An interview with
Charles L. Lawson

Conducted by Thomas Haigh
On
6 and 7 November, 2004
San Clemente, California

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

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

ABSTRACT

Charles L. Lawson discusses his professional life and career with the Jet Propulsion Laboratory, with particular reference to his contributions to the field of mathematical software. Born in 1931, he trained as an optometrist at the University of California, Berkeley. However, after army service Lawson shifted his interests to mathematics, obtaining a M.Sc. (under Peter Henrici) and a Ph.D. (under Theodore Motzkin) from the University of California at Los Angeles. During his studies he worked at the Institute for Numerical Analysis at UCLA using the SWAC and then, following a summer internship, at NASA's Jet Propulsion Laboratory (JPL). From 1960 to 1996, Lawson served as an applied mathematician at JPL, specializing in the production of mathematical software and often working as an internal consultant to assist engineers and scientists with the computational problems. Lawson discusses the work of his Applied Mathematics group in some detail, including the contributions of his colleagues Paul R. Peabody, Richard J. Hanson and Fred T. Krogh. He discusses in some detail the challenges involved in producing ephemeredes needed for spacecraft missions and the state of celestial mechanics during the 1960s. He also recounts the development of computing at JPL, including the organization of computing facilities, the main machines in use, and the challenges posed by the arrival of minicomputers and workstations. Lawson was responsible for mathematical library development within JPL, including the lab's MATH77 Fortran library. Lawson, Hanson, Krogh, D.R. Kincaid and Jack Dongarra were responsible for the conception, design and implementation of the Basic Linear Algebra Subprograms (BLAS) which created a standard, machine-independent application interface to vector routines optimized for specific computer architectures. The BLAS made linear algebra software more efficient, portable and structured. BLAS development was conducted in conjunction with the LINPACK project undertaken by Argonne National Laboratory, and the two projects proved symbiotic. Outside JPL, Lawson was a well known member of the mathematical software community and discusses the general development of this field as well as his personal contributions. Individuals discussed include Cleve Moler, Brian Ford and Gene Golub. In 1968 Lawson became one of seven authors of the book *Computer Approximations*, one of the standard works on this topic throughout the 1970s. Lawson took part in the early Mathematical Software conferences organized by John Rice during the 1970s and was a charter member of the IFIP 2.5 Working Group on Mathematical Software. Lawson edited the ACM SIGNUM Newsletter from 1972 to 1976, and had previously edited the Scientific Applications Department of the Communications of the ACM. In collaboration with Hanson, Lawson wrote the book *Solving Least Squares Problems*, which appeared in 1974.

HAIGH: Dr. Lawson has been kind enough to write some quite extensive notes on his life and career and involvement in the field of numerical analysis software in general. In this interview, I'll frequently be referring to this. We will also try and avoid duplicating precisely the material that has already been documented there. The memoir should be included with the transcript itself.

To begin with, I'd to like to ask you to say a little about your family background and early education.

LAWSON: As I wrote up here, I was born in the small town of Weiser, Idaho, which is on the Snake River where Idaho and Oregon share a boundary. Population there is about 5,000. It's been about the same from 1900 to the present. My mother was also born in that small town, and she was an optometrist. My father grew up on a farm in Wilder, Idaho, which is about 60 miles south of Weiser and went to the University of Idaho and then came to Weiser to teach in the high school -- mathematics and business subjects. I went through the schools through my junior year of high school in Weiser, and then for my last year of high school I lived with my aunt in Hollywood, California. I completed my senior year of high school there. I attended the University of California at Berkeley.

HAIGH: I'll cut you off there and ask some follow-ups on that earlier part of your life. Growing up in a small, relatively remote community, do you think there's anything from that upbringing and experience that has affected the choices that you made later in your career?

LAWSON: I guess it's hard to say. Certainly, in a small community like that, people know each other and care for each other, so there's a certain feeling of looking out for other people when growing up in a small community like that. Beyond that, I don't know what I can say.

HAIGH: You mentioned that your father taught mathematics as well as business. Do you think that exposed you to mathematical ideas early in your life?

LAWSON: I think if there was any influence there, it's probably more hereditary than practice. Actually, my parents were separated when I was in about the sixth grade, so I didn't have very much contact with my father beyond that time until I reestablished contact with him when I was an adult. So I probably couldn't attribute too much math practice, but maybe something in the genes.

HAIGH: Did your interests in school as you were growing up focus on mathematics or science?

LAWSON: I guess I wouldn't say science particularly, but I always found mathematics easy and interesting in high school. In the first year of college, I was really impressed with calculus. That seemed like such a wonderful set of ideas and techniques. I did wonder if there was a possibility of a career in mathematics. Looking back on it, I guess I would have to say that I didn't inquire widely enough to find out what might be possible. Actually, one person I talked to who had been my high school science teacher said, "No, there isn't anything you can do in mathematics. It's all been done. It's all in tables." So I really did not pursue mathematics. Well, I did take a sophomore year of mathematics, which was not required in the optometry curriculum, just because I was interested in it. But I didn't pursue it beyond that at that time. Of course, there

wasn't much in the public press about computers at that time. It was 1948 when I entered the university, so I didn't know much about that for another year or two. There's a book that I mentioned in this write up that I read around 1949 that I think really did influence me to think that computers were going to be something important, and maybe I might get involved with them some day. [Edmund C. Berkeley, *Giant Brains, or Machines That Think*, J. Wiley & Sons, 1949].

HAIGH: You think you would have read that book as an undergraduate student?

LAWSON: Yes, I think probably around the time I was a sophomore in college.

HAIGH: Do you think it was something that students were reading widely in that period?

LAWSON: I doubt it. I don't know how I came across it, but I don't remember it being a topic of discussion or anything like that. It's just something that I somehow picked up and found very interesting.

HAIGH: Why was it that you moved to California?

LAWSON: In order to prepare better to attend the University of California. I must say that my mother had a rather strong influence on my development in many ways. She had more or less always said to me that it would be good if I prepared to go into optometry; that I could do other things if I wanted to, but unless I decided on something else I wanted to do, she thought I should do everything necessary to prepare for that direction. So as it turned out, I was probably the youngest person that every entered and completed the School of Optometry at the University of California Berkeley. Generally people would not have had all of the prerequisites as young as I had them, because we had had the catalog from the school of optometry from the time I was in junior high school. I'd been taking high school classes and my freshmen and sophomore college classes all with the idea being prepared to go into the College of Optometry.

HAIGH: So was Berkeley a school that was particularly known for its optometry program?

LAWSON: There were only ten places in the United States that you could study optometry at that time. That may still be true now; I don't know. Only three of those were on the West Coast. One was Forest Grove outside of Portland; one was connected to the University of California Berkeley; and one was the Los Angeles College of Optometry. So I was oriented towards Berkeley.

HAIGH: As you pursued optometry, was it a subject that you found interesting and motivating?

LAWSON: Not particularly. I wasn't particularly interested in it, and I guess my mind was always kind of searching for a different direction to go. I remember one incident when we were first doing some clinical work. As students we would examine a person's eyes and write down a prescription for a pair of glasses. Then one of the professors in the school of optometry would examine the same person and see what they came up with as a prescription and talk it over with me. I remember one case where the professor asked, "Why did you prescribe this for that person?" I said, "Well, that person seemed to have kind of the same problems with their vision as

I do, so I prescribed the same thing I'm wearing to them." The professor wasn't very pleased with that answer. In general, I was not enthusiastic about the program.

HAIGH: I think that you mentioned that you took a mathematics course that wasn't required. Did you have any other contact with mathematics or computing during your time at Berkeley?

LAWSON: No I didn't. I entered in '48 and completed the five-year program in June of '53. You would have had to have been at maybe MIT and working on Whirlwind to have had any actual contact with computers at that time in the United States. So I had no contact with computers. As far as math goes, I just took a couple of extra courses in my sophomore year but didn't really get acquainted with anybody in the Math Department or anything like that.

HAIGH: So shortly after completing your studies in optometry, you were drafted into the Army?

LAWSON: Yes. In the summer of 1953, I had finished the program in the School of Optometry in June, and I studied and took state board examinations in three states, primarily because my mother had licenses in those three states: Idaho, Oregon, and California. I was negotiating with my draft board all of time, asking them to hold off on drafting me so I could take these exams. Also, the people who had passed State Boards in optometry were eligible for direct commission into the military as a lieutenant, but the military was not obligated to take optometrists in as officers. They had some limited number, and as I understood later, the Army is organized into maybe six different districts in the United States. Each of those districts had a quota of how many people they could bring in as commissioned optometrists in a given time frame or given year. I think I made the mistake of applying in San Francisco where all of the other people in my optometry class were applying, so they had an oversupply of people applying. If I would have gone home to Idaho and applied there, I was told later that I probably might have been the only one applying. I might have gotten it, but I didn't understand that at the time. So I did not get a direct commission, and my draft board finally said, "Hey we've held off as long as we can. You're going in the Army." So I entered the Army in December of '53 as a Private, not doing anything connected with optometry.

HAIGH: Then a year later you were upgraded?

LAWSON: Yes, I applied again because I was told that priority was given to people who were already in the service who applied. I finally did get the direct commission about a year after I had entered the service. Accepting the commission required serving two years in that capacity, so I ended up serving a total of three years.

HAIGH: And you spent your time in the Army in the United States?

LAWSON: Yes, I was never outside of the United States. I had basic training when I was first drafted at Fort Ord, California and in Fort Sam Houston in San Antonio, Texas. Then when I got the direct commission, I was assigned to Fort Jackson, South Carolina and I was there for two years.

HAIGH: Was there anything during your time with the Army that changed the way you were thinking about your career?

LAWSON: I just became more definite that I wanted to do something else. I particularly wanted to do mathematics, and I was more aware that computers were coming on the scene by that time. So I did take two or three correspondence courses in junior and senior level mathematics from UCLA by correspondence during those two years that I was serving as an optometry officer. Then when I got out of the service, which was December of '56, we moved to the Los Angeles area and I enrolled at UCLA in January of '57 to study mathematics.

HAIGH: It seems from what you've written that you essentially spent the time from January of '57 until the fall getting up to speed with mathematics so that you could enter the graduate program.

LAWSON: That's right. They admitted me in what they called "special status." It was kind of between undergraduate and graduate, and I took four classes in upper division mathematics, one of which was an introduction to computers. I got A's in all of those, and then they admitted me to regular graduate status in the fall.

HAIGH: So it was in that introduction to computer course that you were able to use the SWAC [National Bureau of Standards Western Automatic Computer]?

LAWSON: That's right. The SWAC computer was the only computer available to us there and so the only computer I knew anything about and the one I got introduced to. I think that's nice. That was a way to get introduced to computers from the bit level up.

HAIGH: Can you remember who was teaching the course?

LAWSON: I think his name was [Fred] Hollander. I think he was an astronomer by training, but he was one who had gotten very interested in electronics and computers, so he was on the staff of the Institute for Numerical Analysis at UCLA, which operated the SWAC and which was the home for the SWAC. He basically maintained the SWAC and taught the introductory courses in the use of it.

HAIGH: Do you have a sense for roughly how many undergraduates around that time would have had that kind of opportunity to use the computer?

LAWSON: I don't think there was any exceptionally large interest in that. My recollection would be the class I was in was just a typical maybe 20 people or something like that; it was nothing exceptional. I don't think there was any widespread interest in computers at that time. People who felt an interest in it were excited about it, but it wasn't something that society was broadly thinking about at that time.

HAIGH: What kind of programs were people in the class learning to write?

LAWSON: It was so basic. I can't think of what we might have done as exercises. Basically we learned how to address storage locations in the machine and how to add the number in one storage location to the number in another storage location and how to move a block of 32 words between high-speed memory and the backup drum. So basically the techniques of using the computer. I'm sure we must have done some exercises of some kind, but I can't imagine that they would have been very involved. There was nothing like any subroutine library, there was no

operating system, there was no assembler—everything was done in terms of absolute addresses. In the course of a one term class, you weren't going to develop much of a computer program starting from scratch on that.

HAIGH: How was operation of the computer handled? Would students just have signed up for a block of time on the computer and then worked it themselves?

LAWSON: I think that's right. I'm really stretching my memory to remember that. I kind of remember some students that were ahead of me that were actually using the program in number theory type applications, I think generation of new larger Mersenne primes was one of the things that one or two of these students ahead of me were doing. I think people were using the computer into the wee hours of the night probably. We would sign up for an hour or a half hour or something like that. Of course it was not in any sense multi-user, so the person using it was the person using it; there wasn't somebody else doing something on it at the same time.

HAIGH: Can you remember anything about your first exposure to programming and using a computer? Was it something that you took to immediately?

LAWSON: It was the first two summers that I was back in Los Angeles, so it would have been the summer of '57 and the summer of '58 that I worked at Douglas Aircraft in Santa Monica in their computer programming department. They had an IBM 701 computer. That first summer I was put into an introductory course in numerical methods. A person there named Russ Mapes ran that, and there were probably 15 or 20 who were hired for the summer. Some of them, I guess, were intending to continue on as full-time employees, and others were just there for the summer, but we all went through this program. There were a series of problems to work and some standard computational methods, maybe interpolation or maybe solving systems of linear equations, that we were to do first on a desk with an electromechanical calculating machine, a Friden or Marchant probably, and then some of those we would also do on the 701. I know we did do work on the 701. I don't remember how much we did, because, again, it was a question of signing up for time. I don't remember. They probably had at least two 701s, but I'm not sure of that. There was a blackboard, and we were all in a big room with no partitions, just a bunch of desks in a big room. There was a blackboard so you could kind of see it from your desk. You'd sign up your name, and as people used the computer, they'd mark their name off. When your time came up, you'd go and run your program. So I don't think I would have had too much actual time on the computer because they were doing real work besides this training work. We did get some chance to use the computer in that first summer.

In the second summer, I was really assigned to help someone in doing maintenance work on some real programs. There I was using the computer more. But it was the same system, as I recall, of signing up on the blackboard.

HAIGH: So there wasn't a dedicated staff of operators who would actually feed the jobs into the computers for you?

LAWSON: I don't think so, but I could be wrong on that. There might have been someone that helped load a deck of cards, and maybe I would stand there and watch it run. I could kind of tell them by how the lights flashed or how the tapes moved if it looked like it was running okay or not. I don't remember how much actual help there was. I presume there must have been someone

there, because if the program went off the tracks, there probably needed to be someone to recover the computer and let the next person start using it.

HAIGH: Your recollection is that you'd be able to carry your own deck of cards up to the computer.

LAWSON: I think that's very likely the case.

HAIGH: I've just checked. I think Douglas had two 701s. The first one arrived in '53, and then in '54 they got another one.

LAWSON: So they might have had three when I was there.

HAIGH: Could be. They might have gotten one that another company had given back to IBM. It's possible. I know some of the machines were moved around as they came towards the end of their life.

LAWSON: Maybe they had three. They had one of them reserved for production work or something like that. Those of us developing programs might have only had access to one of them. I don't remember that much detail.

HAIGH: Can you remember anything about the permanent computing group that was there at Douglas at this point? Can you remember anything about how big it was, or what kind of jobs they would have been doing for the company?

LAWSON: I think we were pretty much all in this one big room, but there might have been maybe five columns of desks and eight rows. I don't know if there would have been 40, probably something around that magnitude, maybe less, people working as computer programmers. Of course, I, being just a summer employee, I didn't have much of an overview of the scope of the place. There may have been people working on computer programs in some other building that I didn't know anything about. I had a very favorable impression that it was well organized and well run. I remember in the second year when I was working on some actual applications, there was one that was called the Landing Performance Program, and I remember the magic number 50 feet—everything was keyed to the idea of the airplane being 50 feet above the ground when it came to the beginning of the runway. So then the computer program would simulate various uses of control surfaces and power settings and simulate what the airplane would do in the way of landing starting 50 feet above the runway. So this is a pretty obscure piece of information, but that's something that sticks in my head as one program that I was doing a little bit of work to help maintain.

Douglas was actually designing the DC8 at that time, and they were doing computations to simulate the performance of that aircraft. I think it was rolled out sometime in that second summer that I was there. It was built, I think, at Long Beach. Some of the people from Santa Monica went down to see the rollout. I don't think I did.

I also have a vague recollection, it must have been towards the end of that second summer, that Douglas San Monica was getting an IBM 704. That was first computer that Fortran was on; the computer that Fortran was really designed by Backus and his group to run. That sounded

miraculous to those of us who had had a year or two of association with computing programming to have someone tell us we could write, "A = B + C" and we didn't have to get any more detailed than that, and the program would convert that into a running program. That seemed really remarkable to us.

HAIGH: So do you remember what kinds of system tools were available for the 701 at Douglas?

LAWSON: I really don't remember any particular tools.

HAIGH: You give the impression in the memoir that they had some kind of assembler available.

LAWSON: Yes, the 701 definitely had an assembler, and all the work we did there was in assembly language. At that stage of development of computers, anything that we would now call a systems program or something like an assembler, some of them were built by the customer organizations. I think one of the earliest assembly programs on the IBM series of machines was the one they called SAP, Symbolic Assembly Program. Then when Fortran came out, they had FAP, Fortran Assembly Program, which was essentially the old SAP with new features so that you could write subroutines in the assembler that could be called from Fortran. So it added some features in the assembly program that would make it compatible with the way Fortran would call subroutines. But I think SAP and FAP were written by customer organizations. I think some of the aircraft companies wrote those. I don't think IBM originally wrote them, although they must have taken over the maintenance of them at an early stage. Of course, Fortran was an in-house development at IBM. But there was some kind of an IO system that I remember hearing of in those early days that was also developed at one of the large customer organizations. So systems as such were just not recognized as part of the package, or manufacturers like IBM certainly hadn't gotten around to doing anything with them.

HAIGH: Do you remember Douglas having any kind of mathematical library?

LAWSON: I don't remember anything. I would guess that for basic things like polynomial interpolation or solving systems of linear equations there must have been some standard routines, but I don't remember how they were organized or documented or how we knew to use them, other than maybe talk to one of the other employees there and ask them, "Is there a routine to do this?" It may have been more organized than that, and I just never was there long enough to know that.

HAIGH: In terms of really getting you interested in programming, you would say that the Douglas experience was more important than the slightly earlier exposure that you had to the SWAC computer?

LAWSON: Yes. Well, I don't know if more important is the right word, but it certainly was the next developmental step I needed. The work at UCLA on the SWAC was, I think, really valuable throughout my career to kind of know where the bits had to go, because a person never gets that now days. So I appreciate having had that experience. The work at Douglas began to show what a person had to do to really get some work out of a computer.

HAIGH: To take you back slightly chronologically, in the fall of 1957, you began your graduate mathematical work at UCLA?

LAWSON: Yes.

HAIGH: Did you initially find it difficult catching up with mathematics after not having so much of it previously?

LAWSON: Yes, I think I did feel a little bit behind some of the other students, say in advanced analysis class or advanced algebra class or topology class. Well, I wouldn't say so much in topology because the other students hadn't any of that either. I think pretty much every class that I took there in the graduate program was pretty new to me, and for some of the other students it may not have been quite that new. So I felt like I was having to work hard at it. But still, it was interesting, and it was within my capability to do. So I worked along and managed it.

HAIGH: Your Master's, you took from 1957 to '59?

LAWSON: Yes. Essentially, at the end of two years, I completed a Master's. That involved writing a paper and passing an examination.

HAIGH: Did you know in '57 already that you wanted to go on for your Ph.D.?

LAWSON: Well, I was hedging my bets. Again, I think because of my limited awareness of what the world of mathematics as a career was and as a subject matter what it was, I didn't have 100% confidence that I would be able to complete a Ph.D. So I felt I really should get a Master's degree in case that was as far as I could go.

HAIGH: Just to backtrack slightly, I think Douglas was one of the organizations that was involved in a joint project to produce something that they called the PACT Coding System. Does that name ring a bell?

LAWSON: No, I don't think I've heard that name.

HAIGH: You said that your supervisor for the Master's was Peter Henrici.

LAWSON: I think he's a Swiss, but the name may be of French origin.

HAIGH: Do you want to say anything about him?

LAWSON: He's a well-respected person in those years of Numerical Analysis. I unfortunately had little contact with him, even though I did my Master's work under him. I think he was busy with many other things, and having a Master's student wasn't a high priority with him. One fairly distinct recollection I have is stopping in the hall one day and saying that I'd like to make an appointment to see him, I wanted to talk over some things that I was working on with him whenever it was convenient for him. I remember him thinking for a little bit and saying, "I think eleven o'clock next Wednesday I'd have time that you could come and see me, and we could make an appointment for a time for you to come and talk with me." [laughs] So I never had a feeling that I was really in a lot of close communication with him.

HAIGH: Was your Ph.D. a continuation of the same topic that you had been studying for your Master's?

LAWSON: In the Master's degree, as far as a thesis paper, it was a little bit of an exploration of a somewhat obscure method for finding roots of a polynomial associated with the name of Routh. My Ph.D. work I thought of as being under the heading of approximation theory, which didn't have really too much connection with the Master's work, except the Master's work gave me the opportunity to do a little extra reading in general numerical analysis topics, but I wouldn't say it had a lot of connection with approximation. I thought of approximation in terms of fitting curves to data or something like that, primarily, and of course there's a lot more to approximation theory than that, so maybe my work on a polynomial root finding method had some connection, but not a lot.

HAIGH: And your Ph.D. was with Theodore Samuel Motzkin.

LAWSON: Motzkin, yes.

HAIGH: How would you describe that relationship?

LAWSON: Again, I'd say it was not real close, and maybe this is a description of me rather than of all these other people I'm talking about [chuckles]. But I certainly spent a good deal of time with him, whereas with Henrici, you could probably add up the total amount of time I spent with him, it wouldn't have added up to more than a couple of hours the whole time. Maybe that's an exaggeration, but it wasn't much. But with Professor Motzkin, we spent enough time to communicate with each other. His own life history, which is on a website that I reference in that document, I wasn't aware of at the time I worked with him. It's just now in the last couple of months, trying to look over my history, that I looked up that website and found out more about his history. His father was a Russian Jew who had moved from Russia to Berlin, and Professor Theodore Motzkin was born in Berlin. His father had been a math professor, and Theodore Motzkin was essentially raised to be a mathematician, and he had the ability, so he was something of, I don't know if a prodigy, certainly had a great facility for mathematics. Then he eventually came to the United States, particularly to the Institute for Numerical Analysis. He is a very soft-spoken person, he's not aggressive, he's not a person that makes a big effort to be sure everybody knows he's there and what he's doing, so you had to find out, or in my case, maybe I didn't find out very much what he was doing.

His thesis, I think, was fairly basic work that later became famous as the Simplex Algorithm, and is always associated with the name of Dantzig. But of course, others worked on linear inequalities. I think Professor Motzkin, if somebody would have come up to him with a linear programming problem and said, "What would be an algorithm for solving this?" would have written down the Simplex Algorithm. But he wasn't the person to get out and blow his horn or he wasn't in the right place at the right time or something, so he wasn't the one that got credit for that. But obviously, he must have been interested in computational things to have come to the Institute for Numerical Analysis from Europe. So that's a mystery to me. In my connections with him, I'd never got the feeling that he had a strong interest in developing algorithms that people would actually use. He really was a classical mathematician, and he enjoyed discovering relationships in mathematics, and he liked to do things in number theory. When any kind of new mathematical question arose while we were talking to each other, he'd go to the blackboard, and he'd start writing down some numbers. He'd say, "Well, let's see. What's true for one? What's true for two?" He liked to work things out and see relationships. But he didn't put anything together that made a great splash or impact. So I think it was a privilege to have had an

association with a man like that, but it was a little bit of an enigma to me how it was going to relate to what I wanted to do. But we did work through it all.

HAIGH: Right. Now you said that one of the parts for your dissertation was minimax approximation via weighted least squares. So was that the beginning of your interest in least squares?

LAWSON: I guess I would have to say it probably was, although for the purpose of what I was looking at, I kind of took the idea of least squares computation as a given, so I wasn't looking into any different ways to do least squares computation. The standard approaches involve normal equations, and I wasn't thinking of anything beyond that at that time, so that was just a given tool. The thing I was looking at was finding an algorithm by which one could do a least squares approximation to a discrete set of data, compute the residuals at each point, use those residuals to come up with weights, then come back and do a weighted least squares approximation that would be closer to being a minimax approximation, and keep doing that iteratively and have it converge to a minimax approximation. And it's kind of obvious that what you want to do is drive the large errors down, and the typical characterization theorem is that if you're fitting family has n + 1 degrees of freedom, then there's going to be n + 2 points at which the residual curve hits its maximum. You need to be a little more careful in stating things than that, but that's the general idea. And so the obvious thing is to try to put weights into the least squares fit that will drive the large errors down, but you can try different things and you can find things that will drag them down too much, and so it's a matter of finding what worked, and then proving that there was a reason why it worked, and that's really what that part of my dissertation amounted to.

[Tape 1 of 4, Side B]

HAIGH: Was there any aspect of your dissertation which involved working with an electronic computer?

LAWSON: Yes, yes, yes, but let's get the time sequence in here. After the two years of graduate work at UCLA, I had finished the Master's degree, so it was sometime, I suppose, going into the next fall, that I sought out Professor Motzkin and asked him if he would be my Ph.D. advisor. I don't remember exactly when I did that, but it would have been about that time, or maybe later, because there were qualifying exams to get through and things like that. So in that third graduate year, I was really concentrating on preparing for qualifying examinations, and there were even language requirements in those days. I had to show some proficiency in reading mathematical papers in one or two foreign languages, French and German is what I would have chosen, but that was a challenge for me because I had never had too much facility in languages. So I was doing those things in my third graduate year, and I think I was starting to do a little special reading in approximation theory, probably under the direction of Professor Motzkin. But at the end of that third graduate year, I completed all my requirements for the Ph.D. except the dissertation, and I completed the examinations, and so I started regular employment at JPL the following fall after the three full years at UCLA, and so that first year at JPL was also my last year at UCLA. I was working on my dissertation and going, I think weekly, probably, driving back to UCLA to discuss things with Professor Motzkin. So my access to computers at that time was the computers at JPL, which were quite adequate to do things. That was the IBM 7094, which had built-in floating-point arithmetic and it had a Fortran system on it, so I could write programs that would allow me to experiment with these things and the weighted least squares.

HAIGH: So you were using it to experiment with the thesis material?

LAWSON: Right.

HAIGH: Did anything directly related to the computer implementation make its way into the thesis, or did you just kind of leave all that out?

LAWSON: Well, the thesis does report some examples of computing using this method, yes.

HAIGH: What was the place of numerical analysis in the UCLA curriculum at this point?

LAWSON: I think numerical analysis was unfortunately always kind of a poor stepchild of the Math Department. There were individuals at UCLA that were strong in numerical analysis at different times, but I have the impression that that was not recognized as the way to get ahead in the Math Department, and particularly with the Institute for Numerical Analysis falling into harder times in terms of financial support during the time that I was there, any professor who was strongly interested in numerical analysis would probably want to go somewhere else. And in particular, I think George Forsythe was on the faculty in the Math Department at UCLA in the middle ‘50s, and then obviously he went up to Stanford University from UCLA, and eventually, in the middle ‘60s, founded the Computer Science Department up there. There was a saying that I’ve heard over the years, that before the computers, numerical analysts used to be the people that didn’t count for much in math departments, and after the computer field developed more, the numerical analysis people were the ones that didn’t count for much in computers science departments. So I guess numerical analysts in the academic world have maybe always had a reason to feel a little bit put upon, but since I wasn’t pursuing an academic career, I have never gotten into that problem, so I don’t know first hand whether that’s a big problem or not.

HAIGH: And you mentioned holding an assistantship at UCLA on an introductory numerical analysis course.

LAWSON: Yes, that was under Tom Southard, and that was my introductory class on numerical analysis as well as being a teaching assistant, which I guess is not uncommon for a teaching assistant to be learning what they’re helping the professor teach. Southard was very straightforward and just followed the book without embellishing it, is my recollection, and that made it easy for me to follow because I could read the book and that backed up what he said in the lectures, and then I could help the students. I certainly was able to pick it up fast enough to be able to help the students. So that was my kind of textbook introduction to numerical analysis after I’d had the other kind of cookbook approach at Douglas in that summer program. So that was an experience.

HAIGH: And you also mentioned having a research assistant position with the Institute for Numerical Analysis.

LAWSON: Yes, I guess that must have been probably in my third graduate year that I had a research assistantship.

HAIGH: And who were you working with there?

LAWSON: At some point along the way there, I had connected with Professor Motzkin to be my Ph.D. advisor. Professor Charles B. Tompkins was the head of the Institute for Numerical Analysis at that time, and so I was nominally under him from a paycheck point of view, but I don't think he was particularly directing my work. So basically, being a research assistant, I was exploring things that eventually fed into my dissertation.

HAIGH: So you were fairly free to get on with things that you were interested in?

LAWSON: Yes, I don't think there were any particular duties that I had to perform there. We may have helped newer students get acquainted with the SWAC or something like that, but nothing that I remember particularly.

HAIGH: Do you remember anything about the general atmosphere there at that point?

LAWSON: In the Institute for Numerical Analysis, quite informal. Everybody was doing their own thing. There weren't many senior people there I think because of the drop in funding. The other students that I was aware of seemed to be mainly working on number theory type things, so I don't remember having any kind of group association with anybody else in connection with anything that was going on there. That institute, of course, had been a real hotbed of development in Numerical Analysis methods maybe five years earlier with people like Stiefel and Hestenes and the development of the conjugate gradient method and a lot of exciting things, but that time had passed.

HAIGH: Then it was really winding down by 1958.

LAWSON: Yes, I would say so. And there was no prospect or any talk, even, of replacing the SWAC with something new, and of course it was quite obsolete at that time, but as I say, it was a wonderful machine to get an introduction to computing on.

HAIGH: Now, you've said that you went to JPL while you were still working on your dissertation. Did you ever look around and think about what you wanted to do next and where you might go?

LAWSON: I only remember one other place that I interviewed besides JPL at that time, and that was Prudential Life Insurance Company in Los Angeles. They were looking for people that wanted to follow an actuarial career. I didn't know anything about actuarial work in particular. One of the things I'd done as a teaching assistant at UCLA was to teach a beginning class in the math of finance, so I knew a little bit about just the basic formalities of some of their computations. But the main pitch that the person had that interviewed me was that, "Even though our starting salaries are lower than in the aerospace industry, the president of our company started out that way, so you should realize that somebody starting with a Ph.D. in math and actuarial work here might become the president someday." So I balanced the probability that I would become the president of Prudential someday against a little higher salary and a little more obviously immediately interesting math and computing environment at JPL, and JPL came out on top. Otherwise, I might be president of Prudential Life Insurance today, but I didn't go that way.

HAIGH: And did JPL have a high profile in the scientific and mathematical community at that point?

LAWSON: Well, there were a number of major aerospace companies in the Los Angeles area that I had some awareness of, STL and Aerospace Corporation and Douglas and JPL and Aerojet, and those seemed to me to be the places that were making challenging uses of computers, and I think that's what I wanted to be involved with. And JPL had the added attraction that it was not a Defense Department operation, and that appealed to me too.

HAIGH: Why did that appeal to you?

LAWSON: I thought things would be more open and there'd be more chance to interact with other people more freely. I like that idea.

HAIGH: And you joined the Applied Mathematics group?

LAWSON: Right.

HAIGH: What was the place of the Applied Mathematics group within the lab as a whole?

LAWSON: In those days, and we're talking 1960, large institutions would usually have a computing center, and they'd have one or two, maybe more, large mainframe computers, and that would be it aside from the business data processing. But for science and engineering, there'd be a mainframe computer, and that would be programmed and operated by some organization whose job it was to program and operate that, and scientists and engineers that needed something done on the computer would not be writing their own programs, and maybe not even running them. They would be working through some computer organization. So there was such an organization at JPL, and JPL at that time was organized managerially into divisions as the top level breakdown. Then there were sections within divisions and groups within sections. A group was typically the order of 10 people, more or less, and a section was maybe 50 or 100 people, and so this group was in a section that was certainly doing the computing programming. I don't remember whether it was responsible for operating the computers or whether that was a divisional responsibility and there was some other section or division that did that. But essentially everybody at JPL that was doing computer programming was in this section, and this group was supposed to be kind of the mathematical brains of the section, so if the computer programmer was asked to do something and it was beyond what they were acquainted with in terms of mathematical algorithms, they could talk to our group and we were supposed to be able to help them with ideas on how to approach it. So we provided consultation to programmers within the computing organization, but we also provided computational consultation to engineers and scientists anywhere in the laboratory that knew about us and wanted to make the effort to get in touch with us, and of course we did try to encourage them to do that, because we liked to be useful.

