An interview with
G.W. “PETE” STEWART

Conducted by Thomas Haigh
On
March 5-6, 2006
In Washington, DC

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

Society for Industrial and Applied Mathematics
3600 University City Science Center
Philadelphia, PA 19104-2688

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

© Computer History Museum
Mountain View, California
ABSTRACT

Numerical analyst G.W. (“Pete”) Stewart discusses the whole of his career to date. Born in 1940, Stewart spent his teenage years in Knoxville, Tennessee. In 1957 he enrolled in the University of Tennessee, where he majored in mathematics and supported himself through programming work at the nearby Oak Ridge Gaseous Diffusion Plant. Stewart graduated in 1963, winning a Woodrow Wilson fellowship for graduate study at Princeton. He stayed only one year, returning to his job at Oak Ridge where he remained until 1966 with exception of a year spent as a programmer in General Electric’s computer division. He enrolled in the Ph.D. program at the University of Tennessee, earning his Ph.D. in 1968 under the direction of Alston Householder. Stewart discusses his attendance at a University of Michigan Summer School in numerical analysis, his relationship with Householder, computing facilities at the plant, a fellowship at the Oak Ridge National Laboratory, and his thesis work. After receiving his doctorate, Stewart spent four years as a member of the mathematics faculty at the University of Texas, Austin. During this period he created both the Bartels-Stewart method and his best known paper, a collaboration with Cleve Moler on what became known as the QZ algorithm for eigenvalue problems. After spending two years at Carnegie Mellon University, in 1974 he arrived at the University of Maryland, College Park where he has remained since. Stewart discusses the evolution of the computer science department over three decades, including his involvement in its ZMOB parallel computing project and in the IFDAM and MIACS research centers. In the late 1970s Stewart, in collaboration with Cleve Moler, Jim Bunch, and Jack Dongarra, was one of the principle contributors to the LINPACK numerical linear algebra package and associated manual. He discusses the project’s origins, scope, methods, and impact together with the work of each project member. Stewart also surveys the most significant of his many research papers and contributions to mathematical methods, his text books in the field and his interest in the translation of historical mathematical works. He explores his involvement in several professional bodies, including the ACM’s SIGNUM, SIAM, and the Gatlinbough/Householder series of meetings in which he was a long-standing participant and organizer.
HAIGH: Thank you very much for agreeing to take part in the interview.

STEWART: It’s my pleasure.

HAIGH: I wonder if you could begin by talking in general terms about your family background and early life.

STEWART: Well, let’s see. It’s an unlikely combination. My father grew up in North Dakota. My grandfather was a preacher and part-time editor of a local newspaper. My mother grew up on a farm in Alabama. They met in Washington, D.C. during the Depression, where people naturally gravitated there. My father had a degree in journalism. They met there and married, and I was born in 1940 in Doctor’s Hospital in Washington, D.C. So I was one of the few natives of the city.

HAIGH: Then you had mentioned in your biography that you moved around a lot when you were growing up.

STEWART: Well, we spent my first five years in Washington, living in Arlington, where you spent the night. And then my father got a Nieman fellowship to visit Harvard for a year. This is a journalistic award. We moved up there. As a matter of fact, I started first grade in Cambridge, at a school called Shady Hills School. I always liked to kid David Young, who was in Texas when I came there, that he started graduate school in Harvard the same day that I started first grade in Cambridge. So we stayed there for a year and then moved very briefly back to Washington, D.C., and then my father took a job with the State Department as a Press Advisor.

So we moved to New York, in particular, Queens. It was a very interesting place. We moved into a series of apartments for U.N. personnel, people who were at the U.N. At the time, the U.N. building had not yet been built and it was meeting out at Lake Success, out on Long Island. We lived there for about three years, and I attended the city schools in New York. Then we later on moved out to Hempstead, Long Island, where I spent three more years going to a local school. It was high pressure job and my father got tired of it, and he became Assistant Director of Information for the Tennessee Valley Authority, so we moved down to Knoxville. In fact, his boss was a Nieman Fellow at the same time that he was, so...

HAIGH: And that was in 1953?

STEWART: Yes. Yes, so I was 12 going on 13 at that time.

