that you developed in terms of splitting the coefficient for additional accuracy, would they have been applicable in a broad range of different kinds of mathematical software or were they most useful in this niche of special functions?

CODY: Well, special and elementary functions, certainly. I don't know enough about other types of software to be able to answer that question, I guess. My gut feeling is that it's much more important with function software than it is with others.

HAIGH: Yeah. And do you know if it's an idea that was taken up by anybody else?

CODY: I would assume it was, but I really don't know.

HAIGH: Oh, let's pull back and deal with some of the more mundane things about SPECFUN then. So, when did you start work on the project?

CODY: Well I think it sort of evolved, I started roughly about the time that we made the final release of FUNPACK. I began to think about these things at least, it took several years for things to gain momentum.

HAIGH: So, at what point was this package released?

CODY: I'd say in the late 80s, just a few years before I retired. The open literature publication that described the final version was one of the last papers I published. I remember I worked on two or three of them after I retired. That came out in 1993, that was two years after I retired. It took them a while to referee it. ("Algorithm 715: SPECFUN: A portable package of special functions and test drivers," TOMS 19, 1993, pp. 22-32.)

HAIGH: And did that project receive any kind of special funding or grant or was it just something that you worked on as part of your Argonne employment?

CODY: No, it was funded by Argonne. That particular package was very gratifying to me, I released elements of it from time to time and of course the final version was released about a year before I retired, something like that. And I think it was largely the reputation of FUNPACK that helped, but that was gobbled up. I got requests for that package from literally around the world and aside from sending something behind the Iron Curtain, which I couldn't do, it was widely distributed and widely used, as far as I could tell.

HAIGH: How was it distributed?

CODY: I distributed a lot of it myself.

HAIGH: By email?

CODY: Oh, no. No, we'd send a tape. Jack Dongarra had Netlib, and so a lot of it came through Netlib.

HAIGH: And how did your methods of testing and development compare with those of FUNPACK?

CODY: Different. FUNPACK we provided a driver, as I said, it required a magnetic tape where there were combinations or pairs of arguments and results, SPECFUN the individual drivers were largely self-contained and self testing. It wasn't a question in this case of testing last bit accuracy, because we figured we didn't have it anyway. But we could certainly run all the robustness tests we could provide, and in fact there are a number of papers that describe some of the testing techniques. We could provide programs that would look at identities, mathematical relations, that the function had to satisfy. This was something that came out of the work on the elementary functions in that book. If you were careful on your error analysis you could find regions, combinations of identities and regions, or intervals, which the computation error did not build up. In fact it would tend to damp out. So there was solid error analysis behind these things, and we simply ran those identify checks in those areas. Of course you had to program the test drivers with as much care as you did the function program. You wanted to make sure that the test driver was really doing what it was supposed to do, and not taking a short cut that somebody's compiler had suggested.

HAIGH: Did you have a network of test sites again?

CODY: Informal. By that time I had friends all over, various computing centers and so it was just a question of sending it out and asking, you know, what do you think of this one.

HAIGH: And was there any kind if pledge of certification and support?

CODY: No.

HAIGH: They would use it at their own risk?

CODY: Absolutely.

HAIGH: Did you in fact have any reports of problems with it after it was released?

CODY: I think one or two. Not after it's release though. During the testing phase, I think there were a couple of places that reported minor errors that were easily corrected.

HAIGH: Was there anything novel or unusual about the library other than it portability?

CODY: Well it certainly wasn't as complete as some of the other libraries. Fullerton's library, for example, computed a lot more functions than this one did. On the other hand it couldn't stand up to robustness or accuracy. So, I think this had a good reputation, it was solid programming and it was reliable, definitely transportability.

HAIGH: Do you know if the code found its way into any other libraries such as NAG or IMSL?

CODY: I guess I honestly don't know. As I say, it was adopted at CERN and the weapons labs, you know, JPL, every place that the previous code had been adopted. I got letters from places in Italy, in Switzerland, and South American, that had incorporated it into their library, some of them simply asking if it were really true that this was free [LAUGHTER], and others commenting on the quality. So I think that all told I had about maybe sixty or seventy different installations that adopted it, and to me that was success. But did it find its way into commercial libraries? I have no idea.

HAIGH: So, was there any kind of copyright on the code?