HAIGH: So primarily you were working as mathematical consultants for the in-house programming team?

LAWSON: Right. But we were given quite a bit of freedom to do things on our own, so we were doing some development of algorithms and software that we thought might have some general

use around the laboratory, but there wasn't anybody that had come to us and asked us for it maybe right at that point.

HAIGH: Did the basic role of the group change significantly over the time that you were part of it?

LAWSON: Over 36 years, there certainly were many changes. I don't know quite where to start on that. The way computing was done and who did it certainly changed, particularly say, after 1970, certainly after 1975. The smaller computers were more affordable, and so the science and engineering organizations around the lab could get their own computers, and so they didn't have to funnel all the work through this computer organization. And of course, eventually it got to the point where they weren't funneling any work through this computer organization, so the computer organization got broken down where different divisions would maybe hire three programmers out of this section that used to do most of their work. Now they'd become members of that division, and they'd be working on the computer in that division, so the whole approach to developing computer programs and using computers was very dispersed around the laboratory. So we had to make more of an effort to be sure we were communicating with engineers and scientists all around the laboratory and not just programmers within our section, because eventually there weren't any programmers in our section.

HAIGH: All right. So you remained the mathematical consultants, but later on the people you were consulting with were more likely to be the end users than the in-house programming team.

LAWSON: Yes, I think that would be a way to say it.

LAWSON: One thing I might mention about this question of change over the years, I felt like our group was kind of a fixed point in all the change that did take place, because the name and the function of the section we were in, and the name and the function of the division we were in, and the building that we had our offices in changed a number of times over the years. But our group continued to have this name, Applied Mathematics, and we continued to have some of the same core people and function in essentially the same way, at least up until about 1990. So I'd say from 1960 to 1990, the group existed and provided essentially the same kind of function, and had the same name, but the names of the higher-up organizations that we were in changed, reflecting, of course, first of all, the fact that computing was not a central thing. There were a lot of programmers after a certain point, and as time went on there were other reasons why the lab changed their higher level organization structure, but we continued to function.

HAIGH: So prior to 1990, was there ever a need for the group to justify its importance?

LAWSON: Yes, we were asked to do that certainly in some sense every year; maybe more than that sometimes. It depended on what the management structure was at the time and what the needs of the laboratory were at the time just what it took to make that case. I guess all I can say is we were fortunate to be able to make the case, at least up until about 1990. Just how we did it may have been different at different times. It may have been easier or harder at different times depending on what the financial state of the lab was, who we were involved with, and what the trends were. The lab went through various changes as any big organization does of wanting more accountability or some other change in management procedures, and we would have to adjust to some of those things on the way.

I might say that other math groups that I was aware of did disappear through the ‘70s and the ‘80s. We were in touch with math groups at STL and Aerospace in the earlier days I was at JPL. Along the way, they got cut off. The general trend was most organizations wanted to have everybody in some kind of product oriented organization, and the idea of a math consulting operation was usually not the first thing the director of a laboratory would think of. In our case, we were able to survive quite a while, and most of the other large engineering organizations I was aware of that had math groups in the ‘60s lost them by the end of ‘70s or ‘80s, I think.

HAIGH: Are you talking there about private companies particularly? Or would that include groups within government labs?

LAWSON: That would include government laboratories, yes.

HAIGH: Did all the members of your group have Ph.D.s?

LAWSON: No, not all. Fred Krogh and Dick Hanson did, and they were key people. Well Fred, right up to the time I left and Dick Hanson was a key person while he was there. We had others with a Ph.D.: Ed Ng and John Radbill. For shorter periods of time, some people who were specialists in celestial mechanics in the ‘60s, but we also had people who didn’t have their Ph.D.s that made contributions of importance, too.

HAIGH: You’ve written that in 1965 you replaced Bob Peabody as supervisor of the group.

LAWSON: Yes.

HAIGH: What kind of responsibilities did that involve?

LAWSON: It meant interfacing with the next level of management, the section manager, to be sure that that manager had awareness of what we were doing and that we were on a track that seemed useful to him so he could report that to his managers. Otherwise, we were not doing anything dramatic. We weren’t asking for large amounts of new equipment or large increases in the number of people, or doing anything that made us a special problem to the manager. We did increase our size. It was about five people when I came in in 1960, and probably from about 1970 to 1980 we were more like eight people or something, I would guess. Really, someone would have to look at the records to be sure of that. So we did have some increases. Sometimes we were able to do that because of some special source of money we’d get, but I don't know how we did it, but somehow we did it.

HAIGH: You must have liked being there. Did you ever consider any kind of career switch away from JPL?

LAWSON: No, not really. At the time that Sam Conte went to Purdue because he had been either at STL or Aerospace Corporation Los Angeles, and he left there and went to Purdue to be head of their computer science department. I think it may have been a new computer science department at that time. It would have been sometime between ’65 and ’70. He contacted me, and wanted to recruit me to go there, and of course John Rice went there. It would have been exciting to be the same place as John Rice, but I was comfortable where I was, and I didn’t have any particular image in my mind of why an academic position would be better than where I was.

The funding seemed more hodge-podge, because at JPL I had a straight salary and I knew what it was. The academic proposal, you'd get a certain amount of money, but you'd have to be applying for grants all the time to make it up to what you really wanted. That didn't appeal to me.

HAIGH: Was the JPL position better paid than a typical academic job would have been at this time?

LAWSON: I think it probably was. It was probably better paid as an entry position when I started. I think it was probably better paid than a university position would have been. A university professor that writes a lot of textbooks that are successful and gets a lot of grants certainly makes a lot more money. But for just a straight base salary, I think it was probably better.

HAIGH: The main specific project you talk about in your memoir, other than the libraries and tools, concerned celestial mechanics.

LAWSON: That was a very central thing at the time I came to the group, 1960, and continued to be through about 1966 or so. As I say in the memoir, the field of celestial mechanics had gotten into quite a doldrum. At that time, there was a feeling, I think, that the problems that needed to be solved had been solved. Now I probably never got deep enough into the field to be qualified to say things like that, but that was the impression I got. Certainly I think enrollment of students in that field was very small, and the number of places around that country that were regarded as strong in that area seemed to be small as we made inquiries because we were going to have to have some people who had expertise in that field in order to carry out a space program, and it seemed like there weren't very many places that were strong in that field in 1960. And Yale did seem to be a particularly strong place. They had an institute for celestial mechanics; it may not have been exactly that name. It had a couple of senior professors, Dirk Brower and Gerald M. Clemence, and there's a famous celestial mechanics textbook written by them (Methods of Celestial Mechanics, 1961). At least in those days, if you ever talked to an astronomer-type about the Brower and Clemence book, they knew what you were talking about. And the lab got those people to be consultants to us because we needed more know-how. (Currently there is a "Dirk Brower" prize in Astronomy awarded periodically by Yale University.)

The specific part of it that this applied math group was involved with was the question of could the accuracy of the ephemeredes be improved. (An ephemeris is a table of the positions of some celestial object.) Typical things that we would have been interested in to begin with were more the sun and the moon. It's interesting that when one goes through high school and learns about astronomy, one is told that it used to be believed that the sun went around the Earth, but now it's believed that the Earth goes around the sun. When you get into the Nautical Almanac produced by the Naval Observatory, you will find a table of where the sun is going around the Earth because the person navigating a ship or an airplane isn't sitting on the sun looking on the Earth; he's sitting on the Earth looking at the sun. And so the question is where is the sun, and where's the moon, and where's Venus.

The very first thing that the lab tried to do that needed that information was just an experiment really. Can we send a radar signal to the moon, which is our nearest celestial neighbor, and have it bounce back and detect it? You don't have any transponder on the moon—you're just trying to

do a bounce. They tried that with the information from the Naval Observatory's Nautical Almanac that says where the moon is, and they didn't have much luck in receiving a signal. I'm not by any means an electronic engineer, or any other kind of engineer, but my understanding is that the electronics system that was used for this is a vernier type of thing where you are not just measuring the distance flat out, but you're saying I know the distance within a small error, and I want to measure that small error, so I'm looking for a signal that's some small distance away from this standard distance that I think that thing is at. Well, it didn't get any signal because it wasn't within that small distance of where the Naval Observatory said it was going to be. So you start looking into that. Well, the Nautical Almanac published by the Naval Observatory at that time, and I don't know what the state of affairs is now, but I think was good to about five decimal places. It was an accuracy that had been found adequate over many decades for people to navigate ships and airplanes. That was what it was meant for—it wasn't meant for sending a radar signal and knowing where the moon is within a few meters. JPL needed to know where the moon was to maybe nine decimal places. The question that was being worked on in this applied math group when I joined it was: what can we do to get this information more accurate? Bob Peabody had the idea of doing a numerical integration of the Newtonian Equations of Motion of the bodies involved, which is mainly the Earth and the moon and the sun, and maybe it had to take Mercury and Venus and Mars into account as perturbing objects, I don't know. But do a straight numerical integration of the Newtonian equations of motion, and do a numerical fit of that to the Nautical Almanac information. So you were taking the Nautical Almanac as though it were a set of observations with errors in them, and you were doing this numerical integration. If you were just trying to determine one body, say the moon, you've got really six initial conditions. You've got three second order differential equations, so a position and the velocity in x, y and z is six numbers. So any set of six numbers you specify will give you a trajectory for the moon, and you're trying to adjust those six parameters to make the trajectory you compute come as close as possible to the Nautical Almanac numbers. To a numerical analyst, that just seems a very straightforward idea. The techniques were pretty well known. The Naval Observatory had never done it that way because their approach had been developed in 1900 when you couldn't think of doing a numerical integration of all that stuff and you had to work out big symbolic series representations of all these things. They had done that. When you do that, you say how many terms do I need? They had figured out how many terms they needed to get the accuracy that they actually really wanted. They had continued to use that same set of formulas since 1900.

HAIGH: Had they moved the work over to computers by that point?

LAWSON: They had computers, but they were modest compared to the 7094s that we were using. They were using something such as the IBM 1401, if that was in existence at that time.

HAIGH: The 1401, I believe, was announced in 1959, shipped probably the next year, and was pretty widespread by about '62. Before that the 650 was the leading small machine.

LAWSON: This is the time we're talking about. They had either those machines or machines of that caliber so they did not have what we would call a mainframe at that time. They had no motivation to do anything different from what they had been doing because what they had been doing worked. I'm certainly not being critical of them. They were doing what they were supposed to do, and doing it fine. But we had a new requirement, and it wasn't going to be solved by going back to those symbolic equations of 1900 and trying to add some more terms

onto it. I credit this idea to Bob Peabody. Any other sophisticated numerical analyst might very well, if presented with that problem, come up with the same idea, but anyway he's the one that did. So he had a couple of programmers, either in the applied math group or in another group that he was advising, that were developing software to do this. That was the state of affairs when I came in. That coding, by the way, was being done in assembly language at the time I came in, it had been going on for a little while, and Fortran was just coming into people's consciousness at JPL about the same time I came there. I can remember for some people, it was a very new idea, but I'd already had one year experience with it. I felt like I knew about it.

HAIGH: Was that example unusual because of the high degree of involvement you had with the application?

LAWSON: I think you could say that in the course of my career there, I was not usually heavily involved in a particular application. I was more involved with developing general purpose software and helping people pick what general purpose software would be good for their project. There were some times when I got a little more involved, but I would say this involvement in producing ephemeredes was much more project type work than I was usually involved with after that time.

HAIGH: So were there any similar examples you can recall from any of the later projects that JPL was responsible for?

LAWSON: Nothing that was that central to the whole lab and that I was that much involved with. This business of having accurate information of where the moon and planets are was certainly central, more than just this first experiment of doing radar bounces off the moon and Venus. But of course all the business of sending spacecraft out to other planets needed this. So it was really central to what the lab was intending to do. I don't think I've ever had that much direct involvement with something that was that central to the lab since then.

HAIGH: So in other cases, the relationship was more indirect? You would produce libraries and advise programmers, but you wouldn't be as exposed to the actual content of the application?

LAWSON: Yes. There were a few times when I would take on something and maybe work on it for three months maybe to clarify something for some group or to formulate something for some group, but I would say those were just different projects around the lab. They weren't central to the mission of laboratory.

HAIGH: There's a couple of colleagues that you talk about particularly who joined the group in the mid to late 1960s, Richard J. Hanson and Fred T. Krogh. Perhaps you could describe that previous experience and say a little about them as people and colleagues.

LAWSON: Both of them, I think, had undergraduate work at what is now Oregon State University, which was probably Oregon State College at that time, at Corvallis. They were pretty much contemporary. I don't know who got through first. Richard Hanson went to the University of Wisconsin for graduate work and got a Ph.D. under Wasow, doing a dissertation in turning point theory for ordinary differential equations. Then he had a faculty position at the University of Southern California in Los Angeles. Then he was interested in getting out of the academic

world and wanted to get more into real applications and talked to me at JPL. We were happy to hire him. So that was what he had behind him when he came to us.

Fred Krogh got his Ph.D. at Corvallis, and his advisor was William Edmund Milne, who was well known particularly for the "Milne" method of numerical solution of ordinary differential equations, and as the author of early textbooks on that general topic.[1]

Anyway, Fred had worked either at Aerospace or STL over in the Redondo Beach area near the Los Angeles airport. I'm not too sure why he wanted to move to JPL. I think he wanted more opportunity to do research. I think he felt he was being pinched too much to just do project things and he really wanted to do research type things. He had a research paper submitted and accepted to the IFIP world congress of the year that he came to us, which was 1968 maybe. Krogh reminded me of this a few weeks ago because I talked to him about this a little bit. He had submitted a research paper on numerical solution of ordinary differential equations. The IFIP World Congress meets only every three years, and he had submitted a paper and was accepted to present it there. One of the first things he asked for when he came to JPL was to have a trip to go give that talk, and I think it was not approved. He reminded me of that a few weeks ago. He decided to go on his own so he took some vacation time very shortly after joining and he went and gave that talk. But he was already very actively interested in presenting new ideas in the numerical solution of ordinary differential equations, and in being in touch with the international community on that topic.[2]

He was a very strong member of our group through all the years. A very bright person and very creative, and tends to write very sophisticated computer programs. It was really great for us to have him through all those years and work with him. He retired about two years after I did. I retired in 1996 and he retired around '98, maybe '99. He still lives not far from JPL and does consulting for JPL. At the time he left, there was really nobody… Well, the lab didn't have interest in keeping the MATH77 Library under their own maintenance, and they agreed with Fred Krogh that it could be his and he could do with it what he wanted to. He does still support its usage at JPL, but he also has his own website, *mathalacarte.com*, so a person can go to his website and they can order individual routines from that library if they want to. So that's kind of the ultimate place that the MATH77 library has ended up since about 1999.

[Tape 2 of 4, Side A]

LAWSON: So Richard Hanson was this very energetic, productive person, and really wanted to get a little more involved in applications than our group was, and he formed some close relationships with people in what was called the Navigation Group, navigation there meaning the control of the spacecraft when it's in flight to give it a little thrust this way or that way to make sure that it gets to the place you want it to get to. It's a very sophisticated set of mathematical techniques that those people have developed, and of course, a part of it is least squares fitting of computed trajectories, abstractly or in general, the same kind of thing we were doing with

[1] Lawson notes that details on Milne are available at http://osulibrary.oregonstate.edu/archives/archive/mss/milne_des.html, according to which "headed the Mathematics Department at Oregon State College from 1932 until 1955."
[2] Lawson notes: For Krogh's listing in the program of the 1968 IFIP Congress in Edinburg, see: http://www.informatik.uni-trier.de/~ley/db/conf/ifip/ifip1968-1.html

ephemeredes in 1960. You pick initial conditions for a spacecraft and you numerically integrate its motion as perturbed by the affecting planets and the moon, and you see where it ends up, and if it isn't where you want it then you make a correction that you work out by least squares to the initial conditions that will get it to come closer to where you want it to be.

HAIGH: So, in that case, least squares is a way of seeing how far the course you've calculated would depart from where you wanted to go?

LAWSON: Exactly. And the people doing that work had refined and developed it a great deal from 1960 on. We were coming into it around 1965, '66, '67, but as I mentioned in that write-up, Dick Hanson and I both were very much taken by two papers that Gene Golub was a co-author on around 1965, one on using Householder methods and least-squares with Businger, who was a student of Golub's I believe, and then the one that Golub wrote with Kahan on the singular value analysis and singular value decomposition. We could see this as having good application in this navigation and orbit determination area, and Dick Hanson particularly jumped in and communicated very closely with the engineers that were doing that work and figured out what would help them, what they needed, and what we could contribute to them. He developed software that they started using, and they found it was more stable and more reliable than what they had been doing before, so that was good—what can I say? [chuckles] So it was an important contribution to a central area of the lab's work, and of course it also got Dick Hanson and me digging more into how to organize a least squares computation using these orthogonal methods.

HAIGH: You mention that the group dissolved around 1990. Why was that, and what happened to you after that and before you retired?

LAWSON: It was really a lack of financial resources, a lack of budget to support the kind of function that we had been providing. And I guess I would say a couple of things. One, as I said before, the lab changed its management philosophy to some extent at various times, sometimes gradually and sometimes rather abruptly. There were some times in the '80s where they got very excited about some particular specialist in management methods who had written some book and everybody was going to try to do things that way, and they reorganized a lot of things. But anyway, just a combination of gradual changes and some abrupt changes. In funding, we just no longer had a home. If somebody were setting out to design Jet Propulsion Laboratory, they might forget that it would be good to have a math group. The mathematicians have to come forth and say what we can do and why you should have us there. We had been fortunate over many years that there were always some people in higher management positions that had either had some experience with the usefulness of our group when they were in a lower level or, for one reason or another understood that we were useful, and we got to a point where there wasn't anybody that really understood why we were useful, and we didn't make a strong enough case for it, and so they didn't need us anymore.

HAIGH: So what was it, that money was tight at JPL in general, or was it just that they philosophically didn't think that they needed the group anymore?

LAWSON: I don't think it was specifically that money was tight, because our group was never more than about eight people, around the five to eight level.

HAIGH: And how big was the lab, as a whole?

LAWSON: 4,000 to 6,000, that range, at various times. Sometimes they would have a lot of people that worked as full-time and long-term contractors, so besides the direct employees, there were always a lot of long-term contractors which essentially functioned like direct employees. So it was in the 4,000 to 6,000 person range most of the time.

It was not an overall lack of funds. It was a change in organization, and I would have to say we didn't make a big enough case for ourselves, so I have to blame myself as well as the changing management scene. The lab got a Cray T3D Supercomputer around late 1980s, and they formed a little group to provide consultation on use of that, and they attached us to that because there wasn't anything else to attach us to at that time, and then pretty soon they decided they didn't really need very many people attached to that either, so we kind of disappeared. So we went through a little bit of a phase of being combined with some other groups, and then finally we just disappeared. The one of our key people that's still there at the Lab, William Van Snyder, is working at a science-oriented group as their computer specialist, but the rest of us are all gone. But of course I retired basically on my 65th birthday, and I was comfortable retiring at that point, because I wasn't wanting to work 40-hour weeks any more at that point.

HAIGH: Through the 1960s, at the same time that you were beginning work at JPL, you were also working in the area of computer approximation, publishing papers and eventually a book. So I wonder if you could talk about how your interest in that area developed, and what you think were your most significant contributions.

LAWSON: Of course, that was the topic that I did my Ph.D. dissertation on, and it just seemed like an area I could get a grip on. Somebody had some data, and they wanted to get a computable representation of that data. If it was *y* as a function of *x*, then maybe we want a function where *y* is a function of *x*. Or if it's *y* as a function of *x* and *z*, then we want a function of two variables to represent it. And that seemed like an easy enough problem to visualize, and at JPL I saw this kind of problem cropping up in different guises frequently, and so I felt things I did in that area could be useful. Some of the things I did were aimed at specific things at a given time and others were more general with the thought that they might be useful. There was a lot of interest in the general numerical analysis literature in spline curves for interpolation and for approximations. Particularly John Rice had published some interesting things on that, and that seemed like an interesting thing to develop a little more, and I was thinking both in terms of algorithms and in terms of software.

The things I did in the area are rather miscellaneous. There's no great thread that finally adds up to some big thing. It was a lot of different, rather modest things in different directions that I actually wrote papers on or wrote computer programs for. What Dick Hanson and I called the French Curve Program, and he did most of the work on this originally and then I added some things to it later, was to try and get into a computer program something like you would do if somebody gave you a bunch of points and you drew a curve through it. So we took Spline curves as our basis. Eventually we allowed the splines to be of different orders, so it could be a low-order spline for part of the curve and a higher-order spline another place in the curve. And we allowed constraints which could be equality or inequality, and they could apply to the value or to the derivative, so you could say, "I want the value to be exactly this at a certain point," or you could say, "I want it to be greater than or equal to this at a certain point," or you could say, "I want the slope to be positive here," or, "I want the slope to be bigger than two here," and you could put constraints like that any place along the curve. So visually, it just meant if you had

tried to do a curve fit with some conventional computer program and you didn't like the looks of it, but if you could say what was wrong with it, if you could say, "Well I really want this to be higher here," or, "I don't want this to turn down here," you could express that to the program, and it would get you a fit that would satisfy those constraints. So that was an interesting thing, and it combined ideas from approximation theory involving spline curves and linear algebra, and really getting into what would be called linear programming or mathematical programming with the inequalities. So that was something Dick Hanson and I did, say, between 1965 and '70, and then I added some more things after he left the Laboratory that we could put in constraints on area, so if you wanted the curve to have a certain area, you could do that. This has some meaning in probability, because in probability, you sometimes want a curve that has an area of 1, and you could have that constraint.

HAIGH: So that was a specific suite of software that you and Hanson developed together?

LAWSON: It really came down to essentially one computer program.

HAIGH: Did it have a name?

LAWSON: We called it French Curve, but I don't think we ever published anything on it. It's in our MATH77 library, and that's about all we did with it. I don't think we ever published it.

HAIGH: Now, I would imagine this area of curve fitting would be something where the creation of powerful digital computers would open the area up enormously compared with what had been possible before.

LAWSON: Yes, in that you can do things like this with constraints that are going to put a little more burden on the computer than without the constraints, that's for sure. Usually, these problems are not terribly computing-intensive, though, so I don't say they ranked with the programs that really stress a computer. It was more a matter of combining ideas in a way that a user could bring a lot of things to bear at once that would have been separate steps or would have been hard to combine if we hadn't combined them. But that's just one thing that we did.

HAIGH: Did the computer expose a need for new mathematical methods, or was it just a case of taking these methods that had been invented previously and combining them in new ways?

LAWSON: I don't know what to say on that. I mean, the whole idea of a B-spline, and I think the "B" stands for the Frenchman Pierre Bézier (1910-1999) whose name I mentioned in my write-up in another connection. I think that probably only came up since the invention of computers. I don't think people had worked with that idea theoretically, so it wasn't an idea that would have occurred to people to be interested in, although there is a lot of interesting theory in it. Of course, Carl de Boor and others have developed a lot of theory associated with these, but the idea of even thinking about them probably only came after computers were available. Because there's enough of a boost of the computing resources needed over what you can do by hand that people wouldn't have done much with them before they had computers, but it's not a high-cost operation like solving a big eigenvalue problem or something like that.

But also in the approximation area, I was interested in two-dimensional things and things on the surface of a sphere, and building triangular grids. Nowadays there's tons of things on building

triangular grids. There's conferences and there's books and everything, but in 1960 there was very, very little, and I did some things developing contour plotting. Contour plotting always seemed interesting to me because if somebody gave you a rectangular grid of numbers or you had a rectangular grid with a value associated with each cross-point in the grid, and said, "Draw me a line through here that represents a contour value of 10," well, you'd look around, and if you find a value of 15 and a value of 5 next to each other, then you know 10 is in between, so you'd start there, and then you'd work your way through the grid, and it would be easy to trace out a curve. But to do that on the computer took a little of what would now be called computer science. You had to keep track of some data structures, and that hadn't been done very much in numerical analysis at that time, and I found it challenging. It's pretty elementary, but since I hadn't done it before and nobody told me how to do it, it was interesting. And then people get into ambiguities because if you're trying to follow this line of 10 and you go through a point, you have a point of value 10, then what do you do there? Some people thought that was just really complicated, and I didn't think it was that complicated, so I put together some programs that didn't get confused by that, but a lot of people just threw code together and did get confused by that. So I tried to do things that wouldn't get thrown off by special cases.

But also, we did some cross products of B-splines to fit two-dimensional things, and we added some constraint capabilities to that, and then I developed ways of dealing with basis functions for doing continuous interpolation over a triangular grid, and interpolation with continuous first derivative over the triangular grid, both on a plane surface and on the surface of a sphere, and those things were I think fairly new at the time I did them. Some of those things I published, some I didn't. So I had a continuing interest.

One thing I will mention here came fairly early, around 1965, and this went back to pointing the antennas out at Goldstone (JPL operates a set of antennas at Goldstone, CA for communication with spacecraft and satellites.). If there's some mission underway where the antenna had to track something in the sky, most likely a spacecraft, or it might have been the moon, they needed information as to where that antenna should be pointed at all times. What they were doing was running a computer program on a mainframe at JPL that computed that information, and then they were interpolating it to, I think, about one-minute intervals, and they would punch out on punch-cards the position of that thing at one-minute intervals through the whole day, and then they would take I guess the cards, it was probably a reel of magnetic tape, take that to a smaller computer that punched paper tape, and they would read that magnetic tape and punch the paper tape, and have the angles that the antenna was supposed to be at at each minute in that paper tape. The paper tape was notoriously mechanically fragile and would break, and it took an operator sticking with it all the time to be sure that if it broke, then he would straighten things out and get it going again. And it took him about a day to produce what they needed for a day out at Goldstone, so it was rather frantic. And they were doing it on the paper tape because the equipment they had out at Goldstone wanted to read the paper tape to drive the antenna.

I suggested that we could probably fit all the angles for one day with one polynomial, and I did some experimenting and I found you could do that with a degree fifteen polynomial, and do it in a stable way if we used a Chebyshev basis for the polynomials, and we could just send the coefficients of that polynomial. So we just sent two sets of sixteen numbers out to Goldstone instead of this reel of paper tape that had a couple of angles for every minute of the day. So they were able to greatly reduce the physical and mechanical and the logistic work associated with

that. And as I say in my write-up, it removed the need for about forty hours of a technician's work per week, which is to say it eliminated the need for a job. Which wasn't maybe too cheery to that guy, but I thought it was progress, or I sure hope so anyway.

HAIGH: Let's hope they found something else for him to do.

LAWSON: I don't think this really was a reason for him leaving, if he left, because I think I remember coming down to where he was going to be leaving, or at least leaving that job, and people were saying, "Well what are we going to do when he's gone?" and I said, "You're not going to need that anymore," and they said, "Oh, really?" and it turned out yeah, really.

HAIGH: It's interesting that in the memoir you discuss a number of papers that you've published and presented at different places. But when I asked you about your contribution just now, you were focusing much more on specific programs that you produced. Is it fair to say that you think that your contribution came more through actual pieces of software than through academic papers?

LAWSON: That's an interesting observation on your part. I think my internal interest level and motivation was most stimulated by developing an algorithm, writing a program and seeing it work, particularly if it did something useful for somebody. So maybe that's why I'm talking more about that kind of thing. But I understood at some point along the way that it was useful to publish things, and so if I'd done something that I thought was publishable, I was pleased to be able to write it up and publish it. But I guess I never felt like I was in the position where I'm going in to work today to do some more research so I can publish a paper. That is not what was driving me. But I realized the value of publishing papers, and if everybody just works in the dark and doesn't tell anybody what they're doing, then everybody's going to reinvent the same things, so it's important to publish some things. But that was not probably my motivating factor.

HAIGH: So at the lab, you wouldn't have felt the same pressure to publish that you might have done in an academic position?

LAWSON: I think certainly not the same pressure I would have felt in an academic position. My management was pleased when I published things, so I realized it was a good thing to do, but it wasn't a driving thing.

HAIGH: In your memoir, you're quite thorough about the various presentations and meetings. I'm wondering, are there any papers where you're particularly aware that they did have any influence on work that other people were doing, or where someone took the idea up and extended it later?

LAWSON: That's a tough one. I don't know. There were times when maybe some kind of application-oriented people in an audience would talk to me afterward and wonder if I had some software for what I'd been talking about that they could get, and I usually was able to do something for them in that. As far as stimulating other researchers, I don't know. As I mention in the write-up, John Rice took an interest in the thing out of my dissertation, using weighted least squares to get minimax approximation, and he did have at least one student and maybe a couple Ph.D. students that worked on that idea to find out more about it. But I don't know that their work really blossomed particularly into anything. I feel like the benefits of what I've done, and I

hope there are some, were more to people that were able to use the ideas and application or use software. I've had people come up to me at SIAM meetings over the years, and they'd say, "Are you the Lawson of Lawson and Hanson Least Squares book?" I'd say yes, and they'd say, "Well we used that in such-and-such a job, and it really helped us and we were really happy to have that." And the *Computer Approximations* book with John Rice and the others that came out before 1970, the same kind of thing, I've had people come up to me at meetings and say that they had the job of writing sine, cosine, exponential routines for some new computer, and they found this book very helpful. So I've been pleased to get that kind of feedback. But I don't think I've ever felt like I was really on the team with people that were advancing research in the sense that they'd pick up an idea from me and it would do them some good in a research sense.[3]

HAIGH: Let's move now to talk about those two books, then. The book *Computer Approximations*, it seems from what you've written, was specifically about approximating elementary functions.

LAWSON: Yes.

{See ahead to page 43, beginning of "Session 2". The book treats some special functions beyond just the elementary functions.}

HAIGH: And that it came out of an ad hoc meeting at the National Bureau of Standards round about 1964. Now at that point, do you have a sense of how well developed the knowledge of elementary functions for computers was? What was the state of the art?

LAWSON: Pretty primitive. Of course, people knew the Taylor series for the sine function and all kinds of things like that, but there was no body of knowledge collected on what do you really want to do if you're going to write a routine for one of these functions. Like on a sine function, it's pretty obvious, since it's periodic, you're going to concentrate on approximating it over one period, and then you're going to reduce its value, and you're going to get its value in other periods from that one. But there's just a lot of ways that you could break that down, and what's going to be a good way to do it and what isn't wasn't well known by any means. The Fortran language had the sine function, the cosine function, and things like that as part of the language, so if a company was going to come out with a new computer that was going to have Fortran on it, they had to have the sine, the cosine, the square root functions and the job was probably assigned to some assistant programmer, just whatever programmer was working on the compiler. "Okay, you do the elementary functions." And they would probably be starting absolutely from scratch, and they'd probably go back to their college calculus books and they'd look at the

[3] Lawson adds on March, 2006: Regarding the question about a paper of mine influencing later researchers. I just became aware of the web page: http://www.cs.cmu.edu/~quake/triangle.research.html, where the author of the software module, "Triangle", Jonathan Richard Shewchuk,(http://www.cs.berkeley.edu/~jrs/) of UC, Berkeley, says: "Though many researchers have forgotten, the incremental insertion algorithm for Delaunay triangulation was originally proposed by Lawson.", and he cites "C. L. Lawson, Software for C1 Surface Interpolation, in Mathematical Software III, John R. Rice, editor, Academic Press, New York, pp. 161-194, 1977." Shewchuk's module, "Triangle", was awarded the "Wilkinson Prize" for Numerical Software in 2003, and is used in larger software packages for scientific research, including "Quake" at Carnegie Mellon University (http://www.cs.cmu.edu/~quake/) and in NCL, the NCAR Command Language (http://www.ncl.ucar.edu/). Also see citations below under the heading, "From Approximation Theory to Computer Approximations" of papers by other authors who further studied the Lawson minmax approximation algorithm.