HAIGH: Was Knoxville a contrast with New York and Cambridge?

STEWART: Quite a contrast. The school I came into was basically a rural school, starting to become suburban at the time. I lived very near the boundaries where there were only cornfields out there, and the suburbs were slowly moving out west, to Knoxville. Now, they’ve moved almost to Oak Ridge, how much things have changed. But at that time it was all country. So the school was quite a mixture of people, and of course, just the general East Tennessee manner was quite a change for me. It took me a couple of years to adapt, but in point of fact, I think it was a very good thing for me that we did move to Tennessee, and I’m very happy for it.

HAIGH: So what kind of benefits do you think you got from living there?

STEWART: Well, one thing that I did was that the schools were not great. The teachers were nice enough, but I discovered pretty much that I had to be my own person, you know, that I had to control my own learning, and this sort of thing. The other benefit that I think is much more
important, was that I learned to get along with a lot of different types of people, you know. Even now, if I can’t get a smile out of a grocery store clerk, I feel like I’ve somehow missed something or, you know, not done right. And this has been, not so much useful, but a very nice benefit to come out of the change. So looking back on it, I think I’m quite happy that we did move down, although there were times when I wasn’t as happy.

HAIGH: Yes. And your full name is Gilbert Wright Stewart, the Third. So at what point did you pick up the nickname of “Pete”?

STEWART: Well, my grandfather picked it up. He was the first Gilbert Wright Stewart, and he picked up “Pete”. It’s not known for sure, how. Some say it was a college nickname, but the family story that everyone likes to repeat is that he was a newly minted Presbyterian minister, and went out to South Dakota to a cowboy town out there, and it was basically a town with a hotel on one side and a bar on the other side. So my grandfather got settled into the hotel, and then he went over to the bar and talked to the bartender and said, “Look. If you’ll close down the bar for an hour on Sunday, I won’t preach against liquor.” And so the person who owned the bar thought that was a good thing, and so the cowboys would come in and sing hymns for an hour and then go back to their drinking. He was just a marvelous person, I think well known all over the state by the time he died. You can see why, from that touch.

HAIGH: So you were also Pete Stewart the Third?

STEWART: Yes, I was Pete Stewart the Third. And my son is “Michael”. [Chuckles]. We decided that that was enough. And I dropped “the Third” as soon as I could, and after my grandfather died, my grandmother said that my father had become “Senior”. Well, my father didn't want to drop the “Junior” and I then decided to drop “the Third”, and unfortunately, it’s sitting on a couple of my papers.

HAIGH: Yes. And I found that from the mathematical genealogy project.

STEWART: Yes, right. Actually, I sometimes regret that I hadn’t published just under the name of Pete Stewart. It would have stopped a lot of confusion.

HAIGH: So what kind of interests did you have as a teenager?

STEWART: As a teenager?

HAIGH: Or younger, if that’s anything you would wish to pursue?

STEWART: Well, I always had an interest in science. I remember going round right after the Nagasaki and Hiroshima, saying I wanted to be a nuclear physicist. I didn't have the foggiest idea of what that was, but I wanted to. But generally speaking, my interests have been fairly broad, not just confined to science and this sort of thing. I’ve loved reading of all kinds, you know early Hardy Boys and Oz books, and it’s graduated to when I was in high school, I picked up a copy of Gibbons’ *The Decline and Fall of the Roman Empire*, and I’m an avid reader of history. So my intellectual interests have been quite broad, but science has been the primary one, and within that, mathematics is the big one. Other than that, once I got settled in East Tennessee, my life was pretty normal there.

I did skip two grades there, so it was an unusual social thing. When we moved from Jamaica to Hempstead, I had had a year in which I was very sick, and when I got to the class there, I would have been in the fifth grade, and was just overwhelmed by the book, the new people, and a teacher who I thought was rather mean, and et cetera, et cetera. So I asked my parents to put me
back in the fourth grade, which they did. And I spent three years there, a grade behind. Incidentally, in the sixth grade, I got that mean teacher, and she turned out to be the best teacher I’ve ever had, bar none, actually. She was wonderful. But in any case, when we moved to Knoxville, I was in the seventh grade, and it took me one day in that class to realize that this was not for me. You know, it was too little intellectually, although I wouldn’t have thought of it in those terms. So I did the same thing. I jumped into the eighth grade, which happened, interestingly enough at that time, to be over in high school. There was an elementary school and a high school together. So I moved over to the high school, which was a nice transition for me. And then I completed that my high school career in three years and went on to the University of Tennessee.