CODY: No, none at all.

HAIGH: To the literal extent that it didn't include even a little copyright symbol in it?

CODY: Nope.

HAIGH: And by this point had Argonne become more sensitive to these issues, was there a technology office for example, or a strict policy of some kind?

CODY: There was a technology transfer office but their concern, as far as I could tell, was not so much with computer programs as it was with, for example, a device developed in one of the chemistry labs, a sniffer if you wish. A little device that could take an air sample and diagnose what was in the air. The interest at Argonne, of course, was to be able to detect poisonous gasses and dangerous compounds that might have escaped, that sort of thing. But clearly, clearly, that sort of device would have use outside of Argonne's, fire departments. You name it. So the technology transfer division helped, a department, whatever they were, helped transfer that technology to a private company that I believe was founded by the individual who invented the device, so he now was an entrepreneur on the outside and a scientist at Argonne.

HAIGH: But in the case of software you didn't have to get any kind of clearance or permission, you'd just go ahead and give it away?

CODY: I don't recall any problems.

HAIGH: So, how about your parallel work on elementary functions?

CODY: Well of course that's where everything started. You know you go way back to when I first became interested in function computation, and it had to do with testing the libraries that were provided by the computer manufacturers. We developed and programmed replacement libraries for every machine that was delivered, and it just got to be a chore, something that we didn't want to do over and over again. And I finally got the bright idea of writing a book that would lay out all that we had learned about writing such libraries. Provide flow charts, if you wish, for each function, provide the coefficients, provide implementation notes that would help one to write the codes themselves. We developed Machar because we needed it, in fact Machar had been under development prior to that time, but we certainly needed that to determine the characteristics of the machine that would host these functions. We developed self-contained programs, that's the beginnings of what we did with the SPECFUN package, we really honed our skills on that one. And the programs were designed to run on three different types of machines. The implementation notes talked about a binary machine, a hexadecimal machine, and because somebody insisted, a decimal machine. The only decimal machine that I am aware of, that was in existence at that time, was in the electrical engineering department at the University of Colorado, Bill Waite, and so I sent him the notes, he had to do the programming himself. He found a few things and it was a good collaboration there, and eventually he became a coauthor of the book. Now that was one of course that was copyrighted, and there were some royalties, not enough to retire on by any means, but the royalties were split two ways between Bill Waite and myself, and my portion of it went directly back to the laboratory because all the work had been done on laboratory time.

[Start of Tape 3, Side A]

CODY: One of the interesting things about this book: we worried about proofreading after it had been typeset, there were an awful lot of coefficients, an awful lot of things to check, and we decided that the easiest thing to do was to produce that book ourselves in photo ready form. That had never been done before. The people at Prentice-Hall were a little suspicious. Our technical writer had to do some ground work and spade work to find out where we could get these things photo set, I think she eventually went down to the University of Illinois, if I remember right. So we would write a section, proof what we had done, verify by recoding based on flow charts and everything. Then when everything was ready we sent it down and had it photo type set, and when it came back we went through the same process. And so we were convinced, when we finished that that book didn't have any errors in it. Now, of course, you know that's not right. But we sent it off to Prentice-Hall and Prentice-Hall produced the book in something like six weeks, which was amazing. After that I think most of the books that came out of our division were published in that way, but that was the first one, and it set a precedent at Prentice-Hall as well. Now it did turn out that there were a couple of minor glitches, but nothing that was important enough to put out a second edition with an errata.

HAIGH: And that book was Software Manual for the Elementary Functions, published in 1980? (Software Manual for the Elementary Functions, with W. Waite, Prentice Hall, Englewood Cliffs, N.J., 1980.)

CODY: Yes. That was the textbook at a number of places. Kahan loved it.

HAIGH: So, was there a close relationship between that book and the ELEFUNT package?

CODY: Well, yes, the ELEFUNT package is the test programs from the book.

HAIGH: Okay, so that's why the accuracy was so important, was the idea that people would be keying in the routines from the book?

CODY: Yes.

HAIGH: And were they also available in the machine-readable form?

CODY: Well, for the algorithms in the book for the elementary functions there were no programs, it was just flow charts and implementation notes. The test programs were in there.

HAIGH: So, the idea would be that someone would use those algorithms and charts to create some code and then they could use your test programs to test the code that they produced?