Taylor series and figure out something to do. So it seemed to those of us who attended that meeting that it would really be helpful to the field to have some collection of what could be suggested there put together in one place.

HAIGH: Now, you mentioned, among the work that had already been done in this area, that Kahan had made a contribution. Do you know what he'd worked on?

LAWSON: This was something I had forgotten until I worked on preparing this document a month ago. There's an article on one of the SIGNUM newsletters in which he is claiming that he has written the most accurate routine for something. I don't remember if it was for one elementary function or for a group of elementary functions, it may have been just for one elementary function, but that he felt he had written the most accurate routine that was possible for that function on that computer, which I think was probably the IBM 360 series. When the IBM 360 series came out, that really threw up a challenge to numerical analysts, because the built-in arithmetic was based on hexadecimal digits instead of binary bits. Of course, you can say, "Well, a hexadecimal digit is just four binary bits." Yes, but if you had to shift to line some things up to add in a binary machine, you may shift them one bit and then you can add them. In the hexadecimal machine, if you can't add them at this setting, you've got to shift them four bits because it's in hex. So the whole general behavior of round off was just a whole lot worse than anybody had every seen before. A lot of people, maybe including me, would tend to throw up their hands at that. Kahan is the kind of guy that really thrives on what looked like impossible problems, and he dug into that, and he wrote some routines that he claimed were as good as you can do on this machine. So my reference there is really from a SIGNUM article I came across while I was putting this together. Hirondo Kuki is mentioned there. I think he was working either full-time for IBM, or maybe he worked at Argonne but did some collaborative thing with IBM or something. But he I think prepared the whole library of elementary functions for that same machine, for the 360 series.

HAIGH: Yes, I think he was in charge of something called the Shared Numerical Analysis Project where they were trying to improve the quality of routines in the SHARE Library for IBM computers.

LAWSON: He certainly might have done something like that. SHARE, which I don't talk about in my write-up, and I should, was certainly one of the earliest efforts to collect software and make it available to people again. It couldn't be called a library, I don't think, because I think of the library as having some organizational integrity to it, some standards for how something gets put in the library and why it gets put in and how it's documented and everything, and SHARE, anybody could send something in, and--

HAIGH: I think it's more like a literal library where there's a bunch of different books there, and some of them are good, and some of them are terrible, and it's up to the reader to figure out which is which.

LAWSON: Yes, SHARE is like that. If Kuki did something to try to improve elementary functions in the SHARE library, I don't really know that specifically, but I thought he worked in some official way with IBM and actually produced their library that came out with their Fortran compiler on that machine, but I was not quite sure.

HAIGH: That's quite possible as well. I'll try and follow up on that.

LAWSON: I don't know what his real institutional home was at that point, if he was at Argonne or where he was. But I had contact with Cody. I don't remember I ever had much contact with Kuki, although I had met him. Cody seemed to have very close contact with Kuki, and Cody, in my opinion, and I think in a lot of people's opinion, is really the flag-bearer for accuracy and reliability in elementary and non-elementary functions on computers. He is just unswerving in his devotion to accuracy and reliability, and I don't think there's anybody over the years that has worked on that problem as effectively and as consistently over such a long period of time as he did. People like Kahan are very brilliant, and if it was challenging to him to maybe write one routine better than anybody else had written it, then he could probably go in there and do that. But he wasn't going to make a career out of doing this for all the routines for 20 years or something like that.

HAIGH: Now, you wrote in the memoir that Cody chose not to be part of the book project that produced computer approximations.

LAWSON: Yes. He would have been a very natural participant in that, and I don't know specifically why he was not in it. But my guess, from having been acquainted with him now over many years, is that he has such personal views of what's the right way to do things and what the standard should be that he probably would have felt it would have been out of character for him to work with seven other people on something like this, because I don't think he'd want to have his name on something unless he had really verified everything that was in it. So I don't want to say anything disparaging about him. I think he's a great guy and has made wonderful contributions, but he probably didn't want to be in a team effort like this.

HAIGH: So how did the collaborative process work with so many different authors?

LAWSON: Well, as I said in some of the other paragraphs there, I think we fell down on the proofreading side of it. I had thought for a long time that it was maybe kind of a first-publishing effort for all of us involved, but in the course of writing this up, I realized that Thatcher was in fact involved in the AMS-55 project. [*Handbook of Mathematical Functions*, Ed. Milton Abramowitz and Irene A. Stegun, NBS AMS-55, 1964]. So it wasn't a first-time project for him, but for quite a few of the rest of us I think it was, and we probably didn't give enough appreciation to how much effort you've got to really put into the proofreading, especially something like this that's just packed with mathematical formulas.

HAIGH: Beyond the proofreading, did you just split the functions up between you?

LAWSON: I think that's essentially what we did. I don't remember exactly, but I think one person was supposed to write up one or two of the functions. What I did primarily—I don't remember if I wrote up any of the functions—was computing the coefficients. I say in here, I had developed, using some code out of SHARE, the capability to do higher-precision 70-bit and 140-bit computations. And from my approximation theory background, I knew something about the Remez (sometimes spelled Remes) algorithm for minimax approximation and another algorithm, so I coded up some programs that made use of the higher-precision arithmetic and applied those algorithms and did the curve fitting to get the coefficients. So I think that was my main contribution. I don't remember if I wrote any of the parts or not. And then as I mentioned in the

write-up, a colleague of mine at JPL, Neil Block, took the computed coefficients and put them in a form on tape that he could send to a person he knew in Minnesota (I think he was in the St. Paul/Minneapolis area) who was kind of a pioneer on computer typesetting, and actually typeset the coefficients for us in the book. And I think I say there, there was a total of 150 pages of coefficients.

HAIGH: You said in your memoir that you have an impression that the book was widely used for its intended purpose. I wonder if you're aware of any specific cases in which people producing compilers or libraries drew on the material in the book?

LAWSON: No, I don't think I could name any particular place. But like I said, I had people come up to me at SIAM meetings and say that they had used it and found it very useful. But I don't know who that was or what company they were working for. The book would now be pretty much obsoleted by the book that Cody and Waite brought out. What was their date, '79?

HAIGH: I've got 1980 here. (*Software Manual for the Elementary Functions,* Cody and Waite, Prentice-Hall, 1980)

LAWSON: Okay. They took quite a different approach. Ours was kind of, "Here's a lot of things you could do for this function; here's a lot of things you could do for that function; here's a lot of coefficients you could use depending on the word length and the accuracy you want." They took more of the approach, "Here's one good way to do it for this function; here's one good way to do it for that function; and here's the set of coefficients to use." So it's much easier book for somebody to follow; they don't have to put the pieces together as much as they would with ours. I think it was nicely done.

HAIGH: Your book came out in 1968.

LAWSON: Yes.

HAIGH: So through the 1970s, this would have been the book to go to.

LAWSON: That's right. Over and over, I think in history one sees that when there's a competition between perfection and just kind of getting the job done it's sometimes worthwhile to just get the job done because it might take a long time to get the perfection, and people could be making use of what you got done until that happens. And I think this is one example of that. This book is not at the level of complete best presentation of everything that Cody might have liked, but we did get it out, and it was 11 years later before he got a book out. I think this kind of thing happens. Once his book is out, then great, that's the one to go to.

HAIGH: Did you come up with a set of sample implementations of these functions for any particular architecture?

LAWSON: No, we never even thought of doing that, I don't think, because it's so different on different machines. We assumed that people writing these functions would be doing them in assembly language; they wouldn't be doing them in a higher-level language, so there wasn't really any thought of that. One thing if I could mention that there was a thought of, and I think it may have been by Philip Davis who I think brought the meeting together. No, I think it may have

been Hamming who I think attended the meeting. I think it was Hamming. But whoever it was, somebody put out the idea that instead of telling people how to write routines for a certain machine, why don't you put this information into a computer program in such a way that it would automatically write the routine for a given machine? And you can conceive of that, if you made the parameters of this program be some kind of definition of the arithmetic system. Conceivably then you could go through some tests and figure out what's the best way to organize this program from a set of two or three possibilities. You know what accuracy you're looking for, so you could pick the right coefficients. Again, this is kind of a tradeoff between some kind of perfection or grander idea and a simpler idea. We decided on the simpler approach because we thought we weren't really ready to tackle that. I would say an example of tackling that kind of thing successfully—not in the elementary function area, but in BLAS area….

[Tape 2 of 4, Side B]

LAWSON: ….something along those ideas has been done now by Jack Dongarra and his people, I think they call it ATLAS. The ATLAS program will basically write basic linear algebra programs (BLAS, see elsewhere in this document) for a given machine. I guess it kind of automatically tests different ways of writing and then picks the one that was best on that machine, and that's a wonderful thing. But when did they do that? They did it within the last ten years, I guess. We were back here in 1965 or so, and we didn't think we could undertake that sort of thing, and that's the way things go. It's a grand idea, but it just wasn't the right time to try.

HAIGH: As you know, it was in the 1970s that people were looking more at portability, and I think they did come up with these parameter systems, but only for Fortran. I don't believe for assembly.

LAWSON: That's a related story.

HAIGH: Which we'll talk about later.

LAWSON: Yes.

HAIGH: So the least squares book, *Solving Least Squares Problems* by Charles Lawson and Richard J. Hanson, Prentice Hall, 1974.

LAWSON: Well, I'm waving at the microphone three copies of the book. [laughs] There's the original one; there's a Russian translation, which is fun to look at the Russian letters, if you understand Russian it would probably be even more fun; and then the SIAM reprint of it, which we were very happy that they decided to do it a few years ago. I've talked before about how particularly Richard Hanson got closely involved with people doing the orbit determination navigation, and was able to show them that some of these orthogonal methods could improve their reliability. Coming out of that work at that some point—and I don't know the date for sure; I said in there about 1970—we were invited up to Stanford to talk about the work we'd been doing. We had published a paper in *Numerishe Mathematik* that gave an idea of the kinds of things we were doing, to organize the Householder transformations in ways that we thought would give a flexible approach to applications. So Gene Golub invited us to come up to Stanford and give a talk on the work around 1970. George Forsythe was the head of the Computer Science Department there at the time and was present.

HAIGH: How would you describe Forsythe?

LAWSON: Well, I never had any extended periods of time with him. In fact that may be almost the only time I ever met him. I may have met him in passing at some meetings or something, but I never had any extended dealings with him. My recollection is that he had a kind of physical activity he projected. I don't know, maybe the way he moved or something that he seemed to be excited about things. And so he was a very satisfying person to present ideas to. If he liked what he was hearing he gave good, strong feedback when he found things interesting. So my recollection of that rather brief encounter with him was all very positive. I thought he gave us a very strong endorsement that he thought what we were doing was worthwhile and he would like to be involved in helping us get it into publication as a book. So I think the world of him. I don't know him very deeply, but…

He and Gene arranged with Prentice Hall to publish the book, and we worked over the next few years co-authoring it, Richard Hanson and I. And I say in the write-up, my recollection is that we started out with the idea that I'd write a chapter and he'd write a chapter or something, and early on we found that when we looked at what the other one had written, we wanted to change so many of the things that we finally came to the idea we better just sit down together and we did that, and just write every sentence together. Of course, there were no word processors or such at that time, so we were very happy that our secretary was pleasant and accurate in writing the many revisions of this. I particularly remember that she was from Phoenix, Arizona and never felt warm enough in California. She had a little portable heater under her desk that she would have on all the time to keep her feet warm. But she did a nice job for us on that.

HAIGH: Had there been a previous book-length treatment of this topic?

LAWSON: I don't think so. No, I don't think so. No. We were basically giving an exposition of how the Householder transformations and the Givens rotations could be used in least squares computation. Of course, that had been published in papers, particularly in Golub and Businger [G. H. Golub and P. A. Businger, (1965) "Linear Least Squares Solutions by Householder Transformations", Numer. Math., 7, Handbook Series Linear Algebra, 269-276], and, for the singular value things, Golub and Kahan [G. H. Golub and W. Kahan, (1965) "Calculating the Singular Values and Pseudoinverse of a Matrix", SIAM J. Numer. Anal., 2, No. 3, 205-224]. We were also introducing the idea of what we called sequential accumulation, where you solve for part of the data and then you can add some more and solve for some more. Of course, people have always done that in least squares, but they hadn't done it with orthogonal transformations, and we were showing how that can be organized to be reasonably efficient. What we were doing was an exposition primarily for the practitioner, the engineer or scientist, to be able to more easily get a grip on what these methods were and how they could be used, and then we did include about six or seven Fortran codes that implemented different facets of this.

HAIGH: Are you aware if the book is something that might be assigned to graduate students, or would it just be for people who had a kind of pragmatic need to solve one of these problems and implements and stuff like that?

LAWSON: I think it definitely has been used by graduate students. But since the book came out, there have been books by Pete Stewart, and by Åke Björck.

LAWSON: So there have been a number of things written since this. Whether it's appropriate for a graduate student in numerical analysis now or not, I couldn't say. But I think it's still useful to a practitioner, and there might be some ideas in there that a graduate student wouldn't have thought of that might not have appeared in other books, too.

HAIGH: So it had a dual life as a text that could be assigned for someone with a fairly specialized interest, and as a book that would be pragmatically useful to people who needed to implement a routine to solve a specific real pressing problem.

LAWSON: Yes, I think so.

HAIGH: Is there anything else you know about how and where the book was used?

LAWSON: Nothing particular. I remember one science professor at Caltech that was very happy with a particular routine out of the book, and with the book for giving him understanding of the routine. A lot of scientists and engineers really aren't comfortable unless they do feel they've understood an algorithm they're using, and this is a case where we give quite a detailed description of the algorithm we use. We didn't try to give a big library of routines; we've just got a few. But they were all related to quite complete descriptions in the book, and I think some scientists probably appreciate that.

HAIGH: You mentioned that you also produced the *Encyclopedia of Computer Science* article on least squares approximation.

LAWSON: Yes. Tony Ralston was one of the co-editors of that encyclopedia, and I can show you that. It's like that thick [indicating size]. He and I had had passing acquaintance over the years. Of course, he had co-authored a book on numerical methods in the early or middle-'60s that described how to do about six different things in numerical analysis. It was very, very clearly set out, and I always felt that was a very useful book. So he had appreciation for numerical things, but he also has this encyclopedic view of things. He asked me if I would write something on that, and I did, and I was happy to be able to do that. And then as I say in here, I suggested they pass that responsibility on to Richard Hanson after I retired in 1996, and they did that. But I think the nature of the book has maybe changed also. It was very thick and probably very expensive, and had a lot of stuff in it. I know they have come out now with a kind of a shortened version that has no math things in it, and I think they must've found their main market was more of the computer science/system engineering/system programming type side of things. The math things were probably ignored, so they saved a little space by not having it. I don't know if they still do the full encyclopedia or not.

HAIGH: Actually, I should say for the benefit of anyone who's reading the transcript that within the last year or so, an article appeared in Annals of History of Computing on the history of that encyclopedic project. [Anthony Ralston. "Four Editions and Eight Publishers: A History of the Encyclopedia of Computer Science," *IEEE Annals of the History of Computing*, vol. 26, no. 1, pp. , 42-52, January 2004].

LAWSON: Oh, really? Okay.

HAIGH: I can give you a copy of that.

LAWSON: It'll be interesting to see. Because it's mind-boggling what a big undertaking that was, and that Ralston has stayed with it over so many years and I don't know about the others. I refer to it myself for details in computer science or computer history, and I find it very helpful.

HAIGH: Well that seems to have wrapped that up for the two book projects. Is there anything else in that kind of area of publication or conceptual contribution through your career that you think we should discuss?

LAWSON: I guess nothing that occurs to me right now, thank you.

HAIGH: Okay, well that seems to wrap that section up. It seems then that we should move now to consider the topic of libraries in mathematical software. I wonder if I can begin by asking you about your impression of mathematical software libraries that became available during the 1960s. So one of those that you mentioned earlier was the SHARE software library.

LAWSON: I wouldn't call SHARE a software library—maybe that's just me. In my mind, a library of software carries some characteristics along with it in order to be called a library. But there's some systematic nature to it or there's some standards for how things get into it, there's some standards for how things are written up in it, and a number of other conditions that give it a more organized character as distinguished from what I would just call a collection of software, which is something where just people can put things in and people can get things out, and there's no standard as to how things are written up or how things have been tested or anything like that. Both things have their place, but they're different. So SHARE, in my mind, is not a library; it was a collection. Essentially anybody that had the IBM mainframes in the 1960s and wrote some software that they thought might be of general use to other people using that type of machine, could send it into SHARE. I don't know what criteria they had for accepting things, but I think it was pretty open. So if something was in SHARE, then other people could get it. I think that's the way it worked; I don't know. They may have been more organized than that, but I think that was the idea. In my mind, in the 1960s, you couldn't do much more than that because everything was so new. If somebody someplace wrote a routine that looked like it solved some kind of problem better than they were aware that anybody had done it before and they were willing to make it available freely to other people, they'd send it into SHARE and SHARE would accept it and it would probably be useful to other people. So it certainly served a purpose. But the times changed, and as more software got written and people became more discriminating between software that was well written-up and collections of software that were organized. Different ideas came forth as to how this kind of thing should be done.

HAIGH: You mentioned that you took a routine from the SHARE library.

LAWSON: I took a routine for high-precision arithmetic, 140-bit arithmetic.

HAIGH: Did you experiment with many mathematical routines that had been contributed to the SHARE library, or was it this an exception?

LAWSON: I don't remember making many efforts to get things out of the SHARE collection. That routine was pretty unique at the time. It allowed a person to do floating point arithmetic and carry 140 bits of precision. There were two packages: one that carried 70 bit and one that carried 140. And, I forget, one or the other I think I got from SHARE. Of course, I did my own testing

on it once I got it, and I think I pretty well made sure I understood how it was working and it was written in assembly language, so I had to look through that because I didn't really feel I could just trust something like that that came from SHARE.[4]

HAIGH: Did you have any contact with the IBM package SSP that came available in 1966?

LAWSON: No, because we were not using the IBM 360's. I don't know if that ever came out on the IBM 7094's. I think that came out on the 360's. I could be wrong. Is that right?

HAIGH: Yes, I think you're correct. Although it was in Fortran, so I know that it was unofficially used even on some non-IBM machines, but I believe it was only used officially with 360 series.

LAWSON: JPL was not using 360s primarily for science and engineering computing in the late 1960s. They did get those machines for their business data processing, so we had that type of machine, but it was being used for the business data processing. So I don't remember that we ever had any experience with that library. I had some awareness of it and I know that Ed Battiste was the leader of that effort, or certainly involved in it, and at some point decided to break off and be involved in forming IMSL. I had a chance to be acquainted with him through our common membership in the IFIP WG 2.5 working group, and I know he just felt that quality wasn't being controlled the way he thought it should be, and he wanted to be in a company where he could direct more independently how things were done.

HAIGH: Did you have any contact with the Harwell library?

LAWSON: Well, I had contact with the people behind it because I spent that one month on a NATO grant visiting the Harwell people in 1970, so I had some acquaintance with some of the people using it. The pricing on that seemed high to me at the time. Now I personally have had a slow adaptation to the idea of putting a price on software. When I started out in 1960, I remember going to an ACM national meeting and somebody gave a talk on an interesting new piece of software and I asked him afterward, "Is that available? Can we get it?" And I think they were selling it and I was shocked. So in early 1960, I thought people should give software away. Clearly that's not an idea that has survived. The Harwell lab, right from the beginning, seemed to put quite a price on that. I guess they made it available maybe to academic institutions, but I think JPL would have had to pay quite a bit for it. It just never seemed to me to be something I would really recommend to people. I've nothing against the library. I don't know when it really became available—around 1970, or later than that maybe. Up to 1970, you could think in terms of two or three mainframe computers. But after that, you had a dozen different kinds of computers. So if you were trying to help the whole laboratory have a library, you couldn't do that by putting it on one machine. And so it didn't seem to fit into what I thought JPL needed. Now individuals still might very well have gotten it, but our group didn't have any involvement with it.

[4] Lawson adds in March, 2006: Regarding use of codes from SHARE. I have come across documentation showing that I used code from SHARE for stepwise regression in JPL applications in the 1962-1965 time period. This was SHARE code No. 1194 of October, 1961. I called my version of the code, STPRG2.

HAIGH: Prior to about 1970 then, presumably JPL had built up some kind of collection internally of reusable subroutines for different mathematical functions.

LAWSON: Yes. My recollection here is very foggy, but I think in ’60 to ’65, it was just a person-to-person kind of thing—if we had written a routine that we thought was of general usefulness, and we talked to some engineer and it seemed like they could use it, we’d tell them, “Here’s this routine you can use.” So there was not too much organized about it. I would say beyond 1967 or 1968, when we started switching over from the IBM 7094s to the Univac 1108s, we started the idea of having a library of routines to use on the 1108 because it looked like that was going to be the main machine that the lab used for engineering and scientific work. We did start trying to make a kind of systematic collection. It wasn’t terribly systematic at the time; it was just kind of the software we had. We tried to put the write-ups into some more standardized form than we had before, and it was only sitting on the Univac 1108, so it wasn’t useful to anybody on any other kind of machine. That was the state of our affairs from around 1968 to ’70 and on to ’74, ’75 probably.

HAIGH: Did this collection of routines that you accumulated include many that had been developed outside the lab?

LAWSON: I just couldn’t remember what was in that, and I don’t know if I found any copies of an actual manual for that library, so I don’t know. My guess would be probably not very much from outside. Fred Krogh particularly developed a very sophisticated routine for ordinary differential equations, and Richard Hanson and I had done some least squares and other linear algebra things. And then Snyder probably did some special functions, Bessel functions and things like that. But I don’t know what we had in that library. That’s pretty much a blank at this point.

HAIGH: On the other side, are you aware of cases in which routines developed within the lab found their way into use at other sites during the ‘60s?

LAWSON: Found their way into our library?

HAIGH: No, the other way around—that there are routines that you produced found their way into libraries at other labs or corporations.

LAWSON: I don’t know. Of course, when we published the least squares book in ’74, the software was made available through *ACM Transactions on Mathematical Software*. People could freely pick that up and put it into libraries, but I don’t know of that. The only other big noncommercial library project that sticks in my mind is the SLATEC project among some of the Department of Energy laboratories. But they had quite a few people developing software and I imagine their library consisted pretty exclusively of things they had. I don’t know.[5]

HAIGH: I think that was later.

---

[5] Lawson adds in March 2006: Regarding our code being used at other institutions. I have just become aware, due to email correspondence with Prof. A. J. Semtner, that he used a sequential least squares program, LSQSL2, due to Hanson and me in the mid 60’s, while he was doing ocean research at Princeton in the early 70’s, and that the code migrated from there to being used at MIT and Los Alamos. He thinks it was still in use in the ocean research field at least as recently as the 1990’s.

LAWSON: Was it?

HAIGH: My impression is the mid or late 1970s, so no, not the 1960s.

LAWSON: Probably. Yes, up to 1970, there just wasn't very much of what would nowadays be called an organized library, I don't think.

HAIGH: I think in some ways, it's surprising then that people weren't swapping routines more. If you were in touch with people at other sites and they were using the same machines, would it really take you longer to test their routine than to write your own? Why not more exchange?

LAWSON: Well, I imagine there probably was some of that. Of course, it's hard to turn my brain back to that period of time. There was no email; there was no electronic transfer of computer digital material. If you were sending something, you'd either send a box of punch cards or a roll of tape. Sometimes the exchange of tapes between different machines didn't go as well as you expect it to. So there were a lot of things that just made it not as convenient as we think of today. Now certainly, you could do it, and I'm sure it was done, but there were just a lot more difficulties in it, I think.

HAIGH: Let's move on then to the first main contribution you talk about here to the development of shared and portable libraries between sites, the BLAS—Basic Linear Algebra Subprograms. Now I imagine that when you talk about this, you'll probably also want to talk about the LINPACK project. So I should say for the benefit of people who are listening to this that they will also want to consult the transcripts of the interviews with Cleve Moler and Jack Dongarra, with Jim Pool, who was in charge of that part of the (Argonne) lab at the time that the project was begun, and with Jim Cody, who was involved with the same grant that funded work on the LINPACK and EISPACK projects.

LAWSON: Yes, so I guess Cody probably did a FUNPAC. Did he do FUNPAC, more or less, in parallel with that or something?

HAIGH: Yeah, it was part of the same grant as EISPACK. They were both supposed to be researching software development and testing methods, I think.

LAWSON: Mm-hmm [yes]. Well, as I mentioned in the write-up, sometime after 1970, the thought was really in the air every place. For anybody that was thinking at all about libraries, you need to have some portability, but you still needed to maintain some reasonable efficiency making use of hardware improvements. I mentioned here the CDC 7600 or 6600, I don't remember which one. One of those introduced the idea of having an instruction cache of something like 16 instructions, which meant that if a loop was complete with that number of instructions, it could be brought into a cache and then execute the loop any number of times without having to go back to memory to fetch more instructions. To take advantage of that, there was a motivation to go back to assembly language and write your codes so that you got any extraneous instructions out of your loop and got it down to that size. You didn't want to do that with a big piece of software, and so it focused the idea on identifying small pieces of software that are significant in the overall running time of the program. Of course in linear equation solving, it's the classic case. You've got a process that is a factorization algorithm, is a triple-nested loop, and that inner loop represents essentially all your computing time. So if you could

write assembly code to do just what that inner loop was doing, in a way that made best use of the machine you were on, you were going to get some payoff.

HAIGH: So today people would think of that as being a job for an optimizing compiler, but you're saying that at the point the idea was the include some assembly language for that tight inner loop, within the source code inline with the Fortran for the rest of the routine.

LAWSON: Yes. Well, it wouldn't necessarily be inline; it could be even a subroutine call to get to it. But what goes on inside that routine was really the time consuming part—because that's the only place floating-point arithmetic was going on. If you had a factorization routine, you've got these three loops and you've got one arithmetic statement. I mean there's only one statement in the whole program that's doing floating-point arithmetic, and that is the inner loop. So you could afford to call a subroutine that represents that whole inner loop if the internals of that subroutine were going to go very fast. That was, how you say, 90% of the motivating idea. Another 10%, you could say, conceptually—and maybe I gave this more weight than some other people did—was the idea from what was called structured programming. I associated with that idea of structured programming that you have small subroutines that do unique things, and you use a bunch of them to write a big program rather than just writing one big program that does all kinds of things. So the idea of having a small subroutine that does the dot product of two vectors fits into this idea of structured programming. When somebody looks at the code, if it's all written out in line, all they're seeing is a bunch of loops and they'd have to look at it or study it a little bit to figure it out, "Oh, that's really an inner product in there." If instead, you have a call to DDOT, which is double-precision dot product, you just know right away that that's a dot product there. So from the point of view of self-documentation of code, and my notion of what is one of the features of structured programming, this idea of having names and standard calling sequences for certain standard basic operations had an importance apart from the efficiency issue. Now it never would have sold to anybody if it didn't have the efficiency issue, I'm sure, but I felt that was another aspect of it.

HAIGH: You think that by this period of about 1972, 1973, you were already familiar with the idea of structured programming?

LAWSON: Well, the idea had come forth. I'm not a computer scientist, but in my mind, one of the things it meant was organizing a large program as a bunch of smaller ones. That's pretty broad-brush treatment of the structured programming, but that was in my mind one of the characteristics of it. Just the general idea, which maybe I don't have to associate with structured programming, just good practices, that when you look at a program, you make it as easy as possible to see how it's organizing what it's doing. If you could do that in a code, you don't have to depend entirely on comments, that's all the better.

HAIGH: Yep. Actually, I'm just checking some references and the date fits there. Dijkstra's famous letter, "Go-to Considered Harmful" was published in 1968. [Dijkstra, E. W. (1968). "Letters to the Editor: Go To Statement Considered Harmful." Communications of the ACM **11**(3): 147-148]. And then by 1972, the book by Dahl, Dijkstra, and Hoare, *Structured Programming*, had come out. [Dahl, O. J., E. W. Dijkstra, et al. (1972). Structured Programming. New York, Academic Press]. So that would be, I think, an idea that was…

LAWSON: In the air.

HAIGH: Yes, in the air. New and exciting, but really quite widely disseminated at that point.

LAWSON: Yes.

HAIGH: Do you think that this idea of structured programming is something that was widely considered as important within the mathematical software community?

LAWSON: I think it was a personal choice. Different people would have had different ideas about it. It appealed to me a great deal. Sometimes it worked against efficiency, and in my mind I would be biased toward the structured programming and give up some efficiency. Other people I know would go the other way. They would say, "No, I don't want to do that because it's going to cost me something." So different people had different ideas on that, I think.

HAIGH: In discussing your motivations for creating the BLAS, you talked primarily about efficiency, and you've mentioned structured programming. One thing you haven't mentioned discussion the original motivation is portability. Did the portability side come later?

LAWSON: Well, no. I'm sorry. Portability is a key idea. But the idea is to support portability. An individual BLAS routine itself, like a dot product routine, is not intended to be portable; it's intended to be written in very efficient assembly code for a particular machine. But the reason for doing that is to support portability of the routine that's going to call it. So the routine that's going to call it should be written in portable Fortran, and is just avoiding coding up this loop in Fortran. Instead of coding the final loop in Fortran, it's calling this BLAS routine, which one hopes is written in efficient assembly routine for the machine you're on.

Now if you move the big code, the whole code to some other machine, you can't take the assembly code along. So what do you do? You either substitute a Fortran-coded BLAS, which will not be efficient, or you find or write an assembly-coded BLAS for that new machine. And of course, our theory, our assumption was that people would find it was worthwhile to write these efficient assembly code routines once for each different machine, and it would be made available in such a way that if I moved to another machine, I know how to find the assembly-coded BLAS routine for that machine. The intention is to support portability with efficiency, so make it possible for people to write their portable codes, but not have to give up efficiency if they can find this assembly-coded efficient routine on the different machine. That's the long story.

HAIGH: So right from the beginning, you had the idea that people would be producing an implementation of this set of subroutines for each machine family that was important.

LAWSON: Exactly. And of course, the wonderful thing that Dongarra and his people have done now is essentially automate that process.

HAIGH: Do you have a copy of this document you mentioned here, CM 303, on the use of assembly code for heavily used modules in linear algebra? [CM- 303, "On the Use of Assembly Code for Heavily Used Modules in Linear Algebra", Fred T. Krogh, May 2, 1972].

LAWSON: We have to request that from the archives of JPL, but I think there would be a good chance that they would probably have it. If you are making a list of things we want to request, you probably want to note that. Because that really was the first write-up. Fred Krogh, in my

judgment of his personality, would tend to be heavier on efficiency than on structured programming. So I would say his motivation here was almost 100% efficiency, or maybe 99% efficiency and 1% structured programming. My thought would have been maybe 90% efficiency and 10% structured programming, whatever that all means.

HAIGH: That's 1972. And then you've got another document the next year, 1973. This one, "Jet Propulsion Laboratory Technical Memorandum 33-660." [R. J. Hanson, F. T. Krogh, and C. L. Lawson, "A Proposal for Standard Linear Algebra Subprograms", Jet Propulsion Laboratory Technical Memorandum 33-660, November, 1973, vi+14 pp].

LAWSON: Yes, now that should be more easily available. Maybe you can still get it from the archive, but it shouldn't be such a rare thing because this is a higher-level document at the lab, a Technical Memorandum, and they should definitely have that. Krogh's original write-up was kind of first thoughts on some routine that it would be good to do in Assembly code. This was the distillation of our thinking on a package of routines of that type, so this was a really early formulation, something on the order of 38 routines to make up the BLAS. What Fred Krogh wrote up originally was probably talking about two or three routines.

HAIGH: Was it Krogh that had the original idea and then you and Hanson who generalized it for the complete range of functions that you'll need?