HAIGH: Did you have any technical hobbies like chemistry sets, or building radios?

STEWART: I had chemistry sets and the like. I was never a hands-on type person. So they never went far when I was in high school. I was quite interested in entomology, insect collecting and had a small collection of that. But my hobbies didn't really tend toward the technical, or at least the hands-on type technical. That’s been largely true throughout my life. I’m not a good handyman. In my hands, a screwdriver’s a deadly weapon and dangerous to anyone in my vicinity. So as I say, I’ve not been a good technician of any kind.

HAIGH: I know you have a story that you want to relate about your high school algebra experiences.

STEWART: That’s really where I got started in mathematics. I suppose you know from your own interviewing, but there’s sort of a readiness for a particular area of mathematics. If you aren’t ready for it, you just don’t get it. You know. And you see this, again, having taught a good deal of mathematics, you see this in your students. Sometimes it’s just a matter of time. Sometimes it’s a real barrier that the student will never get over. So when I was I guess about 10 or 11, I tried to teach myself algebra. I could do the manipulations, but they had no meaning for me. Then when I moved to Knoxville in the ninth grade, of course the natural thing was that I took algebra I, and all of a sudden it made sense to me. Well, my teacher was a woman who was an indifferent teacher to the ordinary students, but she was very good at finding the good students and motivating them to perform at their best, often by leaving them alone [chuckles]. And in my case, when I discovered that I was really progressing very fast, she said, “Why don’t you just sit in the corner in the back of the class and work homework problems and give them to me.”

And so in about 12 weeks, I had finished the first year algebra course. It went on in the same first year algebra classroom to study second year algebra. At that time, Mrs. Ayeslinger went to the county superintendent of schools and arranged that I could get credit for these courses by passing achievement tests. So I did that, and I did the same thing with geometry and this sort of thing. So this is part of what allowed me to graduate from high school in three years. So by my senior year, I was taking trigonometry and working on teaching myself calculus. So I well started along, but it was a real eye-opener. Suddenly there was this thing that was beautiful and I could do easily. From then on I was always a mathematician in some sense of the word, but there are many senses of the word, as I’m sure you know.

HAIGH: Did you know at that point that you would be going on to major in mathematics in college?
STEWART: No, and I didn't even necessarily know it when I was in college. I had a very checkered career in college with a lot of different sort of majors. At one time I was going to be an engineer and physicist. I basically had the courses for a double major in math and physics in college, at the University of Tennessee. At other times I was majoring in psychology and was going to be a psychometrician. I even considered going into medicine. I actually visited a fraternity brother who had gone to the University of Tennessee campus in Memphis and visited him there. That made it pretty clear to me that I didn't want to be a physician [chuckles]. So I always had a math major as an ace in the hole, not matter what, and it was the subject that I was clearly best in. But it was not necessarily what I thought I was going to do.

HAIGH: So what year did you enroll in the University?

STEWART: That would 1957.

HAIGH: And you went straight from high school?

STEWART: Yes.

HAIGH: So did you consider going to a university further from home?

STEWART: Yes, I did, to some extent. At the time I was only 16 going on 17, but still 16. And I learned later, one of the possibilities was Carnegie Tech, which later merged with Mellon Institute, to become Carnegie Mellon. And I found out that we had relatives in Pittsburgh, where my grandfather was actually from. And they contacted them and they said this was probably not a good idea. This school was probably not a good idea for a 16-year-old boy, since these people worked hard and drank hard and everything else. And the other possibility that I briefly considered was St. John’s College in Annapolis, the Great Books college. But somehow I instinctively felt that this was too constraining of an environment. And you know, I was very happy to go to Tennessee. There was also finance to consider—it was much cheaper for me to go there. In some sense, the middle class was not as prosperous as they are now. The college expenses were really, and even as small as they were, compared today, were a major part of a typical professional person’s income. So it all decided to go there, and I went and have no regrets. I had a good time there.