CODY: Correct.

HAIGH: And so what kind of course would you set it for as a textbook?

CODY: Elementary numerical analysis in a computer science department.

HAIGH: So, they would be learning about how the software worked by writing their own?

CODY: Yes, that's one possible benefit. I think more importantly, if they examined the flow charts they could see the techniques, the tricks that were necessary to extract the accuracy.

HAIGH: Now, did that project receive any kind of special funding?

CODY: No.

HAIGH: So that was something that you were working on?

CODY: Yes.

HAIGH: And you've said that that was a combination of this effort in testing and rewriting software that had gone back all through the 60s and 70s?

CODY: Yes.

HAIGH: So that would also be the basis of the work that you reported on in your paper in John Rice's first mathematical software volume? ("Software for the elementary functions," Mathematical Software, J. Rice (ed.), Academic Press, New York, 1971, pp. 171-186.)

CODY: No, this was a survey article, simply trying to, well I think some of the techniques are in here, this business of breaking coefficients up and that sort of thing, but this was an attempt to survey what was available, and it wasn't very good. This was early. This was what, '69?

HAIGH: The article wasn't very good or what was available wasn't very good?

CODY: What was available wasn't very good. No, I think the article served its purpose.

HAIGH: Are there any of your other publications in this area, prior to or as well as the book, that you think were particularly important?

CODY: Well, I've already mentioned the self-contained power routines, exponentiation, I think that was extremely important, I believe that Clark, Kuki and I changed the way in which that function was computed. I don't know how they do it today. ("Self-contained power routines," with N. A. Clark and H. Kuki, Mathematical Software, J. Rice (ed.), Academic Press, New York, 1971, pp. 399-415.)

It was in the math software volume, from Rice's math software volume. No, I think we've covered most of the important stuff.

HAIGH: Now are you aware that the test software had a usage outside the educational functions of the book itself?

CODY: Oh, absolutely. Yes, I think a great many people began to examine the libraries on their own machines with that software, and in some instances decided that they were not as good as they had thought they were.

HAIGH: Do you know if computer manufacturers or end compiler producers would be using it?

CODY: I think it became a standard set of tests that had to be passed. That doesn't mean that it was definitive. I'm sure they did additional testing, but I think it was accepted to the point that if it didn't pass that test then they had to go back to the drawing board.

HAIGH: And was it straightforward and self-explanatory to use?

CODY: Yes, I thought so. You'd have to talk to users but, yeah, I thought it was.

HAIGH: So basically it would be an automated diagnostic type thing, that you would run it and then you would get this list of things that really weren't working as they should be?

CODY: Well, it was an indication of accuracy. It also checked for things, I don't remember specifically, but say for the square root one of the last things it did might be to feed it a negative argument just to see what the hell happened.

HAIGH: Sure. Did you get any feedback from users?

CODY: Not from manufacturers, I got feedback in the sense that there were a lot of requests for the software. So, it obviously was being used somewhere. Towards the end, I extended the programs or wrote counterparts for the complex elementary functions. I wrote some software for subroutines that were provided or required by the various language standards that were not considered to be elementary functions. For example, Max and Min. Those were done once. I retired before there was any iterative correction to them so I have no idea how good they were.

HAIGH: Now by this point was there any kind of rigorous Fortran standard that would come with a set of test programs?

CODY: Well, I don't know about the test programs. Of course Fortran was subject to standardization efforts dating way back to the very beginning. It initially was a commercial program produced by IBM on the 704, but the standards bodies got involved and X3,J3, I think that was the American Standard Association, ASA, that set up a standards body that wrote the specs for the later versions of Fortran. They're still active. That became an international standards body and there were people at Argonne that were involved in that, Brian Smith, for example. He was also a member of the IFIP Working Group. So the IFIP Working Group 2.5 was active in the sense that they discussed what was going on in the standards committees and made recommendations, suggestions.

HAIGH: So, by that point were any of these broader standardization activities having an impact on the quality and standardness of elementary functions in Fortran systems?

CODY: I don't think that changed much, no. No. I think the libraries were getting better.

HAIGH: Now you'd also mentioned the CELEFUNT package.

CODY: Yes.

HAIGH: What was the relation of that to ELEFUNT?