LAWSON: Yes. I would say that within our group, Fred was the first one that had the idea and took it seriously enough to write some test programs, and actually run some tests and see if it was faster to call an assembly language routine than to have it written in Fortran. He was the one that had the idea and thought strongly enough about it to dig in and experiment with it. And then the three of us together formulated the slightly bigger package idea. But I stress that that's what happened within our little group. Other people at NAG or at Bell Labs or, who knows, all over other places, were probably doing the same things. I know there was a package developed at one of the DOE labs here, probably Livermore or Sandia, for basic vector operations, and they did not only dot product and scalar-times-a-vector-plus-a-vector, but they would do sum of two vectors and some other things, which we did not do. So there's a different mindset of what they were trying to support. We were trying to support solving linear equations or least squares problems. If somebody said, "Well, I'm trying to support a range of vector operations," well then you certainly would want to have a routine that adds two vectors together. But that isn't something that normally comes up in solving linear equations, so we didn't put that in our package.

[Tape 3 of 4, Side A]

LAWSON: We weren't trying to do a complete vector operations package.

HAIGH: You said that the main motivation for this was the arrival of new and more diverse kinds of architecture around the 1972 period. Are you aware of any similar efforts outside this area of numerical software to do the same kind of thing?

LAWSON: To treat something other than the basic things in linear algebra?

HAIGH: Yes. I'm not aware of anything, but I was wondering if designers of compilers or any other kind of area or software might have faced the issue and devised similar solutions.

LAWSON: Again, I'm not a computer scientist, but I know people were talking about it and writing so-called compiler compilers. They were writing software that you presumably would input some characteristics of your machine and it would produce a compiler for you. I think that was a very active area of work among computer scientists. I wasn't involved in that; I don't know how much you can say it compares with this.

HAIGH: As you say here, the original BLAS dealt only with one level of looping, essentially with vectors rather than matrices. Was that because the cache size on these machines was so small that there wasn't any point in doing matrices?

LAWSON: That was our thinking. By the time it actually got published, hardware had passed us up on that. But that was our thinking when these first things were done in 1972 and 1973. We knew about the CDC machines that had these fairly small caches. I don't think we had any real awareness of Cray machines at that time; they were vector machines. I don't know when vector machines first appeared. That would be interesting to look at.

The TOMS article was published in 1979. See, all that time goes by and you look back now and you say, "What was going on? We had this written up in '73; why wasn't it published before '79?" I don't know exactly. Part of it was that the LINPACK people decided they were going to use it, and we were very happy about that. So they dug into adding some loop unrolling into the codes, and before they were published, they had that in it, which was good. But somehow along the line, time went by. So by the time it came out in '79, around that timeframe, I remember I made a visit for some reason to Los Alamos, and a fellow that I was talking to there said, "You know, your BLAS aren't quite all we need on the Cray computer." He explained to me why this wasn't going really help them with what they needed help on with the vector machines, and that you needed to have two levels of looping in your subroutine, and all I could do at that point was say, "Thanks for educating me." I wasn't thinking of making a career out of doing more and more BLAS, but other people did follow up on it, fortunately.

HAIGH: Within JPL, did you have a CDC 7600?

LAWSON: No, we didn't. That was the machine that sticks in my head as being the easiest to talk about in terms of this instruction cache and why it would be worthwhile.

HAIGH: Can you remember any machines that you did assembly BLAS for in this time period?

LAWSON: Fred certainly wrote assembly coded routines on the 1108, and I think his testing showed that it was worthwhile on that machine. Now I don't know exactly why. I don't know what characteristic of the machine made it worthwhile, but he did. I don't know whether we or other people put them on VAX computers because they were becoming pretty popular at the lab.

HAIGH: Can you remember where the name "BLAS" came from?

LAWSON: That's "Basic Linear Algebra Subprograms."

HAIGH: I know, but why did you decide to call it that and not something else. Can you remember when the name arrived or who thought of it?

LAWSON: What did we call the second report?

HAIGH: The second one was called "A Proposal for Standard Linear Algebra Subprograms," which would be SLAS.

LAWSON: Okay. [chuckles] So we had three of the words there—"linear algebra subprogram." I liked the word "subprogram", by the way, because in standard Fortran terminology, "subprogram" is supposed to cover both subroutine and function. We had both subroutines and functions, so that's why the word "subprogram" was there. So the word "basic" must've come in….

Well, I talked in my write-up about a couple of public presentations of this at national meetings. But I'm guessing that it must've dawned on us or somebody must've suggested that we're not solving the world's linear algebra here, we're just doing some very basic things, and so we must've thought we should put the "basic" in. Otherwise, I don't know. We may have not wanted to have the word "standard" because that maybe had a connotation of ANSI standard and we weren't claiming that; we weren't going to want to go that way.

HAIGH: Anything with "standard" in the title would probably take a lot longer.

LAWSON: Yes. [laughter]

HAIGH: You mentioned the arrival of these new machine architectures as a motivating factor. But presumably, thinking of a project like this was worthwhile in terms of portability, you must've also had the idea that there should be such a thing as a portable high-performance mathematical software library.

LAWSON: Oh yes.

HAIGH: And this was also the period when the NAG and IMSL projects were starting up. Can you remember, say in '72 and '73, when you were getting going on this, had you had exposure to any actual specific portable mathematical software libraries? Or did you just have a vague sense that the time for such an idea had come?

LAWSON: I have to look back. I checked out some dates on that by looking at the SIGNUM newsletter when I made my talk in San Antonio about the history of the BLAS, and I gave some dates of when IMSL and NAG came out with the first libraries and then when they came out with a library on a second machine for the first time, which was a little bit later. So their first libraries were not portable.

HAIGH: IMSL's first library was for the 360 in '71, and NAG began with the ICL 1906-A, also with the first version becoming available in 1971.

LAWSON: But those were not portable, not either one of them.

HAIGH: Well within a year or two both had produced versions for other platforms. I also know from the interviews it was several years after that before they really got any tools to assist with the portability. They just moved it to another platform by manually changing code and so finished up with two separate code bases. And then it was the mid-‘70s before they started being to automate the process.

LAWSON: So at this time, ’72 and ’73, we certainly did not have any examples of portable libraries because nobody had one.

Session 2, November 7th, 2004, Dr. Lawson’s home in San Clemente, California

HAIGH: To pick up the discussion where we had left off last time, we’ll continue talking about the BLAS. I understand you have one correction you would like to make first to your comments yesterday.

LAWSON: Yes. You asked me if the coverage in the computer approximation book was just the elementary functions and I said yes. Looking back at the index, I see that in fact there were other functions other than the elementary functions. Beyond what would usually be considered elementary functions, the book treated gamma function and log gamma, error function, Bessel function, and complete elliptic integrals.

HAIGH: So those other functions would be considered to be special functions, would they?

LAWSON: Yes, I think that’s what they would usually be called.

HAIGH: Thank you. Then returning to BLAS, you had said that the objective of creating libraries that were high performance yet portable was the major motivation behind the idea, but you also said that at the point you began work on the project, your personal exposure to portable libraries had been very limited. Now as the project progressed I know that it became quite closely linked with the LINPACK project. So I wonder if you can begin by talking about how you and your colleagues working on the BLAS first became aware of LINPACK, and how the projects worked closely together?

LAWSON: Well I remember one meeting that I attended that was one of the early planning meetings for LINPACK. I don’t remember how we were brought together for that, but probably at Argonne or in that vicinity. So I did have an opportunity to speak with the people that were going to be the principal developers of it. So we did talk about the role of some basic linear algebra subprograms in the project. I don’t remember the date on that so I don’t know really where it fit into thinking about what might have already gone into what became the BLAS. But I was in close communication with Moler and Pete Stewart and Dongarra at the early stage of LINPACK. My recollection is they thought they would give serious consideration to using the BLAS, but also I think there was a feeling I think they wanted to be sure they had input into the BLAS so it would be a package that did what they wanted to see done. In fact Dongarra did have a hands-on involvement with the BLAS before it finally came out to add loop unrolling into the Fortran versions of the code. Of course, the main idea or the underlying idea of the BLAS was that people would hopefully have optimized assembly-coded versions for different machines. But we were going to provide a Fortran set of routines just so somebody would have something they could use if they run a machine that wasn’t equipped with an Assembly coded version. Dongarra

and the rest of the LINPACK group did want those Fortran codes to be as efficient as reasonable. So Dongarra actually reprogrammed a number of them to improve the efficiency. Looking back on it, at times I have thought it would have been appropriate to include Dongarra as one of the co-authors, but I don't remember that coming up as a thought at the time we were doing it, so somehow we didn't do that.

HAIGH: So you mean an author of the final version that appeared in ACM TOMS in 1979.

LAWSON: Yes, the final version showed four co-authors and it certainly would have been appropriate to include Jack Dongara also, looking back at it. But at the time I don't know why that idea never came to surface as far as I know.

HAIGH: So as well as Lawson, Hanson and Krogh, who obviously you have discussed already, another author showing on that final version is David R. Kincaid. Who is that?

LAWSON: Well he was at The University of Texas, and his computers were CDC computers. We did not have CDC computers at JPL, and of course by the time we finished the BLAS, Richard Hanson had moved his affiliation twice. He left JPL and had been at Washington State University in Pullman, Washington and then had gone to Sandia in Albuquerque. He was there at the time we were finishing, and I don't think he was working on CDC computers anywhere either. Kincaid was very much interested in seeing a good version of the BLAS on the CDC machine, so he jumped in and programmed assembly version for the CDC 6600. And he conferred with us on other things to do with the package, so he was included as a co-author. (David has had a distinguished career at UT, publishing extensively, including a widely used textbook on numerical analysis. http://www.cs.utexas.edu/users/kincaid/drk-pub.html)

HAIGH: Were assembly language versions of BLAS available for the major scientific computing platforms of the '70s?

LAWSON: With help from a number of others besides the four listed authors, assembly versions of at least the time-critical subprograms had been written for the Univac 1108, IBM 360/67 and CDC 6600, and accuracy testing of all 38 subprograms was done on the Honeywell 6000, PDP10, CDC 7600, IBM 370/145 and Burroughs 6700. In his work to produce optimum Fortran coded loop-unrolled versions of the time-critical subprograms, Dongarra worked with 40 separate computing facilities that ran timing tests for him.

HAIGH: Is your general impression that the original objective of having very good assembly language coverage was achieved, or did most people just make due with the Fortran version? Do you have a sense of that?

LAWSON: I don't really know. I know that some vendors of computers took it upon themselves to provide assembly-coded routines for their machines. As a sidelight, when Cleve Moler left the academic world, he went to work for a company in the Silicon Valley area that was making a new mini-computer or workstation. The company took the name SAXPY, which was the name of one of the routines in the BLAS. The S standing for single precision, the AXPY standing for A * X + Y, so the scalar times a vector plus a vector. That company was really stressing that their machine was going to be very fast at doing that operation by taking that as the name of the company.

HAIGH: I understood that the company was called Ardent. Did it change its name?

LAWSON: Maybe their machine was called SAXPY. I thought the company was named SAXPY. Maybe you'd have to check with Cleve on that, but I'm quite sure there was either a company or a computer that had the name SAXPY that Cleve worked for. I do remember the name Ardent, now that you mention it. Maybe SAXPY was the name of the computer. Maybe it was the Ardent SAXPY computer or something like that. But yes, vendors did provide these routines. I know some of them did.

One other point. We were talking about the focuses of the original BLAS, and we've mentioned efficiency and we've mentioned some structured programming thoughts. But since we've talked yesterday, it occurred to me that there were also a couple of things in there where we felt that most people, if they were just faced with programming this operation by themselves, would not take all the care that maybe should be taken. By packaging these things up we could be sure that that had been done. One case of that was the Euclidian norm—the square root of the sum of squares of a set of numbers. It's been bandied about among the numerical analysts for a long, long time that if you have a set of elements of a vector of a certain order of magnitude, and the Euclidian norm is going to come out to be around that same order of magnitude, you shouldn't have to suffer overflow because intermediate results are outside the range. Of course the usual thing, forming squares of numbers, you kind of double the necessary exponent range or you produce exponents that are twice as big as what the original data had and what the result is going to have, so these intermediate results are possibly going to cause overflow. A number of people had proposed ways to do scaling to avoid that, but a person just writing the operation of sum of squares of numbers and then taking square root is not apt to go to all that trouble. So in fact we designed that routine in the BLAS to do what we thought was adequate scaling. This was another feature that a person was getting if they were using the BLAS, that they would have this kind of protection that they might not have wanted to program up themselves.

HAIGH: So in that case by protection you mean making sure that the mathematical integrity of the results is preserved?

LAWSON: No. More to the point, making sure that they didn't suffer an overflow that might stop the computation where that overflow was in an intermediate result that could have been avoided. If the original data was in range and the final result was in range, the user deserves to get a result. With a naïve implementation, they might not get a result; they might get a stoppage because of an overflow. So that was the thought there.

Then there was one other thing more or less along the lines of giving people something they wouldn't have bothered to program themselves. There were a couple of authors, and I think Morven Gentleman was one and maybe Sven Hammarling was the other, who independently had written papers talking about a different way to program the Givens rotation computation to avoid a square root. It really complicates the use of that rotation if one decides to do that, because it means that you are going to carry lots of results in a scaled form instead of in their natural form, and so there's a lot of things to keep track of. Most people, if they were just interested in solving some equations and using the Givens transform, wouldn't go through the trouble of all this funny business to save a bunch of square roots.

We did provide what we called a modified Givens transform as one of the functions or pairs of functions in the BLAS. So we did it in such a way that a person would not have to struggle with all this scaling issue; it would be essentially built in. I thought of that. I don't know about my co-authors, but I thought of it as an educational experiment to see if people would be interested in using what we called the Modified Givens Transformation rather than the ordinary one. I had never really gotten any significant feedback or really any feedback on that, so I don't have any evidence that anybody did use it. My guess would be they probably didn't, because even though we made it reasonably easy for a person to use it, they would have to be more or less conscious of the fact they were using it, and most people just probably wouldn't want to bother. But I think of it as kind of an educational experiment.

In the same connection just mentioned: The reason I preferred to call that a Modified Givens Rotation. The authors that I mentioned who had written papers on it, I think both called it the Fast Givens Rotation. In my opinion the word "fast" had really been established in the numerical analysis field in connection with Fourier transforms, where it represented a change in the timing of some operation, where an n got changed to a Log(n). So where some operation normally would have taken maybe n-squared operations, a fast version of the algorithm would take n times log n. And I felt that if one wasn't getting at least that speed up, one shouldn't use the word fast. But that's just my personal take on it. So I preferred not to call this a Fast Givens Transformation because it didn't make any kind of speed up like that; it was just a factor of possible speed up. So I preferred that we called it a Modified Givens Transformation and that is what it is called.

HAIGH: Now you mentioned in the memoir that another meeting was held to discuss the modified BLAS proposal in May 1974 at the Mathematical Software 2 Conference at Purdue University. Do you know if by that point the link with the Argonne people had already been established?

LAWSON: I would think probably so. I wouldn't be sure of that. It was established at that time I would say if not sooner. I don't remember, like I said, when I attended an actual meeting with the LINPACK principals on the subject of LINPACK and BLAS, but I remember I attended such a meeting and I had the feeling that it was really early enough in the whole process. It wasn't a matter of that they had started on the project and this would mean a lot of changes for them. I felt like it was fairly early in the process. These meetings we had to let more people comment on the BLAS while they were being developed, like that one and then I think I mentioned another one or two…

HAIGH: The one at the National Computer Conference in Anaheim, May 1975.

LAWSON: Yes. None of those meetings brought forth any major changes. It would be interesting to pull that early JPL technical memorandum out of the JPL archives and see what it was we did propose, but I think it was pretty close to the same set of 38 routines we ended up with. My impression was there was a difference of one or two routines, and there may have been a change in the name of a routine somewhere but nothing very significant. The name SAXPY was actually suggested by Cleve Moler at one of these meetings, and that was a change from the name we had used originally, but this was the type of thing that might have been changed at these meetings, nothing very significant.

HAIGH: Now were there any other projects underway, aside from LINPACK during this period, where people expressed an interest in using the BLAS or actually did use it? Or was LINPACK pretty much the only game in town as far as applications went?

LAWSON: Well that was the only one I recall that we were aware of and were having direct and regular correspondence with. But I know other people that submitted their own software packages, for instance to the ACM TOMS, in years after the BLAS came out would use the BLAS. Particularly I'm thinking of an eigenvalue package that a student of Beresford Parlett's produced that was submitted to TOMS and is available in Netlib and made use of the BLAS. And I think it became fairly common that linear algebra type routines submitted to TOMS would typically use the BLAS.

HAIGH: So those would all have been in the 1980s after the 1979 publication of the final version.

LAWSON: Yes. Yes, I don't think we were making available early versions of it to people before it was published. I don't think we were doing anything like that, because we didn't want to get the issue confused. We wanted a clear definition of what the BLAS were. In fact my thinking, and I can't speak necessarily for all the co-authors, was that the definition, the actual paper, was more important than the code we delivered at that time. Because the idea was to have a good, clear definite specification of what these BLAS were so other people in the future could work from it to write assembly-coded versions and they would know what it was they were supposed to be accomplishing. So we didn't want to give out versions before it was published that might not be what we finally published. So the published paper was the goal in this case.

HAIGH: Before we move on, for each of the four authors on the final report, Lawson, Hanson, Kincaid and Krogh, can you clarify his final contributions? I think you have answered this in pieces but I just want to be definite about it.

LAWSON: Well we know from these earlier JPL reports that we've talked about that Krogh was the sole author of the first noted JPL memorandum that investigated whether coding an inner loop in assembly language would give an improvement in running time of a linear algebra routine. Once he'd established that, then Dick Hanson and I were all there in the same group with Fred and we jumped on the idea of specifying enough of these routines that you could do all the critical things in the linear algebra code with them. As far as who actually wrote things down or who actually wrote code, I don't know. But the three of us there contributed to all of that. Then after the project had reached some level of development and completeness, Dave Kincaid came in with quite a bit of interest and wrote and checked out the routines for the CDC, and I think since that was the poster machine demonstrating the value of something like this, we were really anxious to have that represented. So his main contribution was that, writing and checking out the routines on that machine.

HAIGH: Now once the final version of the code was published, if somebody wanted to get the routines would they have requested them from the TOMS distribution service?

LAWSON: Well yes. They were available in whatever the distribution system was that TOMS had at that time. TOMS had a distribution system. It may have meant a person had to write to them and get a tape from them.

HAIGH: Presumably if somebody was requesting a copy of LINPACK from the Argonne code center, which I think was their main distribution mechanism, they would have got some BLAS to go with it.

LAWSON: That would be right, yes. But if somebody were just looking for the BLAS on the basis of it having been published in TOMS, there was a way a person could request those codes and get them.

HAIGH: I should say for anyone reading this that John Rice discusses that in his interview transcript. Also Jack Dongarra talks a bit about the online NetLib system, which he believed started around 1984.

LAWSON: Yes, it's always a little shocking or surprising to me to be reminded of how late some of these things happened because one takes them for granted so much now. Well also it might be interesting to tie this to when Fred Krogh became the editor of Algorithms for TOMS, which was at the beginning of the second year of TOMS. I think volume two, number two of TOMS lists Fred Krogh as algorithms editor. So he may have been editor at the time this came out.

HAIGH: Now you've already mentioned that by the time the definitive BLAS publication came out in 1979 the need for the higher-level versions that would work with actual matrices rather than vectors was already apparent. So can you talk about how that project started and who was involved with and those kinds of things?

LAWSON: Well, I was not involved in the higher-level BLAS myself, so I don't think I can really contribute anything to that. I became aware that there was a place for such things, but I was not involved and so I don't think there is anything I could add to that. The people that developed those would be the ones to talk to.

HAIGH: So why weren't you involved with it? Had you lost interest in the topic?

LAWSON: I don't know. I just have to kind of guess, going back. I think the computers at JPL that were being used for engineering and scientific computing were not the ones that people were talking about the most as needing these higher-level BLAS. Also to use the higher-level BLAS required a major redesign of the routines that would use them. So if one had a Gaussian elimination routine or a Householder least squares routine that had been written with no BLAS, you could put the level one BLAS in by just locating that inner loop and putting the call in. So it would be really straightforward and didn't require any new analysis. But to put in a level two and level three BLAS required developing so-called "block algorithms." So you had to restudy the Gaussian elimination and householder transformation and devise ways to work with blocks of the matrix instead of individual elements of the matrix. That was really a major algorithmic redevelopment effort. And it just was not something that struck me as something I wanted to get deeply involved in. It wasn't something that was close enough in payoff to JPL usage that I would have thought was a good thing for me to put my time into. This work was done in connection with a complete redo of the LINPACK functionalities to produce the LAPACK package, and so the BLAS 2 and 3 I don't think can be thought of independently from the LAPACK project. That was largely a new group of people who were willing to jump in and do all that design of the block algorithms.

HAIGH: So what kinds of systems and hardware was JPL using by the 1980s?

LAWSON: There was an internal project between Caltech and JPL and Intel (it went by various names but Hypercube was the name that was often used) to design a multiprocessor computer based on a hypercube topology, and that work was done through the middle 1980's I guess. I don't know how much actual production computing was ever done at JPL on hypercube computers, but it certainly was an interesting development for the people that worked on it. It probably contributed ideas to other multiprocessor computer system developers, so it probably had its place in the evolution of multiprocessor computers. But as a production machine I don't know how significant it was. Intel I guess did actually produce such machines then and sell them, but I don't know how many and I don't know how long that lasted in the evolution of multiprocessor computers.

HAIGH: I believe the final product was called the Paragon. They made a reasonable go of it but then I'm pretty sure that they closed the division down sometime in the mid- to late-1990's.

LAWSON: Probably right at the end of the '80s JPL did get a Cray T3D multiprocessor supercomputer, and I did have an opportunity to do some work on that, which was interesting, and apply it to a particular JPL project involving gravity studies. But I wouldn't say that characterized what was the typical computer being used at the lab because we only had one of them and there were a fairly strong number of engineers and scientists that were trying to use it. So the typical computer was the Sun workstation, or the IBM workstation, the 6300 or 6000 something?

HAIGH: RS/6000? Anyway, it was the early IBM RISC workstation.

LAWSON: Yes, so other RISC-based computers. The Sun seemed to be everywhere around the lab in the middle to late '80s I guess. And of course those machines had the instructions and data cache in them, and that would make them candidates to make them benefit from a blocked linear algebra algorithm. But the magnitude of that LAPACK type thing was such that, like I said, it wasn't something I felt like I wanted to get involved in.

[Tape 3 of 4, Side B]

LAWSON: I might just say I think we said it before, but just so it is not forgotten, that I certainly was aware that there were other projects very much like the BLAS 1 project. There were projects that were done I'm sure by NAG and by IMSL and possibly by others for use for their purposes. So I'm sure NAG had a set of routines with a functionality like the BLAS and essentially for the same reason. But they didn't want to wait until 1979 to see what we came out with, of course, so they did what they needed and used it. So there were other similar projects certainly that were done around that time.

HAIGH: So moving back to the topic of libraries produced within JPL. In the earlier session I talked to you about your recollections of what kind of library of internal software there may have been and where it was coming from. You had said very much that you thought that the internal library that developed at JPL in the '60s in a fairly informal way would have been largely produced in-house. I'm just looking a little bit in your memoir while you talk about this era. You

do say that the group contributed a small number of subroutines to a Fortran IV math library on the IBM 7094.

LAWSON: Yes.

HAIGH: Do you remember anything more about that?

LAWSON: No [chuckles]. In reviewing things for this interviewing, I found some really administrative type memos where maybe I had written a memo to my immediate supervisor or something like that and just said some things that we were doing for Fortran IV library on the IBM 7094. But I really don't know what was in it, and I know it certainly couldn't have been much and it couldn't have been too well organized, so I think if anything it was mainly just getting subroutines that those of us in the group had that would useful to other people, at least identified it some place where it would be easy for us to find them and give them to somebody if they could use them at the lab.

HAIGH: So when you say you contributed them to the Fortran IV library, do you mean that you would give them to IBM and they would redistribute them with the compiler?

LAWSON: No, that would just be a local thing within JPL that we would have had a Fortran IV library that we used within JPL. We're talking about things pretty far back in time and I could be wrong, it could be that we did submit something to IBM but I don't have any recollection of that. I don't think we did.

HAIGH: Now you also mentioned briefly the PORT library produced by Bell Labs. Do you know if any of those routines were ever used within JPL?

LAWSON: Well the people that developed the PORT library did separately publish a paper—probably in TOMS, I may have referenced it there—on routines for storing machine constants like the precision of floating point arithmetic and the overflow limit and things like that. The PORT library as a whole was kind of a restricted distribution thing like the Harwell Library, and I never really was fully aware of what was in it and I didn't ever see any great motivation to get the whole PORT library. But they were astute enough, or generous enough to the rest of the computing world, that they did separately publish a couple of things in TOMS that were very valuable that they had developed really as parts of their PORT library project, I guess, but they must have realized that these parts might be interesting to a lot of people that might not be interested in the whole library. So one of the important parts that I am thinking of was the set of machine constant routines that had names like R1MACH, D1MACH, I1MACH, which gave you the real and double precision and integer constants. And so we did use those, and in a sense they were like BLAS in that the specification was probably more important than the code itself, because you would have to get in and change the numbers in there if you used this in a new machine. But that's fine. They had specified that if you called R1MACH with an argument of three it meant maybe you wanted the machine precision. Anyway, it was specified what it was you were going to get. So if you got yourself a new machine, you knew what you had to do to adapt that routine. So in fact we were happy to follow their lead in that.

HAIGH: When you talk about storing those things as constants, would the idea be that that would make the code more portable because to run on a different machine, you can just change the constants?

LAWSON: Exactly, yes. So this was very much in the same spirit as the BLAS, the idea that some larger program that you'd written in Fortran would not have to have this number stored as a constant directly in the program, but it would just make a call to R1MACH with an argument of three and it would get the right number for that machine on the assumption that somebody had written that R1MACH routine correctly for that machine. So it's the same idea as we had on the BLAS. Eventually, the language people specifying the Fortran standard and the C standard, built these things into the languages. C, I guess may have had something like that really from the beginning, but Fortran didn't have it even in the Fortran 77 standard, so I don't think Fortran had that until the Fortran 90 standard, that you had a built-in function you could call to find out what the machine precision is or to find out what the overflow limit is or things like that.

HAIGH: For the internally developed libraries, you mentioned that there's a document, JPL D829, 1975 that described the final version of the internal library for the Univac 1108. Do you know if there are any other documents or records surviving that would document the internal libraries?

LAWSON: Well, there should be. There are things that I can look through that I might need. These kinds of administrative memos might show that we started at a certain point or that we hit version 3 at a certain point or something like that. There should be manuals, but I didn't turn up any in my personal collection here in preparing for this. They may or may not be in the archives, depending on how they were classified, whether they were classified as formal JPL reports or more informal ones.

HAIGH: All right, well that seems to bring us up to the MATH77 library that you discussed.

LAWSON: The problem, up until 1977 let's say, with the Fortran language was that the existing standard was the 1966 standard, which didn't treat some things that you really need to know if you're going to write portable software. So different vendors—Univac, IBM, Digital, and CDC—had to make their own decision on what feature they were going to put into their Fortran compiler to fill this need. They each made different decisions. And also, you didn't have the wonderful IEEE standard for arithmetic, so word lengths were different on different machines and the number of characters stored in a word would be different. Nowadays, we just assume that a byte is eight bits. Well that wasn't true on all machines by any means until we had the IEEE standard, because on the IBM 7094, we'd had a 36-bit word and the characters were stored in six bits each, so you had six of these 6-bit characters stored per 36-bit word. The machine was word addressable; it wasn't byte addressable, and that was typical of the Univac 1108 and the VAX, which had a different word length and a different number of characters per word. These things all worked against portability. So a combination of the Fortran 77 standard and the IEEE standard for floating point hardware really cleaned up the field. Any library work we did before and that anybody did before 1977 was really hampered by these problems. Along about 1975, to take a couple years before the 77 standard—which I think really carries the date of '78, but we always called it Fortran 77— was pretty well set somewhere around '75. We started around then rewriting our routines with the idea of the Fortran 77 standard. Again, looking back in time, it's always shocking to see how late things really did happen. It may have been 1985 before Fortran

’77 compilers could be assumed to be on all your machines. So although we may have started on the idea of this Math ’77 library around 1975, I guess our first edition didn’t come out until in the ‘80s sometimes.

HAIGH: Who was involved that project?

LAWSON: Primarily Fred Krogh and Van Snyder and me. We had other people in the group through those years that were not as oriented toward general-purpose software as we were. They did more specific project-type things. For some periods of time, we would have one or two people as kind of general programmer helpers in the group and we would have them work on some things to do with this library. But it was primarily Fred Krogh and Van Snyder and me who designed the routines and wrote them and checked them out, for the most part.

HAIGH: In the memoir, you mentioned that Release 2 appeared in October 1987. Can you remember when Release 1 happened?

LAWSON: I’m sorry, I wasn’t able to nail that down. It should be possible to do that, looking at some of these administrative memos more closely, it should be possible to actually find an instance of the first manual, but I wasn’t able to do that.

HAIGH: But it’s your impression it would be probably in 1985 or 1986?

LAWSON: I would think so, probably. It’s surprising to me, looking back, that it wasn’t sooner than that, but that may have been what it was.

HAIGH: Do you know what the main areas covered by the library were?

LAWSON: I have copies here of the last version that came out. I was looking at it this morning, in fact. It’s linear equations, ordinary differential equations, polynomial root findings, special functions, sorting, minimization of nonlinear functions, root finding for nonlinear functions, random numbers with some different distributions. So that’s kind of the general areas. Some of these routines were quite unique, and there’s probably nothing quite like them anywhere else. I think I mentioned, maybe in our interview yesterday, what we called the French Curve program, which was a one-dimensional curve fitting program that allowed for specification of equality constraints and inequality constraints, and these could be on values or on derivatives or eventually on areas. It gave great flexibility, and I’ve never seen anything like that published that I know of. We never published this, either.

The routine for solving ordinary differential equations, was due to Fred Krogh. It had been developed with the needs of the orbit determination people, so it had quite a few special features in it. People talk about *t-stops* and *g-stops* in ordinary differential equation solvers, or at least we used that terminology. A *t-stop* meant you wanted output of the solution information at a certain time, and you could have a list of those times, so you wanted it output at certain times. A *g-stop* meant that you wanted output when a certain function of the solution reached a certain value. That could be used in a variety of ways, but typically in orbit determination you might want to know when does this spacecraft make its closest approach to Mars, so you would be computing distance to Mars at each step as you went along in the solution. That distance, let’s say, got smaller and smaller and then started getting bigger again, then the algorithm would automatically

do a little iteration back and forth there and find the point at which that distance was closest and output solution values at that point. So these are features that were developed particularly for the specific needs there at JPL.

Also, some flexibility in the form of the output. For efficiency in solving ordinary differential equations, you usually want a variable step capability so that the algorithm could take long steps where the solution is smoother and shorter steps where it's not. But you may want your output to be at some equally spaced points for convenience in something else you're going to do, and so there was capability for doing things like that.

HAIGH: Did you adapt the code for some of the routines from things that were available in other libraries or had been published in places like TOMS?

LAWSON: Not very much. We used eigenvalue routines from EISPACK, so we did use those. I think eventually we did use LINPACK routines for the Gaussian elimination. We used least squares routines from the Lawson & Hanson book, of course. Van Snyder worked on most of the special functions, and I think he used code that had been originated by other people, sometimes I think, getting it by direct communication with them. Some may have come from TOMS. So at least I know we did make use of code from other sources in some of the special functions.

HAIGH: It's interesting that even into the late '80s that you were so reliant on internally produced libraries. Were the IMSL or NAG libraries available to use as at JPL?

LAWSON: They were certainly on the market, and I think some individual groups or sections did lease those libraries. Our library, which was probably about half to a third the size compared with either NAG or IMSL in terms of the number of capabilities provided, was freely available to people at the labs. I imagine people would use what we had if it covered what their needs were. If they needed more than that or if they had a feeling that there was more permanent professional backing behind NAG or IMSL and they really wanted to use that kind of library, then they might get that library. I know that both those libraries were used by some people at the laboratory, but neither was ever bought as a lab-wide resource.

HAIGH: Did you ever have to justify to the lab management the need to focus resources on making your own library instead of just relying on the increasingly mature, commercially available products?