HAIGH: So how unusual was it to be going to college at that age?

STEWART: Oh, quite unusual. Typically the person would get out of high school at 18 and then go on to college. I had a rocky start. I mean, in summer school I took a summer course in German and English, and the German was to cover three semesters in one summer, and I just was not prepared for that and didn't have the study habits or the discipline to do it and then dropped out after one-third of the course. And I didn't do so well in English, which is sort of interesting, because my life has been writing since then. My whole family is writers in some sense. My father was a journalist, and my sister is a very good at expressing herself on paper. Then later I found out it was unnecessary, because my scores on my entry test would have placed me out of that English course, so I got a C in that.

HAIGH: So was that an English writing course?

STEWART: Well, it was a basic freshman English course, which included writing and some reading, but basically a writing course. Perhaps one of the reasons is that although I’m a very quick writer when I’ve got all my materials organized, it takes me quite a bit of time to organize things so that I can write them. And writing in class time is not a very good way to do that.
HAIGH: What were your general impressions of the university?

STEWART: Well, it was a pretty good place. I joined a fraternity the fall that I got there, Kappa Sigma, which was very nice because again, it exposed me to a much wider variety of people than I would have met. And well, I lived at home for the first year, but after that I lived at the university for large parts of the time in the fraternity house or outside apartment. But I got exposed to a much wider group of people than I would have if I’d just been in a dormitory and made, you know, the five friends that a lot of people do under that situation. So that was very good. The university itself was not, of course, a top university, but one of the things you have to realize about the school system and the university system at that time is that at almost any major state university you could get a very good education in a wide variety of subjects. I mean they certainly had their fair share of really awful teachers, but within any department you could find one or two who were really good and could help you. And, so for me, it was just a cornucopia of very interesting things to study.

One thing I did that helped my average immensely was I put an awful lot of the required courses until my junior and senior year. So you know things like, Music Appreciation and Psychology, and this sort of thing, I took it at a later time. And by then, of course, they were duck soup—you know it was very easy to pass. But also, I got lot more out of them than I would have, if I had taken them earlier. For example, I’d always liked music and was interested in it, but my lifelong love of music started at that time. As a matter of fact, I took two quarters of it, an unnecessary quarter, just because it was so much fun, the classes. The music departments had very good teachers and it was really wonderful. So for a year after learning all this stuff, I was listening to even popular music and hearing it in a completely different way than I had ever heard it before, and that’s stayed with me.

Another thing is, I went back and re-took my German. You had to have a language. And I did quite well in it, in the vein that, again, I was basically one-quarter shy of a minor in German when I graduated. I got introduced to such things as German lyric poetry, which really sparked my interest in poetry in general. So it was just a wonderful educational experience. As I say, it was very irregular about it, you know. I jumped from this to that, but I managed to graduate with a respectable average, and of course, a very strong math major, which is what I finally declared as my major.

HAIGH: What were your experiences like within the major?

STEWART: Well, actually I didn't have a lot to do with the department. My life was more with my fraternity, and as we’ll get to later, with my friends at Oak Ridge and the people I worked with there. So I basically took my courses and sort of steered clear of the department itself.

HAIGH: So that’s when you didn't form any significant relationships with the faculty?

STEWART: Not really. I mean, some. I just wasn’t around the Department enough to do that. But yes, I mean obviously, I had to get very good recommendations to get into Princeton to graduate school, so I was well acquainted with them. Orville Harrold was one person. He was a topologist.

As a matter of fact, when I took a graduate course while I was still an undergraduate from him, and he was instrumental in his getting me into Princeton and Yale—I had acceptances from both of those places. But the Department itself wasn’t very much a part of my life. It was a place I went to take courses, and some of the courses were excellent. In fact, I thoroughly enjoyed them.
HAIGH: Were there any courses or instructors that you had there that made an influence on how you would approach things subsequently?