CODY: Well ELEFUNT stood for ELEmentary FUnction Tests, CELEFUNT stood for Complex ELEmentary FUnction Tests. So it was a set of programs designed to test the complex elementary functions. As I say, I don't know how successful it was. It made its appearance just shortly before I retired.

HAIGH: So that was the extension that you had alluded to a few minutes earlier?

CODY: Yes.

HAIGH: ELEFUNT was first distributed in the book, but were it and CELEFUNT also available through sources such as the Argonne code center?

CODY: Probably, but certainly through Netlib, and they were submitted for publication in TOMS, so they were available there.

HAIGH: So that would have been in microfiche form with the journal and also available separately in machine-readable format?

CODY: Yeah, I'm not sure how the journal handled it, but yeah.

HAIGH: And can you talk how the applied mathematics effort at Argonne, as a whole, developed through the 80s, and into the early 90s?

CODY: Oh, I think we've pretty well covered the ground. We had people with diverse interests at the laboratory. There were pure mathematicians, there were people who were mathematical physicists, Joe Cook and others, they were logicians, Larry Wos and his group, there were three or four of them in there. There were people who were interested in quadrature, James Lyness. There were people who were interested in optimization, Jorge More. With the exception of the pure mathematicians, each of these groups became deeply interested in mathematical software. So you had Jorge More, for example, helping produce a suite of programs eventually for optimization. I've forgotten what the name of it was; it was a PACK. The logicians produced a series of programs actually for automated reasoning and out of that effort one of the members went on and worked on the genome-sequencing problem in collaboration with people from biology and chemistry and people at other installations. Lyness produced a suite of programs for quadrature. I think he called is QUADPACK, if I remember correctly.

And of course these people became involved in a larger community outside the laboratory. That is they had collaborators from all over, much the same way that the linear algebra projects early on had attracted Wilkinson and Vargas and Birkhoff, and the rest of them. These projects attracted people from other installations that were interested in the same field, and Argonne became an exciting place to be. Especially during the summer when the university people were not teaching classes, my golly, you'd run into some well known people roaming the halls at our division. And with the full and enthusiastic support of management, that makes a big difference.

HAIGH: And that was still the case when you retired in 1991?

CODY: Largely, yes. I suspect that it's still the case today, I see from time to time, I still get publications of course from the laboratory, and I see from time to time articles that highlight activities of members of the division. Some of them were just coming on board when I retired, some of them I've never heard of before, but their contributions are every bit as important I think in their field, nationally, as ours were at the time.

HAIGH: Let's turn to examine the IEEE floating point standards.

CODY: Ah, ha, ha. You know, there are a few things that I'm very proud of. I think SPECFUN, MACHAR, and the book, and you know, they each contributed something at the time, they were on the cutting edge of what was being done. That IEEE effort, it wasn't my idea, that IEEE is amazing. Vel Kahan is the founder of that effort. Now Vel is a very difficult person to get along with if you don't know him. If you know him, and have his respect, he's a great colleague, and friend too. He became disgusted with the design of the arithmetic engines that were available to us, he was right alongside Kuki. He supported Kuki in his complaints about the IBM 360 back in the late 60s, early 70s.

He was well aware of what was going on with the various software projects, an enthusiastic supporter, when something came out that really made his life easier. He was a professor but he was also a consultant at a large number of different places. Remember those hand held scientific calculators of the 1970s? Those things cost an arm and a leg, and there were two companies that were producing most of the scientific models: Hewlett-Packard and Texas Instruments. Somehow or other he got his toe in the door at Hewlett-Packard, when they put out these little HP 35 hand calculators. He convinced them to use some algorithms, for not only the elementary functions but also some other computations on there, that were of his design. They proved to be so far superior to what Texas Instruments was doing that Hewlett-Packard became the dominant name at least as far as the scientific community was concerned. People still bought the Texas Instruments for their students, they were cheaper, but you could demonstrate the superiority of the arithmetic and the elementary functions in the Hewlett-Packard, and his reputation in that area grew.