LAWSON: Yes, I think we did. That probably relates to a question of why our group went out of existence in the early 1990s [chuckles]. I know we realized that there were other resources available—NAG and IMSL, and then particularly, MATLAB. Both NAG and IMSL still require a user to write their program and read the library manual to understand what the interface is to a subroutine and things like that. MATLAB really cut through all that very nicely, and I suspect was probably fairly widely used. I don't know how widely, but I would guess pretty widely used. So yes, we realized that there wasn't a continuing need for JPL to be developing their own libraries beyond a certain point. I retired in '96, and my colleague Fred Krogh retired about two or three years later. At that point, he negotiated with JPL to have the right to basically own that library himself and manage any further distribution or sales of code from it.

HAIGH: Was the library intended to be portable across the range of machines in use of JPL?

LAWSON: It was very definitely intended to be portable to any machine that supported the Fortran 77 standard. We took that as our mantra from the time we started thinking about it, probably back in 1975, and we made a strenuous effort to meet that goal. As far as I know, we did. I don't remember any terrible surprises coming up where the library wasn't usable somewhere on a Fortran 77 system. In the last few years, we actually went through the process of making a C language version of it. As I say in the write-up, we obtained a commercially available converter program that converted Fortran 77 to C. The C language had not been standardized by ANSI until about 1990, and we targeted that version of C, which was in some ways a little different from the Kernigan and Richie C that everybody was familiar with from 1970. But we targeted that, and we did convert the library and checked it out in that form. I don't know how much use was made of that by people. I know some use was made, but I don't know how much.

HAIGH: I know that portability had emerged as a major concern of the mathematical software community by the mid-1970s. As you worked on the Math '77 library, are there any techniques that you think that you refined or invented or did interesting things with in terms of achieving portability, or were you just following techniques that were well-established by that point?

LAWSON: That could be quite a big topic, and my colleagues Fred Krogh and Van Snyder might have more to say than I would on that. We actually developed three or four software tools—I mentioned some of those in the write-up that were aids to developing portable or testing portable software. There was a period of time, I guess it must've been in the '80s, when there was a kind of a collaborative effort that involved NAG and particularly Fred and Van from our group, and Leon Osterweil, who was I think, at the University of Colorado at that time and then went to University of California, Irvine. He'd been kind of a colleague of Lloyd Fosdick at the University of Colorado. There was a kind of collaborative effort to develop a set of software tools to aid portability. That project didn't really bear fruit as a collaborative effort. NAG had really some special ideas about what their needs were and had the resources to do what they wanted to do, so the involvement of our two people in it didn't mesh too well. So I think NAG ended up doing what they felt was useful to themselves and the project kind of withered away. I think Fred and Van tried to continue doing some things that they had wanted to do in that project back home at our shop, but those didn't reach a payoff either. There were some ideas on trying to get a higher-level way of specifying algorithms that could then be preprocessed into the conventional languages, but it was too much to try to undertake.

HAIGH: In terms of the techniques that you did actually deploy for the MATH77 library, do you think that there was anything innovative?

LAWSON: There were some things that we definitely found useful for our purposes, but they were probably so specific to the way we looked at things and the way we wanted to do things that we didn't make any effort to broadcast them, particularly to a wider community. Actually some talks were given at conferences by our people. We mentioned one talk at least about the so-called *specializer* that Fred Krogh gave at one of these portability conferences. I think JPL hosted three separate meetings over a period of three to five years on subjects of software portability, and I think we made a contribution to the field by holding those conferences. I'm not sure that we made a contribution by anything we did as individuals and the ideas we might've put forth, but by bringing people together, I think we contributed. The first one of those meetings

was on the subject of structured Fortran preprocessors. There was a spurt of interest in that kind of thing. I have the dates in there [referring to paper]. Is it around ’74 or ’75?

HAIGH: Yes, the workshop on Fortran preprocessors for numerical software in 1974.

LAWSON: Okay, 1974, we organized this conference on structured Fortran preprocessors. The thing that brought us to do that was that an engineer at JPL, John Flynn, had in fact developed such a thing. He was not in our group at the time; he was in what was called the Navigation Group. That was a group that did the guidance of spacecraft. He was very much taken by the ideas of structured programming and wanted to get rid of the GOTO’s in all their code, and he came up with the idea of a preprocessor. So you’d write “IF THEN… ELSE” and “DO WHILE” and things like that, which were not part of the Fortran language. Then his preprocessor would read that code and turn out regular Fortran that did have GOTO’s in it, and then you could run that through your compiler. But the idea would be that the person writing the code or maintaining it would only look at the SFTRAN code they were writing, which would be cleaner and more structured looking. He developed the whole idea and developed a preprocessor in a remarkably short period of time—I think just a few months. He was not in our group at the time, and I wasn’t necessarily aware of it right from the beginning to the end, but once I did become aware of it, that’s the impression that I got, that he’d done it remarkably quickly. We thought it was an idea that we would like to pick up in our group. And then because our group had more flexibility to do research and development-type things, management moved him into our group, so then he was in our group after that and we did further development of that.

But at the same time, around the country at least—maybe around the world, I don’t know—other people were having that same flash of inspiration. The most widely known example of that probably is what’s called the Ratfor processor that Kernigan wrote at Bell Labs. But there were at least two or three or four other examples of things like that. There was something called MORTRAN and there were some other things. So we contacted all the people that we knew that had done things like that and encouraged them to get together for this conference, and encouraged some members of the Fortran Standards Committee to attend, because this was at a point where they were kind of nearing closing the book on the Fortran 77 standard. Say around ’73 or ’74, their feeling was they didn’t want to think about any big new ideas; they wanted to just polish up what they had. But we got some of them to come to the meeting. My personal feeling is that we did have an impact on the Fortran standard from that, because after that meeting, the Standards Committee did decide to put an “IF THEN… ELSE” in the Fortran 77 standard, which was not there previously. There’s other things they could’ve done, but at least they did that much, which was good. So I feel that by holding that meeting, we contributed something to progress in the programming language that people in numerical computing were using.

HAIGH: Do you know if there was ever a proceedings issued from that workshop?

LAWSON: Yes, there was a proceedings of some sort here. I don’t know if it was as a separate JPL publication or as a special issue of the SIGNUM newsletter, but I think there was definitely a recorded proceedings of that meeting. And then there were two other meetings that we held, at one- or two-year intervals after that at JPL, that in a sense, were followed on to this, one on portability of software, and one on environments for developing portable software. So we continued to try to get people together that were thinking about these things.

HAIGH: I'll ask you a little more about those later in the context of SIGNUM and the mathematical software community more generally.

Now I just have a couple of follow-up questions on MATH77. I think you've talked about how it was produced, what was it in, how it was used internally. I'm wondering, did any of this code make its way into the world beyond JPL?

LAWSON: Not very much, I don't think. I don't know. I know there were individual requesters at other laboratories, or government laboratories, or large companies that asked for some of the things I had done on triangular grids, and I'm sure there must have been people that requested things that Fred had done on ordinary differential equations. So there probably were some individual routines that were sent to individual requesters, but I don't think anything was really published generally from that except what was in the Lawson/Hanson book on least squares. Yes, I think that's about it.

HAIGH: It's interesting that you would put in all this work of many years to develop a high quality, well-documented portable library, and then not be interested in pushing that into the world and trying to see if there would be other sites that would be interested in using it.

LAWSON: Well, I think the work of other places, both commercial and laboratories, kind of overtook the need for that. As I say, there are some routines in our library that are still, I think, quite unique. Maybe we should go back and think of submitting those to TOMS. But as far as libraries go, it's developed into too much of a mature type of product now for us to try to jump into it, I think. As I said, Fred Krogh negotiated to have the right to do further distribution of the library and he does maintain a website, *mathalacarte.com*, and people can obtain routines from him, but he doesn't do any aggressive advertising of that so it hasn't had much impact, probably.

HAIGH: I also know that much of the code in the NAG and IMSL libraries was contributed by academic authors. Did you ever give any thought to contributing anything to any of the other libraries?

LAWSON: Of course, NAG started out as a cooperative effort between a number of universities. So that was their mode of operating right from the beginning, and I presume that Brian decided it was worthwhile to keep very close contacts with the academics, even as the years went by. IMSL, I don't know how they really worked in the beginning, but of course, Dick Hanson, who had worked closely with us, had been a member of our group for a number of years, then spent a number of years with IMSL, and I know he has corresponded with Fred Krogh about some routines and Fred has done some work on things that were intended to go into the IMSL library.

The whole business of JPL work getting put out in the commercial world, would be kind of complex legal issue. I don't think we really wanted to get into it until finally at the end, JPL just said, "Well, you can take the whole library. We don't care." But for us back in, say, 1980, to have said we want to give this routine to a commercial organization like IMSL or NAG would have involved a lot of dealing with lawyers, I think. So we never approached that kind of thing.

HAIGH: So you think difficulties in terms of technology transfer in intellectual property would be the main reason that you wouldn't see that?

LAWSON: I think so. In all your interviews, I don't know if you've looked into the Mathematica company and product and original history, but that goes back to Caltech and some intellectual property issues. There are some stories to be told there, I think. I don't have any firsthand information about it, but there were things that were developed there at Caltech and different people had ideas about how that should get into the commercial world.

HAIGH: You do mention towards the end of your memoir that at least the SFTRAN 3 code was used by Nelson H. F. Beebe at the University of Utah.

LAWSON: Well, Nelson Beebe is a story in himself, and is somebody that you might want to contact in tracing the history of math software who is not thought of as being kind of officially in the math software community, but had all the same thoughts and took all the same kinds of actions that people do that think they're in that community. He was a chemist by training. There was some kind of matrix that chemists in his specialty were all interested in, and they needed to communicate with each other and send copies of these matrices to each other. He just had an innate sense that things ought to be organized. He didn't like to do things or even like to see everybody doing things ad hoc ways, so wherever he saw that was going on, he just seemed to have an innate feeling that he had to get in there and organize it. So he developed some standardized magnetic tape formats and tried to promote the use among his chemistry colleagues for transmitting numerical information between each other and across different machines. He had a real sense of trying to achieve portability in a lot of areas, such as tape transfer, besides just the issue of writing computer programs.

He developed a very extensive graph plotting package. I think the name on it was PLOT79, which suggests it came out in 1979. In those days, portability was just essentially unheard of in the plotting area. It was just assumed that if you were going to have to plot, you were going to have to deal with some very special set of equipment and your interface from Fortran was going to be weird. He came up with a very portable set of plotting routines that would interface to a lot of different actual pieces of equipment. I think it was probably pretty widely used, at least in his chemistry community. But along the way, he found out about our SFTRAN and that appealed to his sense of orderliness also, and he picked up on that. He's a fantastic programmer, and he just jumped right in and made a bunch of changes in SFTRAN to have it do some additional things that he thought it ought to do, and then he made it a part of the things that he distributed. He might be a person that your project would want to talk to if you wanted to get a little bit beyond the people that really thought of themselves as being in the math software community and were really promoting the portability of math software and data in some particular application communities. He is still at the University of Utah and he's got a website that I mention there in the write-up.

[Tape 4 of 4, Side A]

HAIGH: Session three begins on the afternoon of the November 7, 2004

Having finished talking about the software tools and libraries developed at JPL, now is the time to turn to your involvement in the mathematical software community more generally. A good place to start might be to discuss whenever it was that you first started feeling that there was a community of people interested in numerical analysis that you were part of, which may have been during your time as a graduate student or perhaps a little later. After we've talked about

your sense of the community of numerical analysis in general, we can talk more about the subcommunity concerned with software packages.

LAWSON: In my graduate student days at UCLA, I don't think I got much of an impression of a community of either numerical analysis or computational mathematics, and certainly not mathematical software. The UCLA math department had not tried to have a subgroup of specialists in computational mathematics. People of very high standing had been associated with the Institute for Numerical Analysis, associated with UCLA in the middle '50s, but that had gone downhill in terms of funding and in terms of senior personnel by the time I arrived, and I didn't really even know anything about the history of what had been there. So there was no particular notice taken of computational math in the Department, so we didn't have any even visiting speakers in that area, particularly. So I didn't get much of an awareness of community. I mentioned a book in my write-up that was the proceedings of a conference at the University of Wisconsin Math Research Center that was held while I was a graduate student there at UCLA, but I wasn't aware of it when it was held. It was on approximation theory and attempting to address what approximation theory would have to contribute to computing since computers were starting to come onto the scene. I did get some sense from reading that book that there was something going on out there, and I saw the names of some people like Alexander M. Ostrowski, who I just in the last month found out was the thesis advisor for my thesis advisor, Professor Motzkin. So I saw some of those names, but I didn't know much of anything about any of them.

HAIGH: So you feel, then, that it was when you came across the proceedings of this 1958 meeting on numerical approximation that was held in Madison that you first got the sense that there was a community of people out there working on these kinds of topics.

LAWSON: Yes, I think that's right. As I said in the write-up, there was another book by a Cecil Hastings that I saw during my graduate student days that was kind of another side of the coin. He was a kind of freelance consultant on curve fitting, and had done some work for Douglas Aircraft and I think for Rand Corporation. And as I said in here, I think his methods were relatively primitive and he wasn't applying much theory, but he was producing some curve fits that seemed to be useful to people. I kind of put those two things together, the Hastings book and the Wisconsin book. The Wisconsin book showed that there was some theory out there that was kind of ready to be applied, but people weren't organized as to how to apply it; and the Hastings book showed me that there were people that wanted this kind of work done in applications, but they didn't really have all the theory they could have had to apply it. So I kind of connected those two and felt that there was a place in the world for somebody to try to understand some of the theory and methods and approximation theory and communicate those to the engineers and scientists in a way that they could use them.

HAIGH: Do you think that problem with communication was ever satisfactorily addressed?

LAWSON: I feel that's what my career was [chuckles]. If it could be characterized in a small number of words, I felt like I was bringing some knowledge of computing methods, with some specialty in approximation methods maybe, into the hands of engineers and scientists that could use them. That's really what I feel like I was trying to do and what I think I did do, to some extent, at JPL.

HAIGH: So you think JPL was more successful in dealing with that problem than most places?

LAWSON: Well, I think our little group did that, to the extent that three to eight people can do it, and we found the engineers and scientists receptive and happy to have some guidance when they needed it. So I think it was reasonably successful, yes. And many, many other people in many other ways were accomplishing that. But I'm going to guess that you were asking me to relate what I did to what the larger computational math community was doing, and I think the whole computational math community and math software community really has that goal—they're trying to bring the best algorithms that theorists have developed into a form that they're easily usable by the engineers and scientists. So I saw that as my goal and my role, and also I projected that, I guess, on the rest of the community I ended up working with. I felt that's what they were all doing.

HAIGH: Now, you also say in the memoir that the introductory survey paper from the Wisconsin meeting by Ostrowski is very interesting in showing both explicitly and implicitly, the clash between pre-computer and post-computer views of approximation theory and computational methods. What do you think the biggest shift or clash between these two different ways of looking at the subject would be?

LAWSON: I guess it comes down to algorithms and software. These people had dealt very little with algorithms, if at all, and essentially not at all with software. This was, what, 1957.

HAIGH: '58 is the date for the conference, and the book appeared in '59.

LAWSON: So in '58 when they had the conference, they could not have had any significant experience in actual computing, and so their experience in deriving algorithms was probably pretty limited too, because they wouldn't spend a lot of time developing algorithms if they couldn't do something with them. So I just felt there was a place for people to develop usable algorithms and implement them in software.

HAIGH: When was it that you first made personal contact with other members of the emerging computational mathematics community?

LAWSON: I think probably the first meeting at a national level, and there was some international participation, that I went to, was one on approximation theory at Oak Ridge, Tennessee at Gatlinburg, and as I mention in my write-up, I am unclear as to whether that should be regarded as one in the celebrated Gatlinburg series. I think maybe not. This was a conference on approximation theory and what is usually referred to as the Gatlinburg or Householder Series is on linear algebra. So whether this was regarded as in that series or not, it was a meeting. I do remember meeting Lanczos about whom I knew nothing at that time, but it was just interesting to see somebody who was quite a few years older than I was at that time who was willing to talk to a youngster like me, and I found that encouraging. I'm sure I met some other people at that meeting; I don't remember exactly who by name except for Lanczos. I don't know if John Rice was at that meeting. He very well might have been. I don't know when I first met John, but I think I would have to say that I feel he was probably the first individual to have a major impact on bringing me into a wider community. The meeting that I mentioned there at National Bureau of Standards about methods for computing elementary functions that eventually led to writing the computer approximations book was probably the next key meeting I went to. I think there was probably a national, ACM or SIAM meeting in Los Angeles along the way there too that I

attended, but I don't remember too much about that specifically. {ACM National Mtg. in Los Angeles, 1962}

So I certainly appreciate John Rice bringing me into the group that had that meeting at NBS and went on to write *Computer Approximations*, put me in touch with fellow authors on that and other people, and gave me more of a sense of a project that was intended to serve a wide community.

HAIGH: So then you're thinking that it was after that meeting that you already discussed at the National Bureau of Standards around 1964 in the beginning of the project that produced the *Computer Approximations* that you began to feel yourself really being more of a part of this community?

LAWSON: Yes, I would say that, yes. And then the other individual that I want to be sure and mention as having had a significant effect on bringing me into my career would be Gene Golub, first of all by his writing a couple of very good papers with coauthors on least squares computation and the singular value computation in 1965, and then inviting Richard Hanson and me up to Stanford to talk about it sometime around 1970, I guess that we've talked about before, and encouraging me then to write the book on least squares with Richard Hanson. And Gene has, I think, seen that I got invited to some of the Householder meetings and has done other things along the way that helped me get acquainted with people and move along in my professional career.

HAIGH: So how would you describe Golub?

LAWSON: Well I think he is exceptional at disseminating information around the country and around the world in technical information in this field. I mean it could be compared with a honeybee that goes from flower to flower and takes the pollen from here to there. He does a lot of traveling, he meets people, he's known by everybody in numerical and linear algebra, he really gives very gracious attention to young newcomers into the field—I've seen him do that over and over again. And of course he stands on his own for the research he's done and the papers he's authored and coauthored as contributions to not only numerical and linear algebra, but his own background and training was in statistics, and from time to time you see him go back to that and write papers that show how linear algebra would apply to some statistics problem in a way that people hadn't thought about it before. And the books he's coauthored with Van Loan on matrix computations have become absolute standards for that topic, so he's just a giant in the field as far as contributions go. [Gene H. Golub & Charles F. Van Loan, *Matrix Computations,* Johns Hopkins University Press, 1983]. Well the other thing, I have the impression that the people who have done their Ph.D.'s under him all have a kind of family feeling with him and with each other, and that's a very positive contribution to the field too.

HAIGH: Another thing that you discussed at some length in your memoir is John Rice's role and your own participation in the famous series of mathematical software conferences held during the 1970s.

LAWSON: Yes. Well I mentioned that I feel John Rice really brought me into a certain part of the picture by getting me involved with this computer approximations book, but then he has always been good at looking into the future and conceiving of what would be valuable

downstream. We can talk more about Brian Ford with that characteristic too. But speaking of John Rice, I think he really saw the need for a journal about mathematical software and more attention to that as a topic. I know from a kind of mundane point of view, I think he had expressed the feeling that a professor in a university gets a certain amount of credit for publishing papers, but doesn't get any credit for publishing a piece of software, and he thought that wasn't right, and by establishing a journal that published software, he hoped that would change that situation to some extent. But of course that's just one aspect of what he was thinking of; he really wanted a vehicle for people interested in math software to communicate with each other. And as I say in the write-up, as far as I know, he may have coined the term mathematical software—maybe he did, maybe he didn't. But he held that first conference on math software and then a second one and then a third one, and then with other people, established the ACM Transactions on Mathematical Software, so he's done a lot to encourage communication and encourage the identity and recognition of that as a field of work.

HAIGH: You were one of the members of the organizing committee for the first conference. Can you remember what that involved?

LAWSON: I really don't. I saw my name on a list here. I don't remember, I'm sorry to say.

HAIGH: Do you have an idea of why you might have been chosen?

LAWSON: Well, what year was this?

HAIGH: The conference was held in 1970, so I assume that the organizing committee would have probably been active in 1969.

LAWSON: I don't think there would have been much that I would have been known for except my involvement with the computer approximations book. So I would think it probably related the fact that I had worked with John and the others on that book. At a more technical level, I know John was interested in one of the parts of my Ph.D. dissertation about computing minmax approximations as a limit of weighted least squares approximations, and he gave an exposition of it in one of his books that he wrote after that and he had a couple of students do some work on it, so he may have thought that some of my work in that direction was worth thinking about when he was pulling together his committee. I don't know.

HAIGH: Do you recall anything about the meeting itself?

LAWSON: Nothing very specific. I think I do have the proceedings from it, but at this point I don't remember it precisely. One of those meetings—I think it was one at Madison probably, and the first one was at Purdue I think and the second one may have been at Madison?

HAIGH: The first one was at Purdue. I just found in your memoir that the third mathematical software conference was held at Madison. The second one I would guess was probably at Purdue as well, although I don't have any information on that at hand.[6]

[6] The Mathematical Software II conference was held in West Lafayette, Indiana (home of Purdue University) on May 29-31 1974. No proceedings volume was published, though much of the material from the conference was used for Volume 1, Number 1 of the ACM Transactions on Mathematical Software.

LAWSON: Well anyway, one of those meetings he did invite Cecil Hastings to come, and I remember talking to him about that. I didn't know anybody who had ever heard of Cecil Hastings except me, and it was interesting to know that John had come across his book also and was just kind of fascinated with the idea of seeing who this was and meeting him. Because I think Cecil Hastings was living in Hawaii at that time and doing his consulting for Douglas and Rand Corporation I guess by mail, or maybe he would travel over from time to time, or maybe he had finished doing his work and was retired. But we were all fascinated, anyway, by this man that had made a living out of curve fitting and was living in Hawaii—that all sounded kind of attractive.

HAIGH: Are you aware of any other important members of the community that you might have met for the first time at one of these meetings?

LAWSON: Well not really as a special memory. By going to various meetings I met various people along the way, so I don't remember any in particular.

HAIGH: One thing I'm trying to get a sense for is do you think most of the contributors to these conferences would have known each other prior to the meetings, or do you think that the meetings would have played a more important role in introducing people to each other?

LAWSON: I think that probably the first numerical software meeting might have been bringing a number of people together who hadn't met each other before. I'd have to look at that proceedings and kind of guess from that. But these were younger people, I'm sure, like myself at that time, so it might have. And the numerical software community had not been a community that had had meetings, since this was the first one of that title, so very possibly there would have been a number of people for whom this was kind of the first chance to meet others working in the area. I'm sure there were many that were acquainted to some extent with each other before the meeting too, though.

HAIGH: You also mentioned the invitation for the second mathematical conference of Joel Moses reflecting field of symbolic algebraic manipulation. So I wonder if you could say anything about the state of development of the symbolic mathematical software community during the first half of the '70s, and to what extent there were any communications between the symbolic and numerical communities.

LAWSON: Well my impression or my recollection and my impression is that there were some efforts made in planning some of these meetings to bring some symbolic algebra people in because there was a feeling that there should be some commonality there and we could learn something from each other. But I don't have the feeling that much ever came from any of those. Now there may have been some individual contacts that were made that resulted in something further downstream, but for the most part, there does not seem to have been any notable development of numerical math and symbolic math together. I guess both Mathematica and Maple now have a significant amount of numerical computation in them, and I guess MATLAB, which is thought of as mainly numerical, has some symbolic capability, although I think it's by working with the people through Maple. So there are some packages that will give you some combinations, but there is not much of a community you could point to of people who have a strong interest in both directions.

HAIGH: Why do you think that is?

LAWSON: Well, this would be just pure guesswork, but I just think people have different inclinations as to what they're interested in or what they feel capable of working on, and it's just two different modes of emphasizing what you do with your mathematics, and there don't seem to be very many people who feel comfortable or strong in doing both.

HAIGH: Well we should certainly talk about the IFIP 2.5 working group, which I think to some extent may have been a continuation of some of the community that was built through these conferences. But first we should probably talk about SIGNUM, since that was started in the late 1960s. So can you remember when you first became aware that there was such a thing as SIGNUM now in the world?

LAWSON: Well in reviewing for this interview, and I looked through the collection of back issues of the SIGNUM newsletter that I have, and I fortunately do have issue number one, and I found it a little surprising to read through that. What was the date on that, volume one?

HAIGH: June 1966. And at this point, of course, it was called SICNUM, the "C" being for committee.

LAWSON: Yes. When I looked back at that I was a little surprised at how early that was. A lot of things I've looked at for this interview, I was surprised how late they happened. This one, I was a little surprised at how early it was, because '66 is pretty early in a math software world. But it was also striking that it seemed to be a one-man operation. Joe Traub is listed as the editor of the newsletter and nobody else is listed as any kind of officer of the organization, so it seemed to just be Joe Traub. On the request form there, if you wished to become a member of SICNUM, you just sent your name and address to Joe Traub, so it seemed like it was a Joe Traub organization. Well I think it's wonderful that he did that, and I'm sure he must have talked to some other people in formulating the idea and starting it, but it was quite an adventurous thing, I think, to just start out more or less by himself.

And in that first issue, he says that SICNUM has 1,000 members. I was thinking, "How did SICNUM have 1,000 members before the first issue of the newsletter came out. And was I a member or not? I got the newsletter, so maybe I was." So maybe he had done some recruiting at a national ACM meeting. I don't remember that, but you'll be talking, or you have talked to Joe Traub and can maybe fill that in. So it was interesting to me to see kind of what it looked like right there at the beginning and that it did start as early as '66.

So I was certainly receiving the newsletter, and then I noticed the date came when they finally converted from a committee to a group and had a board of directors, and then shortly after that, one of the members of the board of directors, who happened to be Tony Ralston, resigned and there was an opening and they appointed me to that position, so I was a member of the board of directors from some date there.

HAIGH: Yes, 1968 you say here that this was reported in Volume 3, Number 3, October 1968.

LAWSON: So it's reported in the 1968 issue that I went on to the board.

HAIGH: So the chronology on this, that's the same year that *Computer Approximations* appeared. Do you think that would be what would have brought you to their attention?

LAWSON: Well it might have been, and by that time some of the people involved in that would have known me and I would have been acquainted with them. I don't know when I first met Joe Traub. His field was to some extent approximation theory, and he might have been at that very first meeting I went to in Gatlinburg, I don't know.

HAIGH: I also see that you say that Volume 2, Number 2, June 1967 reports that you were the chair of the Los Angeles chapter, one of three local chapters of the SIG, the others being in Boston and Washington D.C.

LAWSON: Well that's interesting. Yes, the aerospace industry was quite active in the Los Angeles area through the '60s, and so we had STL and we had Aerospace Corporation, we had Douglas Aircraft, and JPL, and I think there were people from all those places who thought they were doing numerical analysis and writing numerical software—not necessarily software to be shared or put in a library, but were interested in numerical algorithms. So we had enough people to form a local group.

HAIGH: Now, can you remember if prior to this, your being an active member of the Los Angeles SIG for ACM as a whole?

LAWSON: Yes, there is a local chapter of ACM in Los Angeles, and that's interesting. I don't know. I very possibly would have been a member of the LA Chapter.

HAIGH: I know going back to the 1950's I think that was by far the most active chapter in the entire association.

LAWSON: Yes, they may have been kind of a focal point for some of the things we talked about yesterday where customer companies of IBM would develop things like the assembler or an input/output system or something like that, and I think the LA Chapter ACM could have been a focal point for discussing things like that and for people getting together on those projects. I was not involved in any of those projects, I know. But I don't know. I might have been involved. Somehow I must have been acquainted with some of the people who were doing those things.

I remember two names, Rube Kuritsky and Ed Manderfield who were active in the LA Chapter and active in starting special interest groups locally. And I think those two, neither one of whom is really primarily thought of as a numerical analyst, were just trying to get different special interest groups going, like SIGPLAN for languages and some others. And I think those two were really the ones who administratively got SIGNUM started, and they must have gotten in touch with me to get involved with it. And exactly why they would have known me, I don't know. It might have been from some UCLA contacts also. I don't know.

{Added September, 2005. See new section on Los Angeles SIGNUM added in Lawson's write-up.}

HAIGH: How did it work being a chapter of a SIG? Would you be a subgroup of the local chapter or would you act independently?

LAWSON: Well there was no heavy bureaucratic overhead involved in that. A special interest group of numerical analysis locally was just those people who were interested in doing it and they planned their meeting and they had their meetings and they didn't have to clear it with anybody higher up particularly.

HAIGH: So it was more of a grassroots thing. You never really had to interact with the national ACM?

LAWSON: No. I mean there must have been some informational exchange in order to have this show up in the SIGNUM newsletter that said there was a local group, but there wasn't any requirement that we tell them ahead of time what programs we're going to have in the next six months or anything like that.

HAIGH: Do you have any recollection of what the local group might have been involved in doing or how many people might have come to its events?

LAWSON: My recollection, we would meet at a restaurant and reserve a dining room, and we would have maybe a dozen people, and one person would have been asked to prepare a talk. It might be somebody who was a regular member of the group or it might be somebody from outside who had been asked to come and talk, and they would talk about some numerical algorithm or maybe some software development thing they had done in their work that they thought was of some general interest. I talked a number of times on various things I was doing on least squares approximation or the things to do with the books or contour plotting.

HAIGH: So just to confirm the chronology from that then, the notes in your memoir are showing that in 1966 you took part in a panel discussion for a national meeting of SIGNUM held in conjunction with the national ACM conference. So that's the first time you see your name in their proceedings. Do you remember anything about how large the general meetings of SIGNUM would have been that were held in conjunction with the national meetings for the ACM as a whole?

{Added September, 2005: My first participation in a national ACM conference was in September, 1961 in Los Angeles at the 16th National ACM Meeting. I gave a contributed talk on least maximum approximation via weighted least squares based on my thesis work.}

LAWSON: Well, I don't have a really direct memory of that, but I think anything like that was pretty much just like another session at the meeting. So if a national ACM conference had maybe 50 people or so in the audience at each session, then this would just be kind of like another session and you'd have 50 people maybe.

HAIGH: So it would really be a stream of programming that was sponsored by the SIG, essentially?

LAWSON: Yes. Like now we have the mini-symposiums typically at SIAM national meetings where one person will contact maybe four of the people that he or she is acquainted with who are working on a certain topic and get those three or four people to say they'll each give a talk and then you register this with the program chairman of the SIAM national conference. Maybe you say you'd like to put on a mini-symposium on such and such a topic. So I think a session like this

organized by SIGNUM at an ACM national meeting would be in that nature. I don't remember the term mini-symposium being used that far back, but maybe it was. I'm forgetting. It seems to me that's a term that came into play more through the '70s, but this was in the nature of a mini-symposium, as I recall.

HAIGH: So you joined the board in 1968. Can you remember when you left it?

{Added March, 2006: Lawson was one of a group of four who left the 12-person board of SIGNUM between January and November of 1971.}

LAWSON: I don't. You'd have to look through the newsletters and see. Probably when I became editor I probably moved off the board, but you would have to look at that newsletters and see. The board itself of SIGNUM didn't do very much. I mean they didn't have any meetings other than at maybe a national ACM meeting or possibly at some other related meeting like one of John Rice's math software meetings, but they certainly didn't have any meetings just on their own where all the board members are going to go to Chicago and meet because there wasn't that much going on.

HAIGH: Do you recall any decisions that the board ever made or any controversies?

LAWSON: No, I don't. From working at the newsletter, you can see that from early on there was an interest in making it more international, and they appointed some international representatives and they name about three people or four people in one of the issues, including, I think Robert Barnhill from the United States at University of Utah and three people from European countries as international correspondents for SIGNUM. The idea was to get people over there to notice what SIGNUM was doing and for SIGNUM to get some input hopefully from these people as to what was going on in Europe. So I'm sure the board must have had discussions and talked about how to facilitate that kind of thing. Then when SIGNUM would be putting on a session at an ACM national conference, I'm sure the board probably was involved in saying who was going to chair that and things like that.

HAIGH: So in 1972 you succeeded Cleve Moler as the editor of the SIGNUM Newsletter and then you held that position until August 1976. I think that would have made you the third editor of the newsletter. Was editing the newsletter, in terms of the actual amount of work required, the main post within SIGNUM?