STEWART: Well, there is one story. In the middle of my junior year, I got a little tired of school and dropped down and went back out to Oak Ridge. My dad was jumping around a little, but we can get to that later. When I came back, I had a third quarter of Advanced Calculus to finish. But the only section that was being taught, was being taught by a wonderful guy by the name of John Neuberger, and he was teaching it on what was called the Moore method of teaching. This is a form of teaching in which you are given a set of theorems, and then you have to go off and prove them for yourself. And John Neuberger said, “Oh, you couldn’t possibly come into the third quarter of this class, and survive or do it.” He was very kind about it, but of course it made me angry. And so I borrowed the notes from someone and sat down at my typewriter and typed them all out so that I would have all the theorems—now, not the proofs, but all the theorems—and then I came in with the resolution. You were supposed to prove theorems at the board, and so as I say, I was angry enough, and so I told myself I’d proof one theorem for every theorem that anyone else proved. And of course, I had no trouble proving the theorems, but obviously, I couldn’t monopolize the class in that way. But the point is that this was basically advanced elementary analysis, and from it, I really learned what a proof was and how to manipulate the mathematical entities and this sort of thing. So this was a highly profitable course for me.

HAIGH: And did you say his name was Neuberger?

STEWART: Yes. He then went off to Emory, and I met him occasionally at meetings.

HAIGH: So just before we transition then, to talk about your work at the Oak Ridge Gaseous Diffusion Plant, did computing or any range of numerical analysis topics have a place in the undergraduate curriculum at that point?

STEWART: No. Absolutely not. Well, first, there were no computers at the university. This was in the late 1950s, and only major laboratories and government institutions had computers. They were too expensive. They were just beginning to make an impact on business and this sort of thing. So that, well Stanford would have undoubtedly have had a computer at that time, but I imagine that a university like the University of Tennessee would definitely not. So I was basically completely unaware of computers. The engineering courses that I took were all slide rule, and numerical analysis was basically unknown at that time.

We can transition into Oak Ridge. I found out that I was going to be there, along with a good friend who had been a high school buddy of mine who had gone away to a Wesleyan University. We were both hired for the summer, and we were going over a FORTRAN book that he had somehow unearthed I saw the “X = X + 1” And of course, I had no concept whatsoever of what an assignment statement was. So this seemed to be a mathematical impossibility. What was all this about? Again, when I knew that I found out that I was going to be a programmer, I immediately thought that I would have to go out and study mathematical programming, you know, linear programming and this sort of thing. So this is how completely unprepared I was, or not unprepared but just ignorant of the computing milieu at the time.

HAIGH: Now as you’d been growing up, had you ever read, say, a popular science book about computers or a magazine article about computers? Did you have a sense of what they were?

STEWART: No, not really, not at all. The kinds of books that I would have read were Mathematics for the Millions and Fun with Mathematics, and this I was very well acquainted
with number systems and this sort of thing, including binary. But for me, the binary system was just one system among many and had no computation significance for me at that time.

HAIGH: So you arrived at the Oak Ridge plant in 1959 as a summer intern?

STEWART: I guess you could call it a summer intern, at that time. I was a full-fledged employee in the sense that I was employed by Oak Ridge, had to get a Q clearance. I didn't deal with any top secret documents, at least to my knowledge, but the gaseous diffusion plant was a critical part of our nuclear program. This was where uranium was separated by gigantic buildings that basically were a cascade to separate U235 from U238 by passing it through diffusion barriers. And of course, the nature of the barrier was a top secret. So just being on the place, you would need to have a security clearance. So in that sense, I was a regular employee. Union Carbide ran it, not by government, and I had to take their physical every year, so I got a free physical out of it.

HAIGH: Did many of the students from the university work at the plant?

STEWART: No. None that I knew from the university, none of my acquaintances actually worked out there. There were students from the university, a close friend of mine, Bert Rust was an engineering Co-op student, but we met at Oak Ridge and not at the university. As a matter of fact, he the person that I share an office with at NIST, when I’m out there on during the week.

HAIGH: So other than being in Oak Ridge, were there strong ties between the national laboratory and the gaseous diffusion plant?