For years, as I say, he had been trying to convince the computer manufacturers to do things properly. He was on Cray's case from the very beginning, CDC. He was on the IBM case. You have to know Vel. We used to say there's nothing worse than a loud mouth unless it's a loud mouth that's correct, and that describes Vel. Very abrasive. Well, there suddenly was a new player in the game and that was the chip manufacturers. We were beginning to see micro processors, and it looked like there were going to be some personal computers. Intel was one of the early players, and there was a fellow at Intel who had been one of Kahan's students, and he convinced Intel to bring Kahan on as a consultant. Well, that's when it hit the fan. Kahan saw an opportunity to influence, at the very beginning, the arithmetic design on these chips, and he figured if he couldn't convince the major manufacturers to do it right, but he could convince the micro processor people to do it right that eventually, when those micro processors and personal computers became important, that the major manufacturers would take notice. He certainly had no guarantee, but that's what he hoped.

HAIGH: And do you know roughly what date this was?

CODY: It was mid 70s, 1976. Anyway, somehow the IEEE got involved. I'm not quite sure how that came about, but at that time the IEEE had standards committees on everything. You'd think of them as producing standards for computer busses, connecting cables, ethernet cables, that sort of thing, but they were involved in everything. They were producing standards for anything that came along. Light bulbs, computer languages, they had computer language standard groups working, independent of course from what everybody else was doing, and so they saw an opportunity here and they formed this standards committee to draft a floating point standard for microprocessors.

I can remember some of the big players, there was Fred Ris representing IBM, I'm not sure if the people at IBM knew what he was doing initially. There were people, Mary Payne and a few others from DEC, there were some people from Motorola, there were some people from Intel, there were some people from Apple. There's a list in ("Analysis of proposals for the floating-point standard," Computer 14, No. 3, March 1981, pp. 63-68 and "A proposed radix- and wordlength-independent standard for floating-point arithmetic," with J. T. Coonen, D. M. Gay, K. Hanson, D. Hough, W. Kahan, R. Karpinski, J. Palmer, F. N. Ris, and D. Stevenson, IEEE Micro 4, No. 4, August 1984, pp 86-100). They were the companies you might think of normally as being involved in that sort if thing.

There were a number of proposals being floated. One was Kahan's visionary concept of what this floating point arithmetic would look like, it would be designed to be favorable for numerical computations, including interval arithmetic. He wanted some hooks in there that would make interval arithmetic fairly easy to implement. And there were some counter proposals, as you can imagine, the people from DEC, for example, were interested in an arithmetic system that looked somewhat like the VAX, and they modified things to accommodate the others. There were other proposals. There were three or four of these things floating around, and I gather that the meetings had become rather heated, that in the early meetings, there was a great deal of conflict between Kahan and his supporters and some of the others, to the point where not much was accomplished when they had a meeting. Nothing was ever said, but my private feeling is that they needed somebody to moderate things. Kahan wanted somebody who was aware of the numerical analysis problems, had a working knowledge of what was wrong with the existing arithmetics, that sort of thing. That's how I got invited and joined, about a year, maybe two years after the effort got under way. We spent another, three, four years, something like that, having periodic meetings, once every three months, ironing out the details. Gradually some of these people dropped out as it became clear that they weren't going to win, and that Kahan was largely going to carry the day. We argued over some of the same things time after time after time but it always came out the same way.

HAIGH: And do you think the disagreements were they coming primarily from an intellectual level, or were they motivated by the existing technologies that the manufacturers had in place?

CODY: Both, both. One of the problems was that Intel, with Palmer's goading and direction, had already produced a chip that implemented much of what was to become the floating point standard. So here was hardware in place, and of course that was suspicious to people from DEC and Apple and other places. Yeah, it took a while. There was a selling job that had to be done. Many of us spoke at various meetings around the country, math meetings, computer meetings, I spoke at a math meeting up in Washington, as I recall. A lot of times, in addition to the sessions that were scheduled at a meeting, there would be a special evening session called to discuss the IEEE floating point standard or proposals. There was a lot of writing: we published articles in computer journals, we published articles in math journals, applied math journals, that sort of thing.