LAWSON: I would say so, yes. The other posts only came to life once a year or twice or something. The newsletter editor was doing something, I guess it was quarterly, and of course pulling stuff together to do that, so that was a noticeable amount of work.

HAIGH: What did the job involve?

LAWSON: Well, standard newsletter things: trying to get people to write articles, thinking up ideas for new kinds of departments. For instance, the newsletter decided to carry announcements of kind of internal or informal technical reports from companies or laboratories or university departments. So sometimes a university department would send us a list of half a dozen technical reports that had come out of their department in the last quarter or something or in the last year or something like that. So we got some things like that going. I don't know if we got into the

business of announcing vacancies and employment opportunities; I don't think we got into that very much. We did get a little bit in the way of maybe a book review. But mainly it was little short articles on either an algorithmic idea or more of a thought about whatever was kind of a current interest at the moment. When they got into the IEEE standard work for the hardware, there were people who wrote articles in there. In fact, I think we sometimes carried kind of full drafts of things like that, and on a couple of occasions we'd have special issues of the newsletters that were really proceedings of SIGNUM meetings. So a variety of things like that.

HAIGH: Do you remember it being a struggle ever to find enough material to fill up an issue?

LAWSON: Well, I don't remember it in those terms particularly. At some point I decided to quit doing that as every editor ahead of me had, and part of the reason for that might have been I thought I had worked enough at it. So it is a task that takes some concentration and time. But mostly I think it was pleasant, and it certainly put me in contact with a lot of people, so this whole question of how did I get a feeling of community and get in contact with people, being the newsletter editor immediately put my name in front of everybody and put me in a place to be communicating with lots of people.

[Tape 4 of 4, Side B]

HAIGH: Is there anything you remember trying to do differently from the previous editors with the newsletter or anything you wanted to do to put a stamp on SIGNUM or try out anything that you thought was particularly important?

LAWSON: Well I don't remember anything right now. I think if I reviewed all those issues while I was the editor some things would probably come to mind. But basically I think Traub got it off to a great start and Moler did a great job of running it, and I was just trying to carry it on and if I could think of some other kinds of things to put in it, I would try. So I was just working along at it. And of course, people nowadays should compare it with the Numerical Analysis Newsletter that Cleve Moler edits and puts out weekly over the Internet because it was serving a lot of those same functions, but in those days we didn't have the Internet, so our speed of communication was measured in months rather than in weeks.

HAIGH: Now after you stepped down as editor in 1976, do you remember any further significant development with SIGNUM on your part?

LAWSON: Well these three conferences that we held at JPL were all done I think under auspices or with collaboration with SIGNUM, so these were regarded as SIGNUM meetings, and I think those were after I had finished being editor of the newsletter. I don't know what they're listed under.

HAIGH: The first of those meetings I think was in 1974 on Fortran preprocessors, so that would certainly have been while you were still editor. But the others were held in 1978 on Programming Environment for the Development of Numerical Software; and in 1981 with Fred T. Krogh as the conference chair, Conference on the Computing environment for mathematical software. So those two were later.

LAWSON: When I was doing anything I'm sure Fred was helping me with it and when Fred was doing something I was probably helping him with it, so we were both involved there after I was the editor.

HAIGH: So on a practical level, what did SIGNUM's sponsorship of the conference mean?

LAWSON: Well, it meant that it was appropriate to use the newsletter to publicize it, of course. And we would encourage people who regarded themselves as members of SIGNUM to come to the meeting, so it made it a little more than just a JPL meeting or something, because people said rather than that there's going to be a meeting on JPL on such and such, they didn't know why they should be involved, they might pay any more attention to it if you said it was a SIGNUM meeting. If they identified themselves with the interests of SIGNUM, then they would say, "Well maybe this is a meeting I ought to think about." But I don't think there was any monetary issue. It wasn't like getting the NSF to fund something so a lot of people can travel to a meeting. SIGNUM was just lending its name and establishing a context for the meeting, really.

HAIGH: So it seems that by the 1980s SIGNUM was beginning to lose ground to SIAM as the primary association for people with an interest in mathematical software. Do you have any recollection of that or any perspectives on it?

LAWSON: Well, like these particular meetings were fairly specialized in their topic. They were organized around the interests of people who were trying to develop basically portable libraries, and I don't think SIAM was doing anything that I can think of that particularly addressed that topic or that audience. So I think at least up through these meetings…

HAIGH: The last of which was in 1981.

LAWSON: 1981, yes, there was a purpose being served reasonably well by SIGNUM. But as SIAM developed more of a larger variety of journals, like when Gene Golub established the *Journal on Scientific Computing*, that really allowed SIAM to provide more of a home for people who were really interested in taking their math ideas into computing. So SIAM was kind of in the ascendancy in finally doing that kind of thing. But the whole nature of math software as a concept, particularly associated with the libraries, had its growth and its maturation through the '70s, and into the '80s. It was more of an established operation by NAG and IMSL and not something that people from so many different places were apt to be thinking about and trying to do. So it changed the nature of it. I think conferences like this probably had less general appeal because there were a smaller number of people just in a few places that were trying to do this kind of thing. And then the other function of the SIGNUM newsletter was pretty well superseded by the Numerical Analysis newsletter on the Internet when that came along. I don't know exactly when that started. I guess that probably started at Stanford, and then it was probably picked up by Oak Ridge and Mathworks I guess. But that must have been in the '80s sometime I guess that that started, and once that was going as a weekly electronic newsletter, then the quarterly SIGNUM newsletter was probably of less importance.

HAIGH: Now as you know, there were only three conferences held in the mathematical software series. Do you think it's fair to say that some of the energy of that community was channeled into the IFIP Working Group 2.5 as an alternative mechanism?

LAWSON: Well, the Working Group was relatively small, going from 13 people at the beginning to maybe 30 people by 1980, something like that, and it didn't attract a lot of attention. People who were not members of the group for the most part probably didn't even know it existed, or if they heard of it, they didn't have much of a notion of what it was. So it may have soaked up the energy of the people who were most interested in that kind of thing. But I think the other thing that I mentioned before was more important: that math software libraries, which had been something that a lot of different institutions felt they needed to do at their own institution in the early 1970s, by the early '80s or middle '80s were finding they were happy enough to buy that service from NAG or IMSL and weren't doing that locally, so there just wasn't such a dispersed group of people trying to do that kind of thing. So that lessened the need, I think, for SIGNUM.

HAIGH: Now in the memoir you discuss the origins of the Working Group, giving credit particularly to Bo Einarsson. I wonder if you could describe Einarsson.

LAWSON: Describe Einarsson. Well I think he's reminded me many times that he pronounces him name 'boo.' It's spelled B-O, and I always most naturally wanted to say 'beau,' but I think he says 'boo,' and he pronounces Einarsson different than I would, but I won't try to reproduce that. He is a person who just seemed to have a clear image of what he thought an international group like this interested in numerical computation could do and it would be good to have in existence to do. So he was very active in contacting IFIP to get the thing started, and I guess he was the first chair of it, and he continues to this day to be a very active member and maintains the website for the group and I think has spearheaded a project they've undertaken in the last year to put together a book about software. So along the way, he was interested in Fortran standards and I think he had written a book or two on the Fortran language that was intended to be a textbook for learning Fortran. So he is just a person who's very strongly interested in this kind of work. And he's personable, he's a pleasant person to be with and have dinner with and have chats with. He's a nice guy.[7]

HAIGH: Do you think that over the years he remained the key motivating force behind the group?

LAWSON: That could be. I retired from active involvement in the group in 1996 when I retired from JPL, so I haven't seen the group in action since then. I guess I'd have to say he's probably had the most sustained interest and activity level in it of anybody, as far as I can see. Some other people have popped in and popped out of the group over a few years, but he has really had a continuing interest.

I think for myself, coming from the West Coast here in the United States, I didn't have much concept, and I guess I still don't have much concept, of IFIP. But I think possibly for European countries and other countries outside the United States, IFIP was a way to plug into developments in the computer field generally whereas people in the United States felt it was all right on top of them here and they didn't need to look to an international organization that much. So when you look at any of the official documents of IFIP that list the president and the officers and the committees and everything, you'll see more people from other countries than the United

[7] Lawson adds: The software book mentioned in the above paragraph appeared this summer (June 2005) published by SIAM, with Bo Einarsson as editor. "Accuracy and Reliability in Scientific Computing."

States to a large extent and I think IFIP activities have just been of more interest and probably more value to people in other countries than in the United States. And coming from Sweden, where Bo is, I guess IFIP seemed to be an important window on the world for him.

HAIGH: Do you think being constituted as part of IFIP made much of a difference to the way that the group worked?

LAWSON: Well, it gave it a context. I'll use that word again. If somebody had just set out and said, "Let's have these 13 people get together and form a club that's going to talk about mathematical software and try to promote good ideas in the field," and it was not associated with any other organization, it probably wouldn't have gone anywhere. You probably couldn't get your home institution to pay your travel to go to the meeting, and just all kinds of other things wouldn't bode well for the longevity of the group. So I think if you felt a group like that was valuable, you'd want to tie it with some umbrella organization, and IFIP I guess makes sense if you're thinking of it as an international group primarily.

HAIGH: What do you think were the most important contributions of the group to the development of the field of mathematical software?

LAWSON: Well, my view maybe just parochial and personal. Maybe it isn't a good measure. For me, it was a chance to meet people from other countries and talk with them about topics in this field and hear what was going on in other peoples' offices and institutions and talk about what I was doing. When there was something specific involving me, like the BLAS, it was a good chance to talk about that with a representative group from a number of different countries and get feedback as to whether this is worthwhile or not worthwhile and whether it should have some other features in it or not, and things like that.

I guess personally I was never very enthusiastic about any kind of big project that the group would undertake, and in fact most of the people in the group were not, and we've had some explicit discussions about that and usually I think decided that that wasn't really the role of the group because everybody in the group was busy with other things both in their own work and other organizations and didn't want to take on things that were going to commit hours of work every month.

HAIGH: So can you give some examples of big projects that were proposed?

LAWSON: Well most recent, which I have not followed closely, is the production of a book. I have not even followed it closely enough to really remember exactly what the theme of the book is, but I do get email on the subject from time to time and it seems to be coming together. I think Bo Einarsson seems to be leading it, and I guess Ron Boisvert and others that are active now are doing it. Before the group was ever formed I think it's mentioned in one of Bo's write-ups that there was some discussion of maybe the group should try to write a mathematical software library and I don't think it took anybody very long to figure out that would not be a good idea, so we didn't try to do anything like that. And even things like the BLAS, which had a certain amount of sanction from the group, never a commitment in some sense that the group was going to see that it got done. Well, there were thoughts about trying to author papers giving advice on how to develop high quality software, how to do testing and things like that, and I think most of those people just decided nobody wanted to put that much effort in, so they just didn't do them.

We did have working conferences on topics like that and we did put out proceedings from those conferences, so we did put some material out in written form to the world at large on the subjects. But we didn't try to just sit down and write a paper or write a book or something on these topics.

{Since the book mentioned above did come out this summer (June, 2005), I wish to acknowledge that the current active members of WG2.5 have shown a willingness and ability to undertake and complete a significant sizable project. This did not happen while I was an active member. This new book has thirteen chapters, each written by from one to three of a total of seventeen contributors. Congratulations to Bo, to the current chair, Ron Boisvert, and to all of the other contributors.}

HAIGH: I recall Brian Ford suggesting to me that one of the areas where he thought it might have had some success is in providing a base from which to argue to computer manufacturers that scientific computing was still an important area that they needed to pay more attention to in producing their machines. Do you have any recollection of that?

LAWSON: Well, I don't remember that precisely, but I remember that there were discussions from time to time about the group taking a position on some issue like that and writing a letter to some other organization or to some vendor or something. I guess sometimes the group decided to do something, but generally the group was pretty cautious about that. I guess I didn't feel that we had all that much clout, so we'd just be kind of wasting our time, and maybe that's true, maybe it's not. Some of the things we did not do. Somewhere there was an appeal for input. We did do some things on the IEEE standards for hardware and on the Fortran Language Committee—I think we did write some position papers or at least brief letters and send them to those people and say, "We're this group of people and we work in math software development and we would like to see these features and we would not like to see those features."

HAIGH: Do you think that had any impact in either of those two areas?

LAWSON: Well I think in those two areas it might have, because particularly where they were looking for comment, I mean we weren't coming and crashing down their door and saying, "We want to tell you this." They were at a point where they were saying, "Okay, here we are if people want to tell us what they think of what we have here before we finish it up." So we were doing it at a time where they were looking for some input and we did have some credentials to be talking about these subjects, so I think it probably did have some impact with like the Fortran Language Committee or with the IEEE Committee. But to write to a vendor and say in building their machine they should think of such and such for the benefit of the math software people, I doubt that we would have much clout. I don't know if we ever did try to do that. If Brian remembers it, then it must have happened, but I don't remember exactly.

HAIGH: Had you met Brian Ford prior to the 1975 initial meeting?

LAWSON: That's interesting, because I've mentioned John Rice and Gene Golub that I feel really had major impacts on bringing me into the professional community. Brian Ford I would have to name also, I feel he's done things for me over the years that have helped me get into the community. And of course he was working along with Bo Einarsson and two or three others in the formation of this group, and I would have to think that he must have put my name forward as

one of the charter members because I didn't know Bo Einarsson at all. I think I probably had met Brian Ford before that. I should probably maybe interject into here out of the order we're talking about, the 1970 NATO grant that I had. So in the summer of 1970 I visited for one month at the AERE laboratory at Harwell and at the National Physical Laboratory in Teddington, so one month at each place, and that was really a major event for me to have that kind of time and to meet the people I met at those two places. And I most likely would have met Brian Ford during that time, although I don't specifically remember that, but I probably did. So I'm sure I must have met him before he was involved with selecting me to be on that first group of members of IFIP Working Group 2.5.

HAIGH: How would you describe Brian Ford's contributions in the field of mathematical software?

LAWSON: Well, it has to be right up there with the absolute major contributors. I was mentioning that I admired John Rice's ability to kind of see into the future and try to start things that would have some value downstream, and Brian Ford, I think is just superb at looking ahead and seeing how things are apt to unfold and seeing where some work could be done that would help things move along in that direction. And I think he had much more foresight in what the importance of math libraries could be, and also in what it would take to produce high quality math libraries than almost anybody else around 1970. Over the years he was more adventurous in having NAG try other things that were maybe a little bit off of what they needed to do, get into trying to do an ADA library, for instance, at quite an early stage. Economically that may not have turned out to be the best business decision, but I think from an intellectual point of view and a professional point of view it was really a good thing to do. It got people thinking about different ways to do things. I don't know, of course, the ins and outs of his management of the organization, but the proof was in the pudding I think. The organization has grown from just a volunteer group from a group of universities in England to apparently a very soundly established enterprise, and he hasn't left a lot of beaten and dead bodies along the way. He's contributed to the careers of many people both within his organization and outside by the things he's done, and he's reached out onto the Continent and done cooperative things there and in the United States. So I think he's just done a lot of very exemplary things.

HAIGH: How well do you think the group succeeded in this aim of, I think what is the key aim of IFIP as a whole, of spreading international understanding and harmony, but I think particularly in the context of the Cold War, this idea that science could bring people of different countries together and spread understanding?

LAWSON: Well, the group members were always cordial with each other, and I think we really pushed to meet in what would be regarded as maybe unusual or maybe not the most convenient locations. We didn't meet in Chicago every other year and in London on the other years or something. We were at Novosibirsk and Beijing. I did not personally go to Beijing, but I did go to Novosibirsk. The group went to Beijing. They met in Greece, and they've tried to encourage membership from countries other than just the most well developed countries as much as they could. So I think the group has really tried to have as wide of a group of nationalities represented as possible. I think one of our rules that we decided early on was not to have more than half of the members be from the United States because it would have been easy to fill up the whole group with people from the United States. So the group is operated that way. How much it has affected world peace or international understanding more broadly it would be hard to say

because it's a small group in a kind of specialized area, but I don't think it has hurt anything. I think it's one of probably many hundreds of thousands of organizations that must have to exist to get people from different countries talking to each other and working together and supporting.

HAIGH: I understand that as the group evolved, it had quite a pronounced social aspect, including the participation of family members.

LAWSON: Yes. The spouses came to meetings probably beginning with more or less the second meeting I think. The members were mostly men so the spouses were the ladies, and the ladies got acquainted with each other and enjoyed meeting with each other annually if they could get to that many meetings or whenever they could get to them and got acquainted with each other, and that was another aspect of international understanding. They enjoyed being in a group that spoke a lot of languages and would be in different countries and the different meetings, so that was a nice aspect of it.

HAIGH: Is there anything else you want to say about the development of the mathematical software community or ways in which it might have changed over time?

LAWSON: I think that pretty well covers it.

HAIGH: Actually, I do have a question that I've been asking people. It would be the extent to which you think that moving into the '80s and '90s that developments in mathematical software have continued to be tied to developments in numerical analysis as opposed to just reimplementing the same kinds of methods to work better on new computer architectures?

LAWSON: Well are you asking kind of whether there have been new algorithms or whether new algorithms have been implemented?

HAIGH: Well I've had different answers from it. For example, Cleve Moler was involved with the EISPACK project, which was taking some important new developments in terms of methods and disseminating them really quite rapidly in terms of high quality software. His perspective, as he's been involved with MATLAB over the last few decades, has been that kind of direct connection between more profound developments in mathematical methods and widely used software has broken down to some extent in that new research in numerical analysis has not been finding its way into wide dissemination as rapidly. On the other hand, Jack Dongarra had a different answer. So I was just wondering if you have any personal perspective on whether the connection between developments in numerical analysis and improvements in mathematical software remained strong into the '90s, or whether the two areas were not as connected as they used to be?

LAWSON: Well, both fields of course are much larger now, both the research field and the software field, so it's harder to make an impact because it's a bigger pond now than it used to be. For instance, if I pick up the latest issue of the *SIAM Journal on Scientific Computing*, which may have 15 or 20 articles in it, many of them are probably talking about a new algorithmic idea for solving some special kind of partial differential equation or improving the acceleration of an iterative method for solving linear equations or doing something with non-linear equations. One could ask, "Well is somebody putting that in the software? Is it going to be something I can buy or get within a year?" and the answer's probably no. I probably can't. I think everything is more

complex now, and the improvements that are made are more sophisticated and it's hard to put them into a software package that you think a lot of people are going to want to use.

If you go back to the example of EISPACK where Reinsch and Wilkinson had written a book that incorporated Algol code for the QR algorithm [Wilkinson, J. H. and C. Reinsch (1971). Handbook for Automatic Computation, volume 2: Linear Algebra, Springer-Verlag] and other things that had really just been developed in those last few years before that. Eigenvalue computation is something that's very widely used and these were relatively small algorithms in terms of how many lines of code they take, and they're relatively simple in terms of what the input and output is. The input is a matrix and the output is the eigenvalues maybe with the eigenvectors. There's no big data mining problem associated with them. And nowadays somebody's publishing an article on how they've analyzed DNA or something like that and they've gone into some big database and done something, so the complexity, the interaction with the data is just all more complicated. For a new research result to somehow show up in software that's easy to describe and easy to distribute and easy for somebody to use is just a wildly more complicated problem.

Now the one place where things probably are a little more focused, and since you gave me the hint of Dongarra, would be in the super computers. There, there's a kind of a focus. There's in some sense one central problem. It's not really that easy, but at least the general idea is: how can I break down an algorithm so a number of different processors can work on it and you don't get killed by the communication time between the processors? So that's a problem that can be stated in a few words, and a lot of people work on it and they may be able to come up with some kind of solution that's good for a lot of people, particularly systems that were developed 10 or 15 years ago that I guess are still used as kind of a basic intermediate system level for communicating between the different processors. And that I guess was getting fairly well standardized in the 1990s and I suppose still is. So there are some problems there that can be identified and people can focus on them and you can expect something to come out that you might be able to package and a lot of people can use.

But I think a lot of it is just so complex. But those partial differential equations in itself I don't think have ever been really well represented in libraries. Now our library, we never had anything in MATH77. I suppose NAG and IMSL do have something there, I don't know. I know there was a fellow who used to work for IMSL who produced a software package that solved a class of PDE's. He eventually left IMSL and went off and kind of made his own company to sell that one product because I guess they couldn't find a home for it in the library.[8]

And I think that's the problem. I mean you get into something like structural engineering, there's the NASTRAN package that was first of all developed within NASA is cooperative between some structural engineers at two or three different NASA centers and was submitted to COSMIC, which was one of their prized packages. But then a company was formed to maintain it, and now if anybody wants it they would certainly get it from that company; they wouldn't get the old free copy. And if somebody comes up with a new idea on structural analysis, you're not going to pick that up for free somewhere or you're not going to pick it up in the general library

[8] Lawson adds: Granville Sewell developed a software package called PDE2D that is described at: http://members.aol.com/pde2d/

like IMSL or NAG. You're going to have to go out to this company that sells structural engineering software and get it from them. So I think it's just kind of a different game right now.

HAIGH: So I wonder if you could identify one development in your professional career where you regret something or wish that an idea that you had had turned out differently or that you had followed up on something that you didn't. Do you know what that would be?

LAWSON: I don't know. I don't know. I mean a lot of things that I might have started to do didn't work out, that's for sure, but I don't have a list of them in my mind. That's probably fortunate [chuckles]. And I don't think I have any particular thought on that. I just think things kind of have their beginnings and they have their growth periods and then they kind of mature and they may kind of drop off the screen then because they don't need more work or something, and a lot of things I've worked on have gone that way. They've had their periods of excitement and then there wasn't anything more exciting to do on it, but I guess I don't think of anything that was particularly a negative thing that I remember.

HAIGH: Then for my final question, to reverse that, is there any one particular thing that you think stands out in your career in terms of what you might be most proud of or satisfied with?

LAWSON: Well, I don't know. Of course the least squares book is the thing that has made a number of people be aware of me, and the fact that it is still being reprinted this summer by SIAM is certainly a source of satisfaction. Otherwise, just kind of looking over all the time I was at JPL, I have lots of little memories of engineers and scientists who were appreciative that I was able to help them or the group was able to help them on something, and I think it fulfilled the intention I had to bring some mathematical and computationally oriented knowledge that would be a little more specialized than the general engineer or scientist might have to bear in a way that they could make use of it effectively, so I feel good about that. And of course the opportunity to meet a lot of very interesting people, very bright and intelligent people around the country and around the world that work in this field. So I wouldn't have had any idea people could be that bright and productive if I hadn't met people like Rice and Golub and Brian Ford. That's about it.

HAIGH: Thank you very much for taking part in this interview.

LAWSON: Well, thank you for thinking of me for the interview.

HAIGH: This is the end of Tape 4, Side B, and the end of the interview.

Appendix: Lawson's Autobiographical Notes

**"Notes on the Career of Charles L. Lawson
And the Technology Environment of the Times"**

By Himself, 2004, Sep 27; Oct 14

Personal History and Education

I was born October 19, 1931 in Weiser, Idaho. Weiser is at the confluence of the Snake and Weiser rivers where the Snake river forms the boundary between Idaho and Oregon. The population of Weiser has stayed around 5000 from 1900 to the present. My mother, Clara Vera Vial Lawson was an optometrist. She had her office in our home, which was only three blocks from the main business intersection of the town. She was also born in Weiser. In her teen years she worked in her dad's store which handled jewelry, clocks and watches, sheet music, recorded music, musical instruments and fitting of eyeglasses. She was ahead of her time in many ways. She attended the College of Optometry in Los Angeles, CA. She was one of only two female optometrists in Idaho throughout most of her career. She was a member of the State Board of Optometry for a number of years. My father, Charles Lewis Lawson, grew up on his parents' farm in Wilder, ID, a few miles from Caldwell, ID. He attended the Univ. of Idaho at Moscow, graduating with a bachelor's degree in business administration. He then came to Weiser to teach math and business subjects in the high school. He subsequently left teaching and worked as an accountant and buyer for a wholesale grocery company that had an office in Weiser. He was eventually involved in bringing early accounting and computing machines into the business. He once told me of an early computer he used that had a storage capacity of eight numbers.

I went through schools in Weiser through my junior year of HS. I worked in my mother's office – this included general office work and assembling eye glasses. For my senior year I lived with my aunt in Hollywood, CA, and attended and graduated from Hollywood HS. I attended the University of California in Berkeley from the fall of 1948 to June, 1953, earning a BS in Optometry in June, 1952 and a Certificate of Completion in Optometry in June, 1953.

I realized I had a strong interest in math during my freshman year of calculus at Berkeley, but I didn't have any notion of how one could have a career using mathematics. I think computers were just beginning to be mentioned in the general press around 1949. I remember reading a book about computers (Edmund C. Berkeley, *Giant Brains, or Machines That Think*, J. Wiley & Sons, 1949, http://www.psc.edu/~burkardt/misc/giant_brains.html. See http://www.blinkenlights.com/classiccmp/berkeley/ for a photo and a compact timeline of E. C. Berkeley's activities. Born 1909, died 1988, was a founder of ACM in 1947 and 1948.) and beginning to think there might be a future for me in doing mathematical work on computers. Nevertheless I continued with my optometry path at that time.

I took and passed state board exams in Optometry in California, Oregon and Idaho during the summer of 1953. In December, 1953, I was drafted into the army. After one year as a private, I was granted a direct commission as a second lieutenant to serve as an optometrist and served

another two years in that capacity. My military service was all in California, Texas and South Carolina. During the last two years I started preparing for a change of career from optometry to mathematics. I took two or three upper division math courses by correspondence from UCLA while stationed in South Carolina. In December, 1956, I completed my military service.

I began studies in math at UCLA in January, 1957, taking four senior level math classes, including one in computing, making use of the SWAC computer. (SWAC was short for NBS WAC which stood for National Bureau of Standards Western Automatic Computer. Prior to the SWAC there was SEAC – the NBS Eastern Automatic Computer, at NBS in Wn DC. SEAC used mercury delay lines for its memory. SWAC was dedicated in August, 1950, and at the time was the fastest computer in existence. It was retired in 1967. These details are from the *Encyclopedia of Computer Science*, Third Edition, 1993, Van Nostrand Reinhold). We programmed the SWAC in absolute binary, generally thought of in groups of 4 bits using hexadecimal notation. There was no operating system, no assembly language and no hardware floating point arithmetic. The working memory of the SWAC contained 256 36-bit words stored on the faces of 36 small (about 4 inch) CRT's. (These were called Williams tubes.) The secondary storage was a drum containing 8192 36-bit words. Transfers between memory and the drum were in blocks of 32 36-bit words, facilitating fast transfers. Low volume input was done using a 4-button keypad – one button for each bit of a hex number. Low volume output could be had on a flexowriter – basically an electric typewriter. Higher volume input was via a card reader. Higher volume output was via a card punch, and one could then take the cards to a printer to obtain printed output.

I began graduate math work at UCLA in the fall of 1957, completing an MS in math under Peter Henrici in June, 1959 and a Ph. D. in math in June, 1961. My dissertation was in approximation theory under Prof. Theodore Samuel Motzkin.
(http://www-gap.dcs.st-and.ac.uk/~history/Mathematicians/Motzkin.html)
The dissertation had two distinct parts: (1) Characterization of minmax fits over a 2-dimensional domain. (2) Minmax approximation via weighted least squares. (Prof. Motzkin was born March 26, 1908 in Berlin and passed away in Los Angeles on December 15, 1970.)

During my graduate years at UCLA I had an assistantship for one of the years under Thomas H. Southard. He was teaching a year course in numerical analysis, primarily using the 1956 textbook, "Introduction to Numerical Analysis", by F. B. Hildebrand. I attended his lectures, marked papers, and assisted students in the "computing laboratory", which was a room containing desks and electromechanical desk calculators – Friedens and Marchants. I was learning the subject matter along the way.

Later I had a teaching assistant position, teaching a lower division class in the Mathematics of Finance. Still later, I had a research assistantship with the Institute for Numerical Analysis (INA) at UCLA, which was directed at that time by Charles B. Tompkins. The INA had been set up at UCLA by the NBS in 1947 (and NBS was required to terminate support in 1954) to build and operate the SWAC computer and conduct research in numerical analysis. Both Southard and Tompkins were involved with the creation of the celebrated tome, *Handbook of Mathematical Functions*, Ed. Milton Abramowitz and Irene A. Stegun, NBS AMS-55, 1964. Tompkins was a member of the nine-person committee that directed the project, and Southard authored Chapter 18, Weierstrass Elliptic and Related Functions, pp 627-685.

The summer of 1957, I worked in a special summer program in the computer programming department of Douglas Aircraft in Santa Monica. They had a remarkably well organized training program, run by Russ Mapes, which stepped through the basic numerical methods known at the time. We would work examples on a Frieden or Marchant electromechanical calculator at our desk and also on an IBM 701 computer, programming in assembly language. I think I worked the summer of 1958 in the same department, this time becoming involved in maintaining and writing some application programs – still on the IBM 701. I believe an IBM 704 was being installed there toward the end of that summer and we were treated to a demonstration of it. The 704 was the computer for which IBM first provided a Fortran compiler.

On returning to UCLA after using assembly language at Douglas I was pained by the need to use absolute binary coding on the SWAC. I wrote a very primitive assembler for the SWAC that allowed symbolic names for memory addresses. I found this very helpful in subsequent programming of math algorithms on the SWAC.

In the summer of 1959 I worked for Robert Brown at IBM's Western Data Processing Center on the UCLA campus. This center was mainly involved in business data processing but I was doing Fortran coding of mathematical algorithms under Robert Brown's direction. Brown was a full-time employee of IBM, but also a Ph.D. candidate at the time -- I think under Peter Henrici. (What was I doing in the summer of 1960?)

Working at JPL, 1960 – 1996

I started working at JPL (Jet Propulsion Laboratory, Pasadena, CA. Operated by the California Institute of Technology {CalTech} for NASA.) in August, 1960 while I was still working on my dissertation. I entered JPL in the Applied Mathematics Group within a Section that was responsible for computer programming and computer operation. During my 36 years at JPL I believe this group was the only organization -- group, section or division -- at the lab that had the word *mathematics* in its name. Also I don't believe there were any individual job classifications with the word *mathematics.* I think all members of the technical staff were classified either as engineers or scientists. Over the years I generally referred to this group more simply as the Math group since I tend to associate *Applied Math* with partial differential equations and mathematical physics and this would not have characterized most of our work. Most of our thrust would have been better described as Numerical Mathematics or Computational Mathematics.

In those days engineers and scientists did not generally do their own programming or have direct access to computers. They had to describe their needs to programmers in this section and then the programmers would develop the programs and possibly even make production runs of the programs. The section manager was Bill Hoover. He had a sharp mind and regularly walked around his domain chatting individually with each member of his section. I had the feeling that he maintained a very clear knowledge of what everyone in his section was doing. He left JPL in 1964 to become a senior executive at CSC (Computer Science Corp.) and eventually the Chairman and CEO. CSC, founded in1959, was one of the first companies to provide a range of outsourced computer services to government organizations and large businesses, including writing systems software for computer manufacturers. See:
http://www.csc.com/aboutus/history.shtml

My group supervisor was Paul R. (Bob) Peabody. I think he had a Ph.D. in Math from the Univ. of Illinois (could be wrong on which university). He was very bright in general, but particularly when it came to devising numerical methods for scientific and engineering applications at JPL. I don't know of his ever publishing anything, but if he had been so inclined I think he would have made a mark in the community of numerical analysis. There were about five of us in this group. We were available to provide consultation to programmers in the section or to engineers and scientists throughout the lab on issues of computational methods. We also had a fair amount of latitude to pursue our individually selected interests in numerical methods.

During one of first summers I was at JPL, we had Cleve Moler with us as a summer hire in the group. This was either just before or just after his first year of graduate work at Stanford. It was indeed a pleasure and stimulating to have Cleve with us.

When Bill Hoover left JPL he recruited Bob Peabody to go with him as an assistant on technical issues. I was appointed to the group supervisor position that Bob Peabody vacated. This was by Jan, 1965 or a little sooner. I continued as supervisor of the Math group until the group was dissolved around 1990. The group had about 7 or 8 members from about 1966 to 1980.