STEWART: Practically not. Certainly so far as I knew and in general, and not any. The laboratory, at that time, got itself in a little bit of a bind. They went and built their own computer, or at least, I think there was more than one other kind. It’s called the ORACLE [Oak Ridge Automatic Computer Logical Engine]. And they got their own computer, and that turned out to be a bad move because they were stuck with it. And at the time I moved there and came there in ’59, the gaseous diffusion plant had acquired the supercomputer of its day, which was the IBM 704. So as far as actually doing serious computing, the gaseous diffusion plant was actually the place to be. Of course, it’s not a research environment, but that’s not really what I needed at the time.

HAIGH: And you were hired to work as a programmer?

STEWART: Yes, essentially that. Our training regime was that we spent four days on the 704 assembly language and then a one-day introduction to FORTRAN, and then we were turned loose on the poor suckers of customers. It took me two or three years to develop a decent programming style after that, as you can well imagine.

HAIGH: Were you using training materials supplied by IBM?

STEWART: No. It was just basically one of the persons there who had been programming just gave a course. There was, it was basically the IBM 704 manual that was our text.

HAIGH: And how large was the programming staff?

STEWART: Oh, that would be hard for me to say. It was significantly large. There were maybe 20 people, and that would be reasonable. There was a whole bunch of keypunch support and everything else. It was a growing organization. It was doing all the computing for this large installation. It was responsible for increasing the productivity of it by various mathematical
programming type optimizations and this sort of thing. Managing the cooling towers, that was a problem, and in simulating or discovering how they behaved and this sort of thing. So it was a well-supported group with quite a number of people. In my class, there were about eight people, some of whom were students like myself, and others who were coming in for permanent employment.

HAIGH: And did most of the people have backgrounds as mathematicians? At least the ones that you knew?

STEWART: No, many of them were engineers. They would have been coming from engineering backgrounds. As a matter of fact, I would say that most of them did not have backgrounds as mathematicians.

HAIGH: Was there a separate team of operators who would be responsible for loading jobs?

STEWART: Oh, yes, very definitely. The thing was a hands-on type operation. It was contained in one gigantic, very well-air-conditioned room. If the air conditioning failed, many people had to run out and grab the vacuum tubes out of the computer. Or if the power failed, or something like that. Otherwise, the heat generated with everything would actually melt them down. So you would come in and there was a dispatcher. Betty Christian I believe was her name. They had was a long bar, metal templates would hang on the bar. The bar had the time of day over it, and the templates were in five-minute, ten-minute, thirteen-minute size. So they would write your initials on that and put it down on the thing, and that was their scheduling, very effective.

HAIGH: And then they would take the job and run it for you.

STEWART: No, no. Then you would show up in the machine room, and hand it to the operator with a deck of cards, and they would run it, and you’d bet a cup of coffee on whether it would run or not occasionally, and it was all very informal.

HAIGH: So you wouldn’t run the job on the machine yourself, but you would stand there with the operators, who would run it?

STEWART: Yes. The operators would generally do that. But we were definitely there at the machine while it was being run. This was just for a couple of years, and then they got better batch operating systems and everything. And then we quit going into the machine rooms.

An interesting sidelight was that we generally knew each other and called each other by our initials, because everyone got used to seeing each other’s initials on these little templates. You know, it was a joke, you know, but it was done. And so someone would say, “GWS, let’s go get a cup of coffee!”

HAIGH: It’s a little like calling someone by their email address.

STEWART: Yeah, a little like that.

HAIGH: How reliable was the 704?

STEWART: Very. It was really a nice, sweet machine, with very little down time that I can remember.

HAIGH: Were you writing programs in FORTRAN?

STEWART: Yes. Yes. And I did some assembly language code. Oh, I should add that we did have to know assembly language, even if we weren’t writing in that, because our major way of
correcting codes was to put octal patches into the decks that the rotor would read and actually place the new instructions in the appropriate positions in the compiled code. Compilation was very expensive. It was time-consuming. Everything had to be written out on tapes and this sort of thing. The machine originally had 16K of 36-bit memory and this expanded to 64, which was the maximum address space at 704.

HAIGH: So if you made a change to the FORTRAN code, how long would you have to wait for it to be re-compiled and run again for you?