Eventually the thing was adopted. There are some things in it that I think were perhaps a little bit of overkill, in hindsight. On the other hand we had established a lower bound on what was acceptable. So suddenly, on the micro-processor level at least, we had an arithmetic system for which we knew the characteristics, that they were good, and they were designed to make our job easier. Now the problem was, and I suspect it still is, there were some things in that standard that required implementations in software. If you were going to support NaNs and gradual underflow and all of the alarms that could be raised for various things, you needed language support. How many languages are there out there? Well there was Fortran, there was Algol, there was ADA, there was Pascal, you know, you name it, and none of these languages supported that sort of thing. It would be years and years before the standards got to the point where they would support it. I don't know if they do yet. So that was a stumbling block; you had things on the chip, some of the finer points, that certainly were not accessible. It required a library of, I don't want to call it elementary functions, but fundamental functions, that nobody had. It did not address the issue of how to you get some of these quantities in and out, the I/O conversion problems. For example, NaN was something special to that particular standard. It's called Not A Number. When you divide by zero, you get a NaN. I don't know what it is, but it's Not a Number. Well, how do you detect that, how do you output it when you print out your results? If you want to put a NaN in for some reason, how do you get it in? Those issues were not addressed. It was difficult enough getting that standard as an arithmetic standard through ANSI and ISO, let alone tackling all of the language standards committees. So in a sense the job was only half done, but we couldn't have done the whole thing.

Now after that period, or rather late in its development, somebody at IEEE came up and said, "well, what the heck, that's a binary standard. There are hexadecimal machines out there, there are, who knows there may be some octal machines. That standard doesn't apply." And so they formed a committee for a radix-independent floating point standard. They appointed me chairman. I'd have no idea why. Many of the same people were involved, some of them had dropped out. DEC I think had by that time given up and dropped out, but our charge was to produce a standard that would apply to radices other than the binary, that was in a sense upward compatible. That is, if you complied with the binary standard you complied with ours. We couldn't do things that the binary standard didn't do, but, as I recall, there were a couple of little things that we added in, and that effort took another, I don't know, eight years.

HAIGH: Right, it shows you chairing this group from 1981 to 1987 on your resume.

CODY: Okay, seven years. And the standard was produced and adopted. It too is a standard. It's ignored; the binary standard is the one everyone uses. The reason it's ignored is because nobody wants to do that sort of thing on a hexadecimal machine, for example. We could find no justification for trying to do it on a hexadecimal machine, we limited ourselves to a subset of possible radices, and said that what we had done could be extended to hexadecimal if you read the rules correctly, but there really wasn't much sense in doing it.

HAIGH: So had the idea been that people might want to come along and produce micro processors that didn't use the binary system, or was the idea to extend it to larger kinds of machines?

CODY: I think it was just an IEEE effort to say they had extended it. My gut feeling is there was no practical purpose to it, beyond the fact that it refined certain things that the binary standards seemed to be permissive about.

HAIGH: But presumably you believed, at the time, that it might have some kind of practical impact.

CODY: You know there was a hope, but I don't think any of us were under any illusions, that it was going to be as earth shattering as the binary.

HAIGH: Now did either of the standards finish up having an impact on machines that were larger than the microprocessor based computers?

CODY: Oh, absolutely, absolutely. Towards the middle 80s, on, we were beginning to see things like these hyper cubes and Sequent and Alliant. A lot of machines that were billed as powerful scientific computers were parallel machines and had all sorts of strange architectures. Most of those relied on micro-chips and therefore they had the IEEE floating point arithmetic or something very similar. Kahan told me just a few years before I retired that Cray was even looking into the possibility of using that arithmetic on some of their newer machines. I don't know what happened; I haven't paid any attention since then. But Kahan got the Turing Award for that work, and deservedly so.

HAIGH: And do you feel that the early hardware implementations that you saw were satisfactory?

CODY: Of the IEEE standard?

HAIGH: Yes.

CODY: Yes, yes. They didn't have all of the bells and whistles, as I say, they weren't supported by the software, by the languages the way I would liked to have seen them supported, but, oh, it was a pleasure to write programs for those machines.

HAIGH: And how would you characterize your personal contribution to the floating point standards effort?

CODY: The mediator. They'd get into these arguments about some aspect of it, and go round and round and round, and to me the ultimate concern was how does it affect what I want to do? How does it affect my preparation of software? So I brought that viewpoint in there. I was the only one who was actually writing computational programs, that is mathematical software.

HAIGH: Oh. So were there any examples where you think that without that users viewpoint they might have done something that would have proven unwise?

CODY: Oh, I have no idea. You've been to committee meetings [Laughter].