Richard J. Hanson was in the group from about 1966 to 1971. He was with the Navigation group for another year and then left JPL to take a faculty position at Washington State University, Pullman, WA. During the time Richard was in our group, he and I collaborated closely on algorithms and software for least squares and coauthored the Least Squares book. After leaving the group and the lab, he continued to collaborate with Fred Krogh and me on the BLAS.

Fred T. Krogh joined JPL and our group in 1968. William Van Snyder joined JPL in 1967 and our group in 1974. Both Fred and Van were in the group until the demise of the group around 1990. Fred specialized in algorithms for the numerical solution of ordinary differential equations and wrote a number of papers and attended a number of conferences in that specialty. Fred also developed sophisticated algorithms and software in other topics, including nonlinear least squares, constrained optimization and multivariate polynomial interpolation in ragged tables. Fred and Van were both very interested in the concept of software processors that could aid in the production and maintenance of portable numerical software. They particularly worked on projects of that type in the late 1970's and the 1980's. We all worked on developing a well documented high quality library of mathematical subprograms for general use at JPL.

Others who were in the group at different times for periods of the order of 2 to 10 years were David Adorno, Elizabeth Baxter, Neil Block, Doug Campbell, Stella Y. Chiu, Charles James Devine, John Flynn, Ed Keberle, Prentice Knowlton, Lee Laxdal, Jay Lieske, Darrell Mulholland, Edward W. Ng, Douglas O'Handley, John Radbill, Albert (Bert) J. Semtner, Shirley Singletary Gold, Thang Trinh and Haralambo (Bob) Tsitsivas. Visitors, who were with us for about one to twelve weeks at different times included Brian Marsden, Cleve Moler, David Pierce, Jr., Kris Stewart, Henry C. Thacher and Carol Williams. Drs. Lieske, Mulholland, O'Handley, Marsden, Pierce and Williams were specialists in celestial mechanics and made significant contributions to the JPL ephemeris development work in the 1960's. Drs. Ng and Radbill were respectively an astrophysicist and an engineer and executed research and applications that justified the name *Applied Mathematics* for the group. Albert Semtner

subsequently earned a Ph.D. at Princeton in 1973 and had a distinguished career at NCAR and the U. S. Naval Postgraduate School in oceanography and global climate studies. (See his vita at http://research.nps.navy.mil/cgi-bin/vita.cgi?p=display_vita&id=1023568044).

In general, the group provided consultation on general questions of computational mathematics to requesting engineers and scientists at JPL. We had a particularly close and long lasting relationship with engineers who worked on the planning of the trajectory of spacecraft missions and on the "navigation" of a spacecraft during a mission. Prior to my arrival in 1960 and continuing through about 1967 the group developed software for producing high accuracy ephemeredes of the moon and planets. In celestial mechanics the term *ephemeris* is used for any table of the position of a celestial object as a function of time. The first operational use of these ephemeredes was in support of JPL's radar ranging of the moon and Venus from the earth. Later they were used by engineers in planning and executing spacecraft missions. The least squares work that Richard and I did was motivated particularly by the needs of the engineers who needed to determine the maneuvers of a spacecraft to reach a desired destination in space.

The ephemeredes we were computing were also used by other government laboratories and contractors. Plans were completed in Nov, 1964, to publish the JPL Lunar Ephemeris in the Astronomical Papers of the American Ephemeris.

During my involvement in ephemeris development, I had the opportunity to meet a number of specialists in celestial mechanics. This was a very interesting time for the field of celestial mechanics. There was a Center for Celestial Mechanics at Yale University and there were some specialists of note at the University of Cincinnati and at UCLA, but, in general, the field had not been attracting new blood for many years as there seemed not to be many significant new problems to be researched. Each year the Nautical Almanac Office at the Naval Observatory in Wn DC computed and published the Nautical Almanac which tabulated ephemeredes of the celestial objects that navigators would use to chart their courses in ships and airplanes. The tables were accurate to about 6 decimal digits and were computed from formulas that had been derived in about 1900. (I offer equal time to specialists to correct me on these details.) The wonderful thing about trajectories in outer space is that, with no friction, Newton's laws of motion can predict trajectories that are very accurate for many decades.

The space age breathed new purpose and vigor into the field of Celestial Mechanics. There was a need to know the positions of the moon and planets to about 9 decimal digits in order to communicate with them and navigate spacecraft among them and to them. There were also new and challenging issues in planning optimal trajectories for spacecraft. I enjoyed the experience of becoming acquainted with specialists at Yale and the Nautical Almanac Office (NAO) and having some of them visit JPL and our group for short or extended periods. It was interesting to navigate the cultural differences between the Yale and NAO people who had a tradition of not changing the "official" formulas used to tabulate ephemeredes more than once every 50 years, versus the JPL willingness to change the ephemeredes every week if we had some new observational data to bring into the computation.

CM-471, JPL Ephemeris Development 1960-1967, Lawson, Feb 23, 1981. (This was written much earlier but apparently was put in the files in 1981.)

(The "CM" can be taken to mean Computing Memorandum. This was a series of informal write-ups mainly by members of the Applied Math Group but including some by others in the computation section. As of 2004 these can be retrieved from the JPL Archive.)

JPL introduced a classification of *Senior Scientist* in about 1976. Through the energetic representation of my section manager at the time, Michael Warner, I was among the first half-dozen at the lab to receive this classification.

I joined AMS, MAA, ACM and SIAM about 1960. I discontinued membership in AMS after about 10 years. I continue the other memberships to the present. I was editor of the Scientific Applications Department of the Communications of ACM from 1969 to 1973, and editor of the ACM-SIGNUM Newsletter from 1972 to 1976. I was an active member of the IFIP WG2.5 on Numerical Software from its inception in 1975 until my retirement from JPL and from WG2.5 in 1996.

I retired from JPL in 1996. Fred retired a couple of years later and continues to do consulting for the lab. After leaving JPL in 1972, Richard was at Washington State University for a few years, then at Sandia Laboratory, Albuquerque for a few years and then had a long association with IMSL, Houston, TX, continuing to the present. Richard became a member of IFIP WG2.5. Van is still working at the lab, now with a group doing oceanic research. Van has become a member of the ANS Fortran standards committee and of IFIP WG2.5 and is an associate editor of the ACM Transactions on Mathematical Software.

At various times, NASA awarded our group, or individuals in our group, certificates, sometimes with $100 to $300 cash, in recognition of producing various software packages. The most notable award I received at JPL was the *NASA Exceptional Service Medal* for "exceptional contributions to applied mathematics at the Jet Propulsion Laboratory and for the solution of mathematical problems for space missions", signed May 9, 1991 by Richard H. Truly, Administrator, NASA.

Teaching and Organizing Seminars

Sometime in the 1961-1966 time period I taught an evening class in Computational Methods of Approximation Theory for UCLA Extension. The class was held in Pasadena and the enrollees included a number of JPL engineers and scientists. I later taught a class for UCLA Extension in numerical analysis in-house at a local engineering company. Both of these teaching experiences were satisfying, but very time consuming. The students were interested and I felt I gave them useful information.

A number of JPL'ers, including my colleague, W. Van Snyder, taught regularly for a number of years at West Coast University, which was an engineering-oriented evening school in Los Angeles. I taught one introductory numerical analysis course there, but was very disappointed in the uneven level of qualification of the students.

At JPL, when our MATH77 library was ready for use about 1985, our group put on a series of lectures about the contents of the library. This was a mini-course in numerical analysis, as we talked about each problem type and solution methods in some generality, besides talking

specifically about the use of the subprograms in the MATH77 library. This series was well attended and well received.

Later I picked up on the idea of a *Math Day* (or was it *Math Week?)*. The AMS, MAA and SIAM jointly promoted the idea of a day (or week) once a year, primarily at educational institutions, during which Math departments would schedule special speakers and other activities to increase awareness of mathematics throughout the institution. I organized such an event at JPL. I contacted persons I knew in various engineering and science sections of the lab whom I knew to be doing work of some mathematical sophistication and possibly general interest. I also brought in Solomon W. Golomb from USC as a featured speaker. (http://commsci.usc.edu/faculty/golomb.html). He had done important work on the design of the deep space communication facility at JPL from1956 to 1963, and has been on the faculty at USC since 1963. He has received many honors for his professional work, and is known in recreational mathematics circles for his invention of *polyominoes.* This lecture series was very well received with overflow audiences at some of the lectures.

From Approximation Theory to *Computer Approximations*

I will mention a couple of books that influenced me during my last years at UCLA or first years at JPL. One was *Approximations for Digital Computers* by Cecil Hastings, Jr., Princeton University Press, 1955, with 5th printing in 1966, 201 pp. Mr. Hastings was a consultant to Douglas Aircraft and possibly to the Rand Corp., both in Santa Monica, CA. It appears that his specialty was deriving fairly simple polynomial or rational expressions that could be used on computers to represent special mathematical functions or empirical curves. His methods were very primitive, but had some theory behind them and got the job done. This signaled to me, and I think also to John Rice and probably others, that there was an opportunity to do something more systematic in the area of curve fitting and the representation of functions on a computer.

Another interesting book was "On Numerical Approximation", Ed. Rudolph E. Langer, 1959, University of Wisconsin Press, 462 pp. This was the proceedings of the first symposium held by the Mathematics Research Center, UW, Madison, April 21-23, 1958. This was a few months before I had arranged to pursue a doctorate in approximation theory under Prof. Motzkin, so I was not aware of the symposium when it occurred. Prof. Motzkin, however, was one of the speakers, and his Ph.D. advisor, Alexander M. Ostrowski, of Basel, Switzerland was also. The twenty speakers and ten committee members and session chairs included a very impressive assemblage of the persons who were then prominent in approximation theory or numerical methods more generally. For example: Householder, Hestenes, Forsythe, David Young, Wasow, Philip Davis, Stiefel, Leslie Fox, Bauer, John Tukey, Collatz and John Todd. Thus the proceedings gave a good snapshot of the state of the art at that time. The introductory survey paper by Ostrowski is very interesting in showing both explicitly and implicitly the clash between pre-computer and post-computer views of approximation theory and computational methods. I think I took from this book the thought that there was a large body of theory that was known to specialists but was not being communicated to the potential user community of engineers and scientists effectively.

I gave a talk on least maximum approximation via weighted least squares (from my dissertation) at the 16th National ACM meeting in Los Angeles, September, 1961.

John Rice found my minmax approximation via weighted least squares to be of interest to him. He included an exposition of it (with my permission) as Section 13-11 in his book, *The Approximation of Functions, Vol. II – Advanced Topics*, Addison-Wesley, 1969. He had a Ph.D. student, Usow, who studied the method. The method has been mentioned in papers, and to some extent studied by some other authors, but does not seem to have found a lasting place in the field.

> {Added April 10, 2006. Other papers by authors who further studied my minmax algorithm.
>
> J.R. Rice & K.H. Usow, "The Lawson Algorithm and Extensions",Math. Comp., 22, 1968, pp. 118-127.
>
> Stephen Ellacott & J. Williams, "Linear Chebyshev Approximation in the Complex Plane Using Lawson's Algorithm", Math. Comp., 30, No. 113, 1976, pp. 35-44.
>
> Lloyd Nicholas Trefethen, "Chebyshev Approximation by Polynomials in the Complex Plane", A thesis in partial fulfillment of honors requirements for a B.A. at Harvard College, Cambridge, Mass., May 2, 1977, 87 pages.}

I gave a talk on "Segmented Rational Minmax Approximation, Characteristic Properties and Computational Methods" at the SIAM Invitational Conference on Approximations at Gatlinburg, TN, Oct 21-26, 1963. I met Cornelius Lanczos and other persons of national and international significance at this meeting. (I am unsure as to whether this conference had any connection with the "Gatlinburg" series organized by A. S. Householder.) This work was also published, I think in Numerishe Mathematik. It was documented at JPL as JPL Tech. Report 32-579, Dec 19, 1964.

I attended a "Symposium on the Approximation of Functions", General Motors Research Laboratories, Warren, Michigan, Aug 31 – Sep 2, 1964. Hard cover proceedings were published as *Approximation of Functions*, Ed. Henry L. Garabedian, Elsevier Publ. Co., 1965. The symposium was chaired by Garabedian. The proceedings contains thirteen papers, including ones by John Rice and by Carl R. DeBoor who were working at GM Research at that time.

I attended an ad-hoc meeting at NBS in Wn DC (~1964?) on the general subject of systematizing the computation of elementary functions on computers. (sine, arcsine, sinh, exp, etc.) Each new computer on the market required someone to write code for these functions and it seemed that this work was always being done from scratch, and in some cases the reliability, accuracy and efficiency of these codes left something to be desired. Some excellent work had already been done in this area, particularly by Hans Maehly, Hirondo Kuki, W. Kahan and Jim Cody (William J. Cody, Jr.) This meeting, instigated (I think) by John Rice, and chaired (I think) by Phillip Davis brought together about 20 persons who were known to John Rice to have interest, and possibly experience, that could be brought to bear on this issue.

The outcome was that a group of seven persons decided to work together and bring out a book that could be used as a guide by anyone needing to write code to compute elementary functions. The seven were J. F Hart, E. W. Cheney, C. L. Lawson, C. K. Mesztenyi, J. R. Rice, H. C.

Thacher, Jr. and C. Witzgall. It is worth noting that Thacher was also a member of the committee that directed the creation of the *Handbook of Mathematical Functions, AMS-55,* and Phillip Davis authored Chapter 6 and coauthored Chapter 25 of that book. (See the previous mention of this book in this write-up.) An eighth person, Hans J. Maehly, is listed along with these seven as a coauthor of the book because it was felt his work on this issue had already been very significant and he died at a young age just before work on the book began.

The book, *Computer Approximations*, was published by John Wiley in 1968. Jim Cody chose not to be a part of this book project, but he continued to do very high quality work in this area and later coauthored an excellent book with William Waite, *Software Manual for the Elementary Functions*, Publ. by Prentice Hall, 1980.

It is my impression that *Computer Approximations* was widely used for its intended purpose during the decade following its publication. This was in spite of the large number of misprints. Corrections to these were collected and published in a reasonably timely fashion in the periodical that was then called *Mathematical Tables and Aids to Computation*. (At some point the name of this periodical was changed to *Mathematics of Computation*.) The corrections were incorporated into a reprint of the book that was published by the Robert E. Krieger Publ. Co. in 1978.

After the initial meeting at NBS, the authors of the Computer Approximations book communicated by phone and postal mail, sometimes mailing reels of computer tape. Somehow we came up with a list of chapters and sections, and assigned the different chapters or sections to different authors. We had a general plan for the points to be covered for each of the elementary functions treated. I think there were assignments of persons to proofread the work of others; however, it appears from the number of misprints in the book that this part of the job was not handled well.

We planned to include lists of coefficients of polynomials and of rational functions that would give good (frequently minmax) approximations to elementary functions over specific domains and achieve various desired levels of accuracy. The book ended up having 150 pages of such coefficients plus another two pages of special values (*pi, log* base *e* of 2, etc.) listed to 35 decimal places and 40 octal places. We wanted all these numbers to be machine computed and machine transmitted to the printing process so there would not be any hand transcription errors. I had help in my office from Neil Block. He was essentially a programmer, but with a number of interests that ranged beyond his immediate tasks. He was interested in the computer control of typesetting, which was a very new field at that time. He was acquainted with someone in Minnesota working either in the use or the manufacture of a computer driven typesetting system, which I believe was called Photon. To make a long story short, I computed most of the coefficients listed in the book, making use of extended precision arithmetic subroutines (70-bit and 140-bit) I had partially written myself and partially adapted from code available in SHARE. Neil managed the transmission of the coefficients to his friend for computer controlled typesetting.

Solving Least Squares Problems

The book, *Solving Least Squares Problems*, Charles L. Lawson and Richard J. Hanson, Prentice-Hall, appeared in 1974. In 1986 a Russian translation was published. The translator was

Khakim Ikramov. The translator added references to Russian publications and an appendix discussing relations to Russian work. The Prentice-Hall book was republished by SIAM in 1995. In the 1995 edition the original authors wrote a new Appendix D giving a brief summary of work on the topics of the book from 1974 to 1995 and fifteen pages of references to works in that period. In 2004, we were happy to hear from SIAM that the book was being reprinted again to keep up with the continuing sales.

In the period of 1965-1969 Richard Hanson and I, at JPL, became acquainted with orthogonal methods (Givens and Householder orthogonal transformations) in least squares algorithms, and in the singular value decomposition and its application to least squares computation and analysis. This was primarily from publications of Gene Golub at Stanford Univ. (Golub and P. A. Businger, 1965, "Linear Least Squares Solutions by Householder Transformations", *Numer. Math*., 7, Handbook Series Linear Algebra, 269-276.) (Golub and Kahan, 1965, Calculating the Singular Values and Pseudoinverse of a Matrix", *SIAM J. Numer. Anal*., 2, No. 3, 205-224.) Richard and I began writing codes for these methods and introducing them to JPL engineers and programmers who were responsible for the design of spacecraft trajectories and the operational navigation of spacecraft. They found these methods more reliable than previous methods based on manipulation of systems of normal equations. Richard and I coauthored a paper about this work: "Extensions and Applications of the Householder Algorithm for Solving Linear Least Squares Problems", *Math. Comp*.(1969) 23, 787-812.

At some point, about 1970, Richard and I were invited by Gene Golub to give a talk at Stanford about this work George Forsythe, then Chair of the Computer Science Dept., was in the audience. After the talk George and Gene suggested to us that we should write a book on these topics. I believe George was an advisor or editor for Prentice-Hall at that time for their series in "Automatic Computation", and he paved the way for us to do the book with Prentice-Hall. (In the preface of the book we give thanks to the "late" George Forsythe. He passed away at the age of 55 on April 10, 1972. In 1961, Forsythe said,
"Enough is known already of the diverse applications of computing for us to recognize the birth of a coherent body of technique, which I call *computer science*. Whether computers are used for engineering design, medical data processing, composing music or other purposes, the structure of computing is much the same...." George founded the Stanford CS Dept. in 1965 as exclusively a graduate program. It was one of the first CS departments.
(http://www-db.stanford.edu/pub/voy/museum/ForsytheNews.html)

Richard and I worked up an outline of chapters and started writing separate chapters individually. We quickly found that this was not leading to a cohesive book. We then settled into an approach where we sat together and authored the book together, line by line. This worked. This process was significantly facilitated by our secretary at that time, Mrs. Roberta Cerami, who, as we noted in the book, "carefully and cheerfully" typed "the manuscript in all of its stages."

I gave an invited talk on least squares computations at a national SIAM mtg some time around 1974. (Date and place?)

I wrote an article on "Least Squares Approximation" for the *Encyclopedia of Computer Science,* Ed. Ralston & Reilly. The first edition of this encyclopedia was published by Van Nostrand

Reinhold in 1976. The third edition in 1993. There was another edition being compiled in 1996. As I was retiring from JPL in that year I suggested they look to Richard Hanson to pick up responsibility for the topic of Least Squares, and they did so.

Gerald J. Bierman was an engineer/mathematician in the Navigation group at JPL who worked on statistical estimation problems that are intrinsic to orbit determination and spacecraft navigation. He used computational methods generally called Kalman filtering in the engineering literature. These methods in their originally presented forms involve manipulations of the inverse matrix of a system of normal equations. It was generally understood by mathematicians working in numerical linear algebra, say by 1965, that one should not compute an inverse matrix as an intermediate entity in a computing process unless it was really needed as an end product, as it tends to be more expensive in computing time and sometimes leads to significantly less accuracy, or even complete failure, in the end product. Better alternative approaches use appropriate matrix factorizations. These ideas were appreciated by some engineers but had not been adequately communicated to the engineering community in the terminology to which they were accustomed. Gerald Bierman was one who came to enthusiastically embrace factorization formulations of Kalman filtering, after becoming acquainted with the sequential least squares software that Richard Hanson and I developed and that Richard particularly introduced into the Navigation group. During about the same period that Richard and I were writing our Least Squares book, Gerald wrote a book presenting factorization methods in the terminology of Kalman filtering: *Factorization Methods for Discrete Sequential* Estimation, Academic Press, 1977.

To solve least squares problems whose matrices were too large to be dealt with in the primary memory of a computer, John Ekelund and I designed and implemented methods that used two levels of storage. The secondary storage might have been a drum, tape or disc. (CM-435, “An Out-of-Core Least Squares Program”, Lawson and John Ekelund, Dec 6, 1977.) (CM-446, “Solving Large Dense Least Squares Problems Using Two Levels of Storage”, Lawson, May 7, 1979.)

I developed least squares software on a parallel computer for a JPL application in gravity research (late 1980’s to early 1990’s). The computer was the Cray T3D. I think it had 128 processing nodes. I developed a pipelined algorithm that solved a very large least squares problem by passing intermediate results from one node to the next. The code was used successfully by the gravity research team for about 1 or 2 years. Initially it gave them a capability to solve much larger problems than previously in a reasonable amount of time, which is what they wanted. Eventually the speed, memory size and storage capacity of individual work stations improved so much that it was more efficient for the scientists to do the whole computation on an individual workstation than to use a combination of their own workstations and the T3D.

BLAS -- Basic Linear Algebra Subprograms

Computers from different manufacturers appearing in the 1970-1972 time period had new and different designs of their CPU’s and arithmetic units. To achieve the intended efficiency of these designs, one would need to code some mathematical computations in custom designed assembly code. For example the CDC 7600 had an instruction cache of about 16 instructions. If a loop

was complete in this number of instructions it would execute much faster than would a longer instruction sequence. To avoid the extreme non-portability of coding sizable mathematical computations in assembly language, differently for different machines, the idea arose to code relatively small critical sequences in assembly language but using a standardized name and argument list for each of these basic operations, so it could be called from a portable Fortran program, and the relatively small assembly coded basic operation could be recoded in assembly language for optimal efficiency on different types of computers.

This general idea occurred to different people at different institutions around the 1970-1972 time period – particularly to anyone trying to develop packages or libraries of mathematical software with an eye to portability and efficiency.

In our group one of the first investigations of this idea was documented in CM- 303, "On the Use of Assembly Code for Heavily Used Modules in Linear Algebra", Fred T. Krogh, May 2, 1972.

Fred and Richard Hanson and I set out to define a set of subprogram names and argument specifications for the operations that typically constituted the inner loops of linear algebra programs. Our first draft of this project was

R. J. Hanson, F. T. Krogh, and C. L. Lawson, "A Proposal for Standard Linear Algebra Subprograms", Jet Propulsion Laboratory Technical Memorandum 33-660, November, 1973, vi+14 pp.

This project was discussed in the annual meetings of IFIP WG2.5. The ACM SIGNUM Newsletter was used as a vehicle to communicate this proposal widely and report its subsequent evolution. An open meeting was held to discuss and modify the proposal in May, 1974, at the Math Software II Conference at Purdue University, and another at the National Computer Conference in Anaheim, May, 1975. David Kincaid, University of Texas, became actively involved, giving particular attention to assembly coded versions for the CDC 6600. By 1977, Hanson was at Sandia, Albuquerque, and a Sandia report issued in 1977 was essentially the final version:

R. J. Hanson, C. L. Lawson, D. R. Kincaid, and F. T. Krogh, Basic Linear Algebra Subprograms for FORTRAN Usage – An extended report, Sandia Tech. Rep. SAND 77-0898, Sandia Lab., Albuquerque, NM, 1977.

The truly final version appeared in *ACM TOMS* in September, 1979:
C. L. Lawson, R. J. Hanson, D. R. Kincaid, and F. T. Krogh, Basic Linear Algebra Subprograms for Fortran Usage, *ACM TOMS*, v. 5, No. 3, September, 1979, pp. 308-323, and Algorithm 539, pp. 324-325.

The paper specified 38 subprograms and included instances of each written in Fortran, as well as assembly coded versions of some of the subprograms for some of the popular computer types of the time. These subprograms each contained at most one level of looping.

Two errors in the BLAS paper were noted in1982 by David S. Dodson and Roger G. Grimes of Boeing Computer Services Co. After consultation with Lawson these were submitted to TOMS and appeared as "Remark on Algorithm 539", *ACM TOMS*, 8 (4): 403-404 1982.

The LINPACK package of linear algebra subprograms was being produced in these same years by J. J. Dongarra, C. B. Moler, J. R. Bunch and G. W. Stewart, and they made the decision to use the BLAS in that package. This had the effect of eventually giving the BLAS very wide distribution and recognition. In fact Jack Dongarra made substantial contributions to the testing and coding of the BLAS before they were published, particularly adding loop-unrolling to the Fortran versions.

At JPL, in the mid-1980's, Alan Klumpp became very interested in using the DoD- sponsored programming language, Ada, in JPL projects. He asked me to help him in transcribing some LINPACK and BLAS codes from Fortran to Ada. This work was noted in CM-520, "Four LINPACK subprograms in Ada", Klumpp and Lawson, Oct 16, 1986.

As computer design evolved further it was observed that significantly greater efficiency could be achieved by having basic subroutines that contained two or three levels of looping. This led to the development of BLAS2 and BLAS3, implementing two and three levels of looping respectively. For clarity and consistency, the original BLAS package is now generally called BLAS1. BLAS2 and BLAS3 were published respectively as follows:

"An Extended Set of FORTRAN Basic Linear Algebra Subprograms", J. J. Dongarra, J. Du Croz, S. Hammarling and R. J. Hanson, *ACM TOMS*, v. 14, 1988. pp. 1-17 and Algorithm 656, pp. 18-32.

"A Set of Level 3 Basic Linear Algebra Subprograms" by J. J. Dongarra, J. Du Croz, I. S. Duff and S. Hammarling, *ACM TOMS*, v. 16, 1990, pp. 1-17 and Algorithm 679, pp. 18-28.

To make effective use of BLAS2 and BLAS3 it was necessary to make major changes to the usual algorithms for the solution of linear equations and eigenvalue algorithms. This required the derivation of *block* algorithms. A considerable effort went into this. The focus of this effort was the production of a new software package, LAPACK, to use blocked algorithms and encompass the functionality of LINPACK and EISPACK and a bit more. LAPACK is described in

LAPACK Users' Guide, E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov and D. Sorensen, SIAM, 1992. (Royalties from the sales of this book are placed in a fund to help students attend SIAM meetings and other SIAM related activities.)

In recognition of the 25th anniversary of the publication the article in the SIGNUM Newsletter, proposing the original BLAS project, James C. T. Pool, Sven Hammarling and Jack J. Dongarra organized a minisymposium at the SIAM Conference on Parallel Processors (PP99), San Antonio, TX, March 23, 1999, to describe the history and planned future developments for the BLAS family of packages. The speakers were Lawson, Hammarling, Iain Duff and Dongarra. I spoke on "Background, Motivation and a Retrospective View of the BLAS". The others spoke

on BLAS2, BLAS3, software capable of producing efficient BLAS codes customized to a new computer type, and BLAS-type packages for parallel processing computer systems.

Developing JPL Libraries of Math Software

When I joined JPL in 1960 the main computers were IBM 7094's. The IBM 7094 had a 36-bit word and hardware floating point arithmetic – both single and double precision. Fortran was very new, so most of the application software at JPL had previously been written in assembly language (SAP or FAP) and many programmers were reluctant to move from assembly language to Fortran.

Our group developed libraries of mathematical software for use at JPL -- first in a quite informal way, with increasing systematization as the years went on. We contributed a small number of subroutines to a "Fortran IV" math library on the IBM 7094. "Fortran IV" was IBM's version of Fortran. (Of course, Fortran was an IBM development to begin with.)

In the time frame of 1968-1969 JPL was converting from use of IBM 7094 computers to Univac 1108 computers for engineering and scientific computing. The 1108 also had a 36-bit word and both single and double precision hardware arithmetic, but the double precision had about 8 bits more precision than on the 7094 because the exponent field of the second word was not wasted. There are memos from that period spelling out our group's plans to produce a "Fortran V" math library on the 1108's. "Fortran V" was Univac's designation of their version of Fortran.

Following about 1972, less expensive computers appeared on the market and individual sections around the lab began buying their own computers. For example, VAX minicomputers first appeared on the market in 1968. (http://www.mitem.com/solutions/platforms/digitalVax.asp) Although we took cognizance of portability issues, it was very difficult to make a library portable in the 1970's. There was a 1966 ANSI Fortran standard, but it lacked important features and each computer manufacturer tended to provide these features in different ways in their unique Fortran compilers. Apparently the final version of our math library for use on the 1108's was Edition 5, described in JPL D-829, 1975.

With the details of the forthcoming new Fortran 77 standard mostly in place by 1975, we started planning and working toward producing a math library, that we called MATH77, which would be portable to all computers when they supported the Fortran 77 standard. It wasn't until about 1985 to 1988 that one could count on most computers in use at JPL having a reliable Fortran 77 compiler.

MATH77, Release 2.0, JPL D-1341, Rev. A, Oct, 1987, contained 43 write-ups describing 330 user-callable entries.

MATH77, Release 2.1, CM-526, June, 1988.

MATH77, Release 2.2, CM-536, December, 1988.

MATH77, Release 3.0, JPL D-1341, Rev. B, May, 1989, contained 51 write-ups describing 412 user-callable entries.

By 1988, an ANSI committee was essentially finished with the first effort to produce an ANSI standard for the C language, which was eventually issued as C90. In 1988 we started converting the MATH77 library to C90. (CM-532, Conversion of Fortran 77 to C, Lawson, Sep 27, 1988) We used a commercial conversion package that helped considerably in converting the Fortran 77 codes to C90; however a significant amount of manual editing was needed also. We gave the name *mathc90* to our C version of MATH77.

MATH77 and *mathc90,* Release 5.0, JPL D-1341, Rev. D, Jan 1995, contained 77 write-ups and over 450 user-callable names in Fortran, and about the same number in C.

MATH77 and *mathc90,* Release 6.0, JPL D-1341, Rev. E, May 1998, contained over 550 user-callable names in Fortran, and about the same number in C. This was the "final" version of these libraries. It was completed by Fred and Van after I retired from JPL.

In the course of trying to develop high quality portable libraries with high quality users' manuals we developed a fair amount of auxiliary software and specifications to support the effort. This included software for applying transformations to code, software to systemize testing, software to support printing users' manuals – to include mathematical notation and graphs, and specifications of tape or file formats to facilitate moving code between different computer systems.

We developed a number of software tools to aid in developing portable software for the libraries. This included:
Specializer (Krogh, CM-359, 1974);
MARVEL (Lawson, CM-463, 1980);
CHGTYP (Lawson, CM-525, Apr, 1988);
M77CON (Krogh, 1994-1996, included in MATH77, Release 6.0)

We also developed software packages to execute tests of numerical results:
(1) For basic math functions of one or two variables STFN and DTFN (Lawson, 1964, Snyder, 1996, included in MATH77, Release 6.0), and
(2) To compare results for one or more library codes after either modifying the code or moving it to a different computer system: CDEMO and CDEMOM (Krogh, 1995, included in MATH77, Release 6.0)

We tried various approaches to manual publication at different times from 1965 to 1998 depending on what printing and plotting hardware and software was available. At one point we specified a printing language, GPPS (General Purpose Photocomposition System), CM-351, Jan, 1974 by Prentiss Knowlton, JPL. In the later years we made use of Don Knuth's TEX system.

SFTRAN and Other Preprocessors

SFTRAN was a preprocessor for Structured Fortran. The idea of SFTRAN originated with John Flynn about 1972 or 1973, then a programmer at JPL and not at the time in my group. John was given management support to develop his idea. He designed the SFTRAN language as essentially Fortran (in its not very well standardized state of the early 1970's) plus some

additional selection and looping statements and an internal procedure statement, so one could write codes that were functionally equivalent to the Fortran code of the day that would not need to have any statement numbers or GO TO's. Among other benefits, this made it much easier to write, read and modify code without needing to keep track of Fortran statement numbers and references to them. John wrote a preprocessor in Fortran (and then rewritten in SFTRAN) that could convert a code written in SFTRAN to Fortran code which could then be passed through a Fortran compiler to produce executable code.

"SFTRAN User Guide (Structured Programming to FORTRAN Translator", John Flynn, July 31, 1973. Identified both as CM-337 and JPL-1846-7.

CM-424 "Notes on the Transfer of SFTRAN to a DEC-10 Computer", Lawson, Feb 23, 1977.

CM-437 "SFTRAN Language Constructs Supported by a Portable Preprocessor", Lawson, Jan 25, 1978.

JPL-1846-98, "SFTRAN3 Programer's Reference Manual", Dec 1, 1978.

The original SFTRAN preprocessor was not portable. By 1978 we had produced a portable version. We changed the name of the language and the preprocessor to SFTRAN3 at that time. In this current write-up we will (mostly) stay with the name SFTRAN for simplicity.

SFTRAN was welcomed and used by programmers in the spacecraft navigation group where John resided at the time. SFTRAN was also welcomed and used in my group. Eventually John was moved to my group to better continue his R&D type of work. I believe SFTRAN served as a useful tool in helping the application programmers who used it do a better job of designing and implementing code during approximately the period from 1973 to 1988. Beyond that time Fortran compilers conforming to the ANSI 1977 standard were widely available and most of the features of SFTRAN were available directly in Fortran 77. (Strict Fortran 77 did not have the DO-ENDDO or the EXIT of SFTRAN, but these features were being added into what was to become Fortran 90 and many Fortran 77 compilers implemented these features.)

During the early 1970's, preprocessors very similar in purpose and features to SFTRAN were developed at at least a half dozen other places in the US. Probably the most widely known was Ratfor, developed by Brian Kernighan (then at Bell Laboratories), and eventually generally distributed with Unix and Unix-like operating systems.

Lawson organized a national meeting in Pasadena, jointly sponsored by JPL and SIGNUM in 1974: "Workshop on Fortran Preprocessors for Numerical Software". We particularly encouraged members of the Fortran standards committee (X3J3) to attend this workshop, and were pleased that a number did. At that date members of X3J3 did not intend to entertain any further changes to the new standard and were going into public reviews and reviews by their parent organization. I think the strong interest in cleaner control structures, such as IF-THEN-ELSE-ENDIF, evidenced convincingly at this meeting, nudged members of X3J3 toward their eventual action of making at least this structure a part of Fortran 77.

X3J3 held two Fortran Forums in 1976, one in New York and one in California. This gave X3J3 more user feedback.

In October, 1978, JPL and SIGNUM sponsored another meeting in Pasadena: "Conference on the Programming Environment for the Development of Numerical Software". This meeting was coordinated with a meeting of X3J3 so that a half-day session was scheduled as a joint meeting of the SIGNUM attendees and X3J3 attendees. The proceedings were published in pp. 28-117 of the SIGNUM Newsletter, v. 4, n. 1, March, 1979.

A third JPL/SIGNUM meeting was held July 29-31, 1981, in Pasadena with Fred T. Krogh of JPL as the conference chair: "Conference on the Computing Environment for Mathematical Software". Proceedings of this meeting were published in a special issue of the SIGNUM Newsletter, Oct, 1981.

Surface Fitting and Contour Determination

Algorithm and code for contour plotting. CM-106, "FORTRAN IV Subprograms for Contour Plotting", Lawson, Block, Garrett, May 4, 1965.

I gave a talk on my algorithm and code for contour plotting at the University of Texas Computation Center, Dec 7, 1965. (JPL SPS 37-32, Vol. IV, pp. 18-22)

Algorithm and codes for building a triangular grid. For points in a plane. For points on the surface of a sphere. Simplicial grid for points in general N-dimensional space.

Lawson, Charles L. Transforming Triangulations. Discrete Math. 3 (1972), 365--372.

JPL application: Surface fitting. Methods using b-splines, incorporating conditions on 1st and 2nd derivatives. Different methods using triangular grids, aiming at either C^0 or C^1 continuity. Contact with Robert E. Barnhill. Mtg in Albany NY. (Date?) Met Pierre Bézier.

NASA Workshop on Surface Fitting at Texas A&M University, College Station, TX, May 17-19,1982. About 26 participants. Organized by L. F. Guseman, Jr. and Larry L. Schumaker, both of Texas A&M. NASA funded. NASA's interest in this topic at the time came specifically from the "AgRISTARS" Program at NASA, Houston, which aimed to study world-wide distribution of vegetation from analysis of images obtained from orbiting vehicles. From the proceedings: "The purpose of the workshop was to bring together leading experts from academia, industry and government laboratories for an exchange of views and a discussion of the 'state of the art'." I gave a talk on "C^1 surface interpolation for scattered data on a sphere".

Symposium on Surfaces, held in conjunction with a SIAM mtg at Stanford, July, 1982. Organized by R. E. Barnhill, Univ. of Utah, Salt Lake, and Gregory M. Nielson, Ariz. State Univ., Tempe, and encouraged by Gene Golub of Stanford. Seventeen papers were presented and these were all published as the Winter, 1984 issue, Vol. 14, of the *Rocky Mountain Journal of Mathematics*. Includes paper by Lawson: "C^1 surface interpolation for scattered data on a sphere", pp 177-202.

CM-501 "A Piecewise C^2 Basis for Function Representation over the Surface of a Sphere", Lawson, Aug 1, 1984.

Robert E. Barnhill of the University of Utah and an international group of colleagues founded a new journal to publish research in computer aided geometric design. The journal is called *Computer Aided Geometric Design* and is published by Elsevier (North Holland). I have an incomplete collection of issues from v. 1, n. 4, Dec, 1984 to v. 11, n. 6, Dec, 1994. The v. 1, n. 4 issue shows Barnhill and Wolfgang Boehm of F. R. Germany as co-Editors-in-Chief. This journal is still active.
(http://www.elsevier.com/wps/find/journaldescription.cws_home/505604/description)

"Properties of N-Dimensional Triangulations", Lawson, *Computer Aided Geometric Design*, v. 3, April 1986, pp. 231-246.

Automatic Differentiation

An early paper on what was later generally called *Automatic Differentiation* was "A Simple Automatic Derivative Evaluation Program" by R. E. Wengert, Comm. ACM, v. 1, Aug 1964, pp 463-464.

Starting from the ideas in this paper, I developed software to carry out these operations extending the ideas to derivatives of arbitrary positive integer order, and called it W-arithmetic -- the ""W" denoting Wengert. I extended the ideas to compute partial derivatives, of first and second order, of functions of *k* variables and called that U-arithmetic, because *u* is frequently used as a name for a function of more than one variable. I documented this in an internal write-up, CM 289, "Computing Derivatives using W-arithmetic and U-arithmetic", Sept 1971.

These procedures can be viewed as automating the application of the chain rule for differentiation. They can also be viewed as operations on scaled truncated Taylor series. When one encounters functions defined implicitly, it is useful to think in terms of series reversion. I looked into this problem and did a write-up, "Series Reversion as the Reversed Chain Rule", CM-523, Sep 15, 1987. I published this in the *SIGNUM Newsletter*, v. 23, n. 1, Jan, 1988.

Over the years others developed sophisticated theory, algorithms and software for automatic differentiation. Andreas Griewank, Argonne National Laboratory, and George F. Corliss, Marquette University, were particularly active in this area and organized the first conference devoted to this topic: the "SIAM Workshop on the Automatic Differentiation of Algorithms", Breckenridge, CO, Jan. 6-8, 1991. About 31 papers were presented. These were published by SIAM in 1992 as the book: *Automatic Differentiation of Algorithms*, Ed. Griewank and Corliss. I attended this workshop and presented a paper: "Automatic Differentiation of Inverse Functions", pp. 87-94.

SIGNUM

I have a nearly complete collection of all the issues of the *SIGNUM Newsletter* from 1966 to 1998. This Newsletter served as an important means for relatively rapid communication of announcements and very short technical notes such as are now handled (much more rapidly) by the "NA-Digest" which is edited by Cleve Moler and disseminated weekly via the internet. (Back copies of the NA-Digest can be viewed at www.netlib.org/na-digest-html. These go back to Feb, 1987.)

Vol. 1, No. 1 of the *SICNUM Newsletter* is dated June, 1966 and lists Joseph F. Traub as editor. In fact it appears that Joe was the only "officer" of SICNUM at the time, as no other officers are listed. (The "C" is for Committee. Later when ACM established a more formal structure for Special Interest Committees and Special Interest Groups, the "C" changed to "G" for Group.) Joe was apparently visiting at Stanford University at that time, as he gave his address as CS, Dept., Stanford before Aug 1, and Math. Research Center, Bell Telephone Labs., Murray Hill, NJ, after Aug 1. He reported that ACM SICNUM had almost 1000 members from over 20 countries at that point. (Membership was free, one only needed to send in one's name and address to Joe. I don't know how these first 1000 members, of which I must have been one, were recruited.)

Vol.1, No. 2, July, 1966, announced there would be a meeting of SICNUM on Aug 30, 1966, at the National ACM Conference at the Ambassador Hotel, Los Angeles, CA. John Rice would give an invited talk on "The Automated French Curve", and there would be a panel discussion on "Machine Implementation of Numerical Algorithms", chaired by W. J. Cody, with panel members R. Hamming, W. Kahan, H. Kuki and C. Lawson.

Vol. 2, No 2, June, 1967, reports there are local chapters of SICNUM in Boston, Los Angeles and Wn DC. The chair of the L.A. Chapter is listed as Charles Lawson.

Vol. 2, No. 3, December, 1967, reports that a Board of Directors and a set of officers and committee chairs have been established for SICNUM, with the Board having met for the first time, Aug 29, 1967, during the ACM National Conference in Washington, D.C. Joe Traub became the first Chair of SICNUM and Cleve Moler (then at the Univ. of Michigan) was to take over the editorship of the Newsletter as of January, 1968. The Board had 10 members: Conte, Forsythe, Gear, Givens, Householder, Hull, Ralston, Werner Rheinboldt, J. B. Rosen and David Young.

Vol. 3, No 3, Oct, 1968, has 3 items of Lawson news. Anthony Ralston resigned from the SICNUM Board and Lawson was appointed to the vacant seat. (Later, in 1972-1974, Ralston was the president of ACM.) Lawson was also named to plan the technical program for a SICNUM meeting at the 1969 National ACM Conference. Lawson had been one of the speakers at the SICNUM meeting at the 1968 National ACM Conference (August in Las Vegas) and this issue contained an eight-page writeup by C. Lawson of his talk, "Survey of Computer Methods for Fitting Curves to Discrete Data or Approximating Continuous Functions". The writeup mentions that I had also collected a "Bibliography of Recent Publications in Approximation Theory with Emphasis on Computer Applications" which was to appear in Computer Reviews,

v. 9, Nov, 1968. In collecting this bibliography I had written to about 120 persons and received 35 responses. The bibliography included 394 titles.

Vol. 4, No. 1, Jan, 1969, uses the name SIGNUM rather than SICNUM for the first time. It is announced that the ACM Council approved converting the Committee to a Group.

Vol. 6, No. 1, Jan, 1971, announces four persons in the UK, Sweden, Israel and Switzerland were appointed as international coordinators for SIGNUM for the purpose of working toward a more international scope for SIGNUM and its Newsletter. "It is hoped that this Newsletter can eventually serve as one definitive focal point for international communication in numerical analysis." It is noted that Fred Krogh of JPL is now the chair of the Los Angeles SIGNUM, replacing Lawson. It was announced that the SIAM council and the SIGNUM Board agreed to "hold a joint meeting on Numerical Mathematics as part of the SIAM 1972 fall meeting which will be held either in Austin or San Antonio. Program co-chairmen for the meeting are G. W. Stewart of the University of Texas and C. B. Moler of the University of Michigan. The meeting marks an important first, being the first jointly sponsored symposium of SIAM and SIGNUM."

Vol. 7, No. 1, April, 1972. The editorship passed from Moler to Lawson starting with this issue.

Vol. 11, No. 2, Aug, 1976. *Myron Ginsberg wrote:* "With this issue of the Newsletter the editorship has changed hands from Chuck Lawson of Jet Propulsion Laboratory to Myron Ginsberg of Southern Methodist University. During the past four years Chuck has diligently increased the scope and quality of the Newsletter while continuing to distinguish himself in the numerical linear algebra and math software areas with many high quality papers, active and effective participation in several conferences, and as a co-author of an excellent book on least squares. The entire membership of SIGNUM owes him a high degree of appreciation for his unselfish efforts in their behalf."

A Special Issue in Feb, 1979, consisted of a 48-page report by a committee of eight numerical analysts who worked with NSF funding to produce a report on "Numerical Computation – Its Nature and Research Directions". The participants were Rice, Gear, Ortega, Parlett, Schultz, Shampine, Wolfe and Traub.

Los Angeles SIGNUM

A Los Angeles SIGNUM was formed in 1964 to have monthly meetings in the Los Angeles area. The following details come from Robert Mercer, who has been a long time active member of LA ACM and LA SIGNUM and performed newsletter and meeting announcement functions for both organizations. The first meeting of LA SIGNUM was February 6, 1964, at Raffles Restaurant with Dr. Paul Peabody (JPL) speaking on "Numerical Integration of Orbits." Officers were elected as follows: Chairman, Rube Kuritsky (S&ID); Vice Chairman, Charles Lawson (JPL); 2nd Vice Chairman, Ralph Dames (JPL); Secretary, Jay Kleinbard (S&ID), and Treasurer, Florence Anderson (NAA/LAD). "Forty enthusiastic people attended...."

Lawson became chair in the fall of 1966, followed in subsequent years by George Kuby and Fred Krogh. After Alston Householder retired to Malibu, CA, from the Univ. of Tennessee in 1974, he

attended meetings of LA SIGNUM and was chair for a while. Attendance fell off in the 80's and 90's. Apparently the last meeting held was No. 212 in 1998.

1970 Visit to AERE and NPL

In 1970, Edgar Eichorn, in a management staff position at JPL, brought to my attention an opportunity to apply for a NATO grant to visit one or more technical institutions in Europe. The upshot was that I obtained the grant to spend a month at the Atomic Energy Research Establishment (AERE) in Harwell, England, and a month at the National Physical Laboratory (NPL) in Teddington, England (June and July, 1970). At AERE I was attached to a Math group headed by Alan Curtis that included John Reid, Roger Fletcher and Michael J. B. Powell. I worked on a paper on a theoretical aspect of a triangular gridding algorithm. (CM-259, "Transforming Triangulations", Oct 28, 1970) This paper was subsequently published as "Transforming Triangulations". *Discrete Math*. 3 (1972), 365--372.

At NPL I was attached to Geoff Hayes who also had Maurice Cox working under him. Geoff had worked out some interesting applications of curve and surface fitting with applications in the design and construction of the hulls of ships. I had frequent conversations there with Phillip Gill and Walter Murray. Phil and Walter were introducing orthogonal transformation methods into the linear algebra parts of optimization algorithms. I had a small amount of contact with Jim Wilkinson at NPL.

Mathematical Software Conferences Organized by John Rice.

It is possible that John R. Rice invented the term *mathematical software*, but if not, he certainly contributed effectively to giving it a meaning and bringing it into use.
(http://www.cs.purdue.edu/people/jrr)
(http://www.cs.purdue.edu/homes/jrr/vita/99312vitaShort.html)

A "Mathematical Software Symposium" was held at Purdue Univ., April 1-3, 1970. This was the brainchild of John Rice. He appointed an "organizing committee" of helpers consisting of Robert Ashenhurst, Charles Lawson, Stuart Lynn and Joseph Traub. (I think the main connecting thread among these folks was that they were all involved as editors of ACM or ACM-SIGNUM periodicals related to the concept of mathematical software.) ACM and ACM-SIGNUM sponsored (publicized) the symposium. John obtained a grant from ONR to fund the conference and arranged for publication of the papers as a hard cover book: *Mathematical Software*, Ed. John R. Rice, Academic Press, 1971. In the preface John says the symposium's "primary objective was to focus attention on this area and to provide a starting point and meeting place for a more systematic development. The editor believes that mathematical software will develop into an important subdiscipline of the mathematics-computer science area and that it will make a significant impact on both theoretical developments and practical applications."
The book contains about 35 papers, including one by Lawson: "Applications of Singular Value Analysis", pp. 347-356.

"Workshop on Mathematical Computations", Granby Colorado, August, 1972. Funded by NSF. Apparently organized by Rice (and others?). Led to formation of a working group consisting Rice, Tom Hull, Stuart Lynn, Joe Traub and Lloyd Fosdick to explore ways to improve channels

of communication regarding the development of mathematical software. Their work had the major results of organizing the Mathematical Software II conference and founding the publication: *ACM Transactions on Mathematical Software*.

"Mathematical Software II", Purdue, May 29-31, 1974. Organized by the persons named in the preceding paragraph plus Joel Moses, representing the field of symbolic algebraic manipulation. Sponsored by ACM-SIGNUM, ACM-SIGSAM (Symbolic and Algebraic Manipulation), SIAM and NSF. There were about 15 invited talks and 70 contributed papers. Four of the contributed papers were given by members of our group at JPL – by Lawson, Krogh, Hanson and Edward Ng. Lawson spoke on "Standardization of Fortran Callable Subprograms for Basic Linear Algebra". A number of papers from this conference were published in the first two issues of *ACM Trans. Math. Software* March and June, 1975. There was also a 324 page "informal proceedings" distributed at the conference

The first issue of the *ACM Transactions on Mathematical Software* appeared in March, 1975. John was the Editor in Chief. Lloyd Fosdick, who had been Algorithms Editor of the *Communications of ACM,* was the Algorithms Editor. The Algorithms department of *CACM* was discontinued, and all of its functions, which included refereeing and publication of both papers and software, as well as cataloging and distributing the software, were transferred to the new *ACM Trans. Math. Software*. The Algorithms Editor position passed from Lloyd Fosdick to Fred Krogh of JPL in Vol. 2, No. 2, Sep, 1976.

"Mathematical Software III" was held at Madison, WI, March 28-30, 1977, sponsored by the Math Research Center, Univ. of Wisc., Madison with financial support from the US Army and the NSF. This was again a John Rice production. He appointed a program committee consisting of himself, Carl W. de Boor and J. M. Yohe, and session chairs: C. B. Moler, J. H. Griesmer, W. J. Cody, W. Kahan and T E. Hull. The fourteen papers of the symposium were published in the book: *Mathematical Software III*, Ed. John R. Rice, Academic Press, 1977. Included is a paper by Lawson: "Software for C1 Surface Interpolation", pp. 161-194.

IFIP Working Group 2.5 on Numerical Software

IFIP WG2.5 (International Federation for Information Processing, Working Group 2.5 on Numerical Software) (http://go.to/wg2.5). There is an interesting six page write-up by Bo Einarsson in the "informal proceedings" of the 1974 "Mathematical Software II" conference at Purdue about the initial stages that led to the formation of WG2.5. The following indented text is summarized from Bo's write-up.

> There was a meeting on "Subroutine Libraries Being Developed" organized by SIGNUM at the IFIP Congress of 1971 in Ljubljana. The concept of forming an IFIP working group on "Numerical Program Libraries" (Note that this is not the name ultimately adopted for the group.) was raised and IFIP officials recommended the appropriate organizational home for such a group would be under IFIP TC2 (Technical Committee on Programming). The concept was subsequently presented (by whom?) at meetings of TC2 in Zakopane, Aug, 1972 and Munich, Apr, 1973. At Toronto, Oct, 1973, an Organizing committee was appointed, consisting of Einarsson, (chair), Brian Ford, Thomas E. Hull, James C. T. Pool and Christian H. Reinsch.

> The concept was discussed at an open evening session at the 1974 "Mathematical Software II" conference at Purdue. Finalizing discussions were expected at a TC2 meeting and IFIP Congress in Aug. 1974, Stockholm.

I was among those invited to be one of the 13 initial members. Brian hosted the initial meeting at Oxford University in January, 1975. Subsequently, the group held meetings of about 2 to 3 days duration each year. The group was a very useful international sounding board for a variety of projects relating to numerical software, e.g., the BLAS. The main outcome of these meetings was the establishment and continuation of professional working relations between these individuals from diverse countries.

The second meeting of IFIP WG2.5 was at Oak Brook, IL, June, 1976. In connection with that meeting Wayne Cowell of Argonne National Laboratory organized a "Workshop on Portability of Numerical Software". There were 43 participants including Lawson and Fred Krogh from JPL. The proceedings were published in soft cover by Springer-Verlag in 1977 as *Lecture Notes in Computer Science, 57, Portability of Numerical Software*, Ed. Wayne Cowell. Two of the papers involving JPL authors were:

"Two Numerical Analysts' Views on the Draft Proposed ANS FORTRAN", C. L. Lawson and J. K. Reid, pp. 257-268.

"Features for FORTRAN Portability", Fred T. Krogh, pp. 361-367. This described a preprocessor, the "Specializer" that Fred had developed to aid in producing portable code, working under the limitations and uncertainties of the 1966 Fortran standard.

The eleventh meeting of WG2.5 was held at JPL in 1984. I made the arrangements for this meeting with activities running from Sunday, June 24 to Thursday, June 28. Sessions open to JPL employees were scheduled for Tuesday, 8:30 AM to 2:30 PM and Thursday from 8:30 AM to Noon. At these sessions eleven talks were given by members of WG2.5 as well as one talk by a Caltech research fellow, John Bolstad, on multigrid algorithms, and one talk by a JPL project leader, David Rogstad, on the Caltech/JPL Hypercube Ensemble Computer Project. My wife, Dottie, and I hosted a social evening for the group and a few JPL'ers in our home on the Monday evening.

Starting in 1978, the group organized a series of "Working Conferences", holding one about every three years. Each WC was on a selected topic related to numerical software. At these WC's a total of about 100 persons would be invited, selected on the basis of their activity in the topic of the particular WC. A WC would last about five days.

The first WC was on the topic of "Performance Evaluation of Numerical Software", December 11-15, 1978, at the Hotel Gutenbrunn in Baden, Austria. The conference chair and program chair was Lloyd D. Fosdick and I was a member of the program committee. The proceedings were edited by Fosdick and published in hard cover by North-Holland Publ. Co., 1979.

The second WC was on the topic of "The Relationship Between Numerical Computation and Programming Languages", August 3-7, 1981, in Boulder, CO. The conference chair was C.

Lawson. The co-chairs of the program committee were Lawson and John K. Reid. The proceedings were edited by Reid and published in hard cover by North-Holland Publ. Co., 1982.

Gatlinburg and Householder Meetings

The first four "Gatlinburg" conferences (1961, 1963, 1964, 1969) were held in Gatlinburg, TN, and organized by Alston Scott Householder, then at the Oak Ridge National Laboratory. (http://www-gap.dcs.st-and.ac.uk/~history/Mathematicians/Householder.html) Householder had joined ORNL in its Mathematics Division in 1946 and became the Director of ORNL in 1948. He retired from ORNL in 1969. He continued to be professionally active as a professor at the University of Tennesee from 1969 to 1974, and then retired to Malibu, CA. The conferences were continued in other locations, still called "Gatlinburg" meetings until 1990 after which they were called "Householder" meetings.

I attended "Gatlinburg VI, Symposium on Numerical Algebra", Dec 15-22, 1974, in Hopfen am See, near Munich, Germany. The program committee consisted of Bauer (of Munich), Fiedler, Golub, Householder, Marchuk, Todd, Varga and Wilkinson.

I attended the 12th "Householder" meeting not far from Pasadena, at Lake Arrowhead, the week of June 14, 1993. I saw Alston Householder for the last time at that meeting. He was then 89 years old and in quite fragile health. He passed away July 4, 1993.

Other Meetings I Attended

I attended the "International Mathematical Programming Symposium", Aug 27-31, 1973 at Stanford University. There were 523 registrants and 272 papers given. A notable amount of attention was paid to the recent trend to the use of orthogonal methods (Householder and Givens transformations) in optimization codes. Leadership in this trend by P. Gill and W. Murray was noted by a number of speakers.

In 1973, Los Angeles had both a local SIAM and a local SIGNUM organization. They held a one-day Southern California Joint Applied Math Meeting, Dec 1, 1973. There were at least about three annual joint local meetings between these two groups held during this period.

I traveled to Germany, Austria and England, Dec 16, 1974 to Jan 17, 1975. I attended the "Gatlinburg VI" meeting near Munich, the "Optimization in Action" conference in Bristol, and the initial meeting of IFIP WG2.5 in Oxford.

I attended the Dundee Conference on Numerical Analysis, July 1-4, 1975, University of Dundee, Scotland. This was the 6th biennial conference of this title at Dundee. About 200 persons attended. There were 16 invited talks and 45 contributed talks. My invited talk was "On the Discovery and Description of Mathematical Programming Algorithms". The proceedings were published as *Lecture Notes in Mathematics, 506, Numerical Analysis Dundee 1975*, Ed. G. A. Watson, Springer-Verlag, 1976. My talk is on pp. 157-165.

I was interested in multigrid methods in the 1980's. I attended a short course (~2 days) given by Ake Brandt (the "father" of multigrid methods). I never got into this enough to develop any software or deal with any applications myself.

A conference on "Reliable Numerical Computation" was held at the National Physical Laboratory, Teddington, Middlesex, UK, July 8-10, 1987. This conference was in honor of James Hardy Wilkinson, FRS, who had contributed in many unique and significant ways to the development of numerical analysis, particularly numerical linear algebra, from the earliest days of digital computers. He had a wide circle of friends in, and beyond, the numerical linear algebra community. He passed away Sunday, October 5, 1986. The conference organization was handled by Maurice G. Cox of NPL (National Physical Laboratory) and Sven Hammarling of NAG (Numerical Algorithms Group, Ltd.). About 18 technical talks were given plus remembrances of Jim Wilkinson by Gene Golub and Leslie Fox. The proceedings were published in hard cover by Clarendon Press, Oxford as *Reliable Numerical Computation*, 1990. I attended the conference and gave a talk, coauthored with Kajal K. Gupta: "The Lanczos algorithm for a pure imaginary Hermitian matrix", pp. 25-34.

I attended ICIAM 91, the 2nd International Conference on Industrial and Applied Mathematics, July 8-12, 1991, Sheraton Washington Hotel, Wn DC.

Beyond Jpl -- Toward Order from Chaos

The machine constants subprograms, IMACH, RMACH and DMACH from the AT&T Bell Labs PORT library were very useful in the development of portable math software and provided an example for similar developments. We used these names in our MATH77 library with essentially the functionality defined by their authors. "Algorithm 528: Framework for a Portable Library", *ACM TOMS*, v. 4, n. 2, June, 1978, 177-188, by P. A. Fox, A. D. Hall and N. L. Schryer.

W. Stan Brown's model of computer floating point arithmetic, also from Bell Labs, clarified thinking about the use of floating point arithmetic.

The Unix operating system, also from Bell Labs, (Ken Thompson and Dennis Ritchie, 1969 on a PDP-7 computer) eventually became a defacto standard, so most operating systems in computers for engineering and scientific computing were Unix or very similar to Unix. This is a big improvement from the wild diversity of operating systems from about 1960 -1975.

The IEEE standard for binary floating-point arithmetic is now implemented in the arithmetic units of essentially all computers used for engineering and scientific computing. This is a very big improvement from the diversity of word-lengths and implementations of floating point arithmetic in computers prior to its adoption. The following indented details are from the *Encyclopedia of Computer Science, Third Edition*, 1993, pp. 86-87.

.

> In 1978 the IEEE established a committee to develop a floating-point standard. At the time at least 20 different floating-point formats existed in different computer models. The development of the standard is indicated by the following publications:

J. Coonen, W. Kahan, J. Palmer, T. Pittman and D. Stevenson, "A Proposed Standard for Floating-Point Arithmetic", *SIGNUM Newsletter*, Oct, 1979.

W. J. Cody, "Analysis of Proposals for the Floating-Point Standard", *IEEE Computer J.*, March, 1981.

"Binary Floating-Point Arithmetic", IEEE Standard 754, *IEEE*, 1985.

The IEEE standard was being implemented in hardware by Intel and other manufacturers while the development of the written standard was under way, so computers adhering to the standard came on the market in a timely manner.

Beyond JPL – Collection and Distribution of Math Software

COSMIC was an organization set up by NASA to collect software produced by NASA projects that might be of more general usefulness. COSMIC was to collect, catalog, advertise and distribute such software to requestors, both within and outside of NASA. COSMIC probably had the greatest chance to serve a useful purpose from about 1975 to 1990. I think NASTRAN was their most popular package. Our group contributed software to COSMIC during that period. After that there was much more commercially available software. NASA gradually withdrew funding from the University of Georgia Foundation that operated the COSMIC function, and COSMIC eventually could not create enough income to continue. COSMIC officially quit taking orders in Feb, 1998. (http://www.cosmic.uga.edu/) The COSMIC inventory was transferred to the Open Channel Foundation (http://www.openchannelfoundation.org/) and (http://www.openchannelfoundation.org/cosmic/) which performs a collection and distribution function for a number of institutions.

Nelson H. F. Beebe, University of Utah, Salt Lake City. (www.math.utah.edu/~beebe)
I became aware of N. H. F. Beebe about 1979-1980. I believe his background was in chemistry. He was very active in trying to improve the use of computers by people in his field. He had developed a plotting package, PLOT79, that was much more portable across different computers and different plotting equipment than was usual. It included a variety of 2D and 3D coordinate transformations of particular use in his field. I believe he provided the package to requestors at just the cost of shipping a tape. He had also developed a tape format for interchange of data among scientists. He was very interested in our SFTRAN3 Structured Fortran preprocessor. We gave it to him. He made improvements in it and added it to his collection of software that he distributed.

From his current web site it appears that he is very active in TUG – the TEX User Group, and continues to make lists of interesting things.

A large collection of mathematical software in maintained and made available at no charge at Netlib (http://netlib.org). This collection includes all the software refereed and published via ACM TOMS.

A large collection of statistical software in maintained and made available at no charge at Statlib (http://www.stat.unipg.it/stat/statlib/).

Beyond JPL – Math Packages, Libraries and Environments

Ronald F. Boisvert authored an excellent write-up on Math Libraries & Packages in the *Encyclopedia of Computer Science*, Third Edition, Anthony Ralston and Edwin D. Reilly, Van Nostrand Reinhold, 1993, pp. 1229-1232.

J. H. Wilkinson and C. Reinsch, *Handbook for automatic computation, Volume II. Linear algebra,* Springer-Verlag, Berlin, 1971. (This was preceded by Volumes Ia and Ib about the Algol language, which is used in Vol. II describe computational algorithms.)

Math libraries. SHARE, Harwell, NAG, IMSL, SLATEC, PORT

Math software packages. EISPACK, BLAS, LINPACK, LAPACK, SPOOLES, ARPACK, NASTRAN, PLOT79

Garret N. Vanderplaats – Book and related software for finite element analysis and optimization of designs. The Book is *Numerical Optimization Techniques for Engineering Design*, Third Edition, Fourth Printing, 2001, published by his own company: Vanderplaats Research and Development, Inc., Colorado Springs, CO. *(www.vrand.com)*. He was awarded the 2002 Multidisciplinary Optimization Award in Sep, 2002, at the Ninth AIAA/ISSMO symposium on Multidisciplinary Analysis and Optimization.

William H. Press, Numerical Recipes – Books and software. (www.nr.com).

Problem solving environment for math, engineering and science. Matlab, Mathworks, Cleve Moler.

Miscellaneous Topics

Code and published papers on interval arithmetic by Richard Hanson. CM-160, "An Interval Arithmetic Package for Use with FORTRAN IV on the IBM 7094 Version I and II", R. Hanson, Feb 20, 1967.

CM-538, "The LIPACK Package for Large Integer Arithmetic", Lawson, Jan, 1988, Rev. 6-89, Word Perfect file: LIPACK4.wpj. Includes solution of underdetermined systems of equations having integer coefficients, i.e., Linear Diophantine equations.

JPL application: Polynomial encoding of data to control the pointing antennas at the moon and Venus. (about 1963?). Eliminated the need for about 40 hours per week of a technician's time.

CM-203, "French Curve", Hanson, Lang, Campbell, Sep 9, 1968. A Fortran subroutine for constrained curve fitting using B-spline basis functions. Originally allowed constraints on values and derivatives. Around 1980, Lawson added features to allow constraints also on integrals and

to use the de Boor parameterization of B-splines. This code, named DSFITC, is in Section 11.5 of the manual for the MATH77 and *mathc90* libraries, Release 6.0, May, 1998. (John Rice also used the name “French Curve” for a concept and code of his.)