HAIGH: Yes. So apart from Kahan, were there any other individuals that you think were particularly important contributors?

CODY: Well, I think John Palmer, obviously, from Intel. David Stevenson, I don't know where he is now, he's changed jobs so often. Jerry Coonen was one of Kahan's students; he wrote a paper in defense of the standard and that paper, we finally convinced him, was good enough to serve as his Ph.D. thesis. It took a long selling point on that, and I think I was the one that finally convinced him, and so he's still on the West Coast somewhere working for one of the manufacturers.

HAIGH: And you'd also mentioned that you had served as an unofficial liaison between this group and the IFIP Working Group 2.5?

CODY: Yes.

HAIGH: What kinds of communications passed between the groups?

CODY: Well, IFIP Working Group relied on me to keep them informed on how things were progressing, what the stumbling blocks were. It came to the point that we were beginning to worry about getting the standard through the various committees, to get IEEE, and eventually the international standards organization, to approve and adopt the standard. WG 2.5 supported us with proclamations of support, formal letters to the organizations expressing their support for what had been done. You need that sort of thing from a lot of different people in order to succeed, so I think they contributed.

HAIGH: Now the other group that you were a member of was the SIGADA Numerics Working Group?

CODY: Yes.

HAIGH: What kind of challenges was that group addressing?

CODY: Well, ADA as you know is, or was, the official programming language for the Department of Defense. That was a mistake; they should never have done that. But it was an effort on the Department of Defense's part to standardize the programming language throughout all of their interested groups. The problem was that Ada was not sufficiently specified. There were some gaping holes, compilers didn't exist, it was a rat's nest. But, somehow or other, there was a group that was formed to look into the elementary functions for ADA. One of the people at Argonne had a subcontract with Department of Defense to work on ADA. He became involved in this subcommittee and he dragged Brian Smith and me, and some people from NAG, as I recall, were also involved. We attempted to specify standards for elementary functions, for test programs, test procedures. The work expanded to include complex arithmetic, complex elementary functions, and at the time I left they were still struggling. I have copies of some very thick reports that were produced just before I retired that indicate that the work was still progressing, and they were making progress, albeit very slowly.

HAIGH: And was the idea that this would make Ada into a language that was suitable for doing freestanding scientific computations, or was the concern more that you had to have good mathematical functions there for it to work for real time control and those kinds of applications?

CODY: I think both, I think both. But certainly the scientific computation was a major concern.

HAIGH: And do you have any sense of whether Ada eventually became a significant force in that area?

CODY: I have no idea. My gut feeling is that Ada is much like PL/1, it's a lost cause. But there are people all over the world that are concerned with Ada the same way they were concerned with Algol, so who knows.

HAIGH: Well that was my impression too.

[Start of Tape 3, Side B]

You also served towards the end of your career as a consultant for a number of groups. According to your resume, as a consultant to IMSL from 1973 to '81, as a consultant to C. Abaci from 1982 onwards, and as a consultant to Apple from 1986.

CODY: Yes.

HAIGH: What kinds of work were you doing for those groups?

CODY: Well IMSL, of course, was interested in producing libraries and so I reviewed function programs for them. I made recommendations of things that might be added into their library. That's about it. We made sure of course that they got the latest work from Argonne as soon as it was in the public domain. They did not get anything ahead of time, that's important because that would have been a conflict of interest.

HAIGH: And how would you compare the strengths and weaknesses of the IMSL and NAG libraries?

CODY: Well I think in the early days they were probably comparable. My gut feeling right now is that NAG is a far superior library. I don't have anything specific to base that on.

HAIGH: And in what kind of time do you think NAG would have pulled ahead, you think sometimes in the 1980s?

CODY: I think so. If you've talked to some of the people down there, you're well aware that IMSL were undergoing some management problems, and I think NAG got the jump on them. Of course one of the things that really bothered Ed Batiste, you had to know Ed he was paranoid about a lot of things, was that NAG was a major competitor and was funded, at least initially, by the British Government. He found it difficult to reconcile the fact that he had to use private funding to compete with the British Government. But that aside, I think NAG had more visionary leadership, they had a wider participation and people that were contributing software, they had just a better way of doing things.

HAIGH: So, back in the 1970s there weren't any particular areas that stood out for one library or the other being stronger?

CODY: Oh, I think IMSL had the superiority in statistical routines, that of course was Ed Batiste's strong point, but I don't know how they compare now.

That was about the time that the IEEE floating point arithmetic standards were being implemented, and Apple came out with a Macintosh that had a new chip in it with the IEEE arithmetic, and they had a new Fortran compiler that they were working on that was supposed to support that chip. And so they gave me a laboratory mockup of that computer and the compiler and I was supposed to run it through its paces and report back to them.

HAIGH: Was that chip the standard Motorola coprocessor or was it something custom?

CODY: I've got a feeling it was the Motorola processor.

HAIGH: Yeah. I know I think the earlier 68000s, I think the 68000 and 68020 definitely had an optional co-processor. I think maybe the final chips in that line had it integrated, maybe the 6840 onwards.

CODY: I don't know that detail. I just know it was fun to play with.

HAIGH: Yeah. And do you think they did a decent job of supporting it?

CODY: The compiler was under development, yeah, I was impressed. I don't know what the final compiler looked like. I have never run anything beyond that initial compiler on that lab mockup. That was a fun one to play with. And of course eventually Hewlett-Packard came out with HP 71B. It was a little bit larger than the usual hand calculator. BASIC was the imbedded language, but this was Kahan's design and it had a full implementation of the IEEE standard. Oh that was fun. I've still got that.

HAIGH: And C. Abaci which was Batiste's new company?

CODY: Yeah, there I served in the same role as I did for IMSL. I don't remember when that terminated. There was, I think that was sort of honorary, I don't recall much money passing forth, passing back and forth.

HAIGH: And you've given the impression that the company wasn't terribly successful.

CODY: I don't think it was.

HAIGH: Why not?

CODY: Well, for one thing Ed was constrained, when he left IMSL, when he sold his interest in IMSL. I believe one of the conditions was that he not start a competitor. And so he had to find a new niche. His approach here was to produce something he called the scientist's workbench, or some such thing; it was sort of a glorified SSP. That's perhaps not being kind to Ed, but it was an outgrowth of that experience, I believe. It was an attempt to provide an environment that you would install on your computer that would automatically make available all of these subroutines that we could gather, things that were in the public domain, much like MATLAB is today. If you've talked to Cleve Moler, okay, it was a poor man's version of MATLAB, and it just wasn't very successful.

HAIGH: Are there any other topics you think we should discuss?

CODY: No, I think you've pretty well covered them. Brings back a lot of memories, things I hadn't thought about in ages.

HAIGH: Well, I'll finish up with the same two questions I've asked the other interviewees then. So, the first question would be: looking back on your career, what would you say the single thing that you would most regret is, that perhaps you wish had come out differently or that you might have done something differently?

CODY: I don't know how the others answered, but you know I had one hell of a ride. I wouldn't change a thing, successes or failures. It was a great life.

HAIGH: And so what do you think the single thing that you would be most proud of from your career would be?

CODY: Oh, boy. I've got a lot of things I'm proud of. Single thing, wow. Well, I'm proud of the book, I'm proud of the special function packages, especially SPECFUN. I'm proud of that MACHAR program. To my knowledge, that's still being used. I get communications from people even today that want to convert it for another language or another machine.

HAIGH: And that was the thing that automatically detects characteristics?

CODY: Yes, yes. It detected the machine characteristics. I'm proud that I was able to serve on the IEEE committee, I certainly was not a major player but I'm sure proud of the end result. I've got a lot of things I'm proud of.

HAIGH: If you had to pick one, to chisel on your gravestone or whatever, what you would most like to remembered for?

CODY: Family man.

HAIGH: Okay.

CODY: I never ever put the professional considerations ahead of the family. In fact I looked around me at Argonne, I was offered to be considered for division director at one time, told them I didn't want it. I looked around me at Argonne and saw so many of my colleagues that were divorced and to me that's sad. I turned down opportunities to speak, to travel to various places, if it conflicted with what I considered to be my primary concern, which was my family. At first I think people, including the management, were a little bit skeptical but after that I got respect. If I said no, there was no question about why. It wasn't a question of being afraid to travel or anything like that, it was just it didn't fit, it was the wrong thing to do. I think that's the thing I'm proudest of.

HAIGH: Well, thanks for participating in the interview.

CODY: You're welcome.