STEWART: Well, you would just have to do it. You would have to re-compile it, and the main thing was that you would have to stand around and wait for it to compile. So it was just much easier to make your patches and go in and run it. It’s much easier to get a five-minute slice than a fifteen-minute slice.

HAIGH: Oh. So it would take about fifteen minutes if you needed to run the compiler through, or maybe even longer?

STEWART: Well, I don’t remember the exact times, but it was significant and there was a general tendency to avoid re-compilation until things got really hairy and just for the sake of your program, you needed to get a fresh version.

HAIGH: So were the programmers there in the computer center the only people who were writing programs or would there also be people in other areas of the plant who might be doing their own FORTRAN programming?

STEWART: By and large, I would say at that time we were mostly the only ones. And of course, this was actually where I got my start as a numerical analyst. The machine was relatively new. They had had a 650, and that was a very difficult machine to program. So with FORTRAN, the 704 was much easier, but not many people had mastered the art. So they had this machine, and not a lot of people who knew how to use it, and a lot of people who thought they wanted to use it or wanted to use it. So even though I was still wet behind the ears, I was basically thrown out, working directly with various people who wanted to use the machines, scientists and engineers, and we would. I had to pick up everything I knew, or everything I did, on the fly as I was going. If they wanted to integrate something, I would have to read up on numerical integration and do it. I had a ball.

HAIGH: Now other than the built-in FORTRAN functions themselves, was there any kind of library being developed at that point, or would you research from scratch for each job?

STEWART: There wasn’t much in the way of library resources. There was the FORTRAN library, but beyond that… there was the SHARE organization that had a library.

HAIGH: Yes. They were active fairly early on, so I think they would have had hundreds of routines of varying quality in their library, by that point.

STEWART: Yes, but I never really had an occasion to use any of them. And it was fairly early. I don’t know how many they would have had at time. But in any case…

HAIGH: Well, I can actually show you some of their materials after we stop the tape. I have some scanned things from in my computer from the archives. But it does seem that they had an impressively long list of routines. I also know that there were some quality issues associated with them. But it sounds like you never even tried one.
STEWART: No.

HAIGH: Was that because it was more fun to do it yourself?

STEWART: Well, for most of the things that I was doing it would be more trouble, in many ways, given the communications at that stage, of trying to go out and get a program. After all, for numerical integration at that stage you would be using something like the trapezoidal rule, or a Gaussian quadrature, and you’d just look up the weights and the abscissas in a handbook and program it up. And as far as more difficult jobs, like solving differential equations and this sort of thing, I would be very surprised if there were any reliable routines at that time in the SHARE Library, or any really reliable routines at that time. It was some years later, that we began to get really reliable routines for, say, solving ordinary differential equations. That was a major area of research in the early computer days of numerical analysis.

HAIGH: So when you realized that you needed to perform a particular mathematical operation, would you go to a textbook and just look at the standard methods that were presented there?

STEWART: Yes, by and large. We talked with each other a lot, so it wasn’t that I was entirely on my own. But the major work was primarily translating from the engineering or the scientific problem. For example, you might get someone who wanted to solve a non-linear equation, and the real problem would be to talk with them for a while and find out, and then you’d find out that Newton’s Method would be perfectly fine. They had some nice estimates for their starting points, for their values, and the engineers were usually pretty good about that, because the engineers at that time had to do an awful lot by intuition. So they knew a lot about their problems—if not mathematically, then the gut feeling for what it should be. So that was probably a major part of the job, and then there would be always something to learn and to execute it.

HAIGH: How about your personal feelings toward programming? As you began to experiment with things and started writing programs, was it something that you found that you enjoyed doing a lot?

STEWART: Oh, I love it. I still do. I mean it’s wonderful when I can spare the time to get down to do some serious programming. I wasn’t a very good programmer for a little bit. It took some time. But after a while, you know, I seriously concentrated. For example, of course, by this time I was a fairly good programmer, but when Dykstra came out with structured programming and this sort of thing, I took it very seriously. And since I was still working in FORTRAN, I actually made formal constructs in FORTRAN to simulate the structures of the If-Then-ElseIf and this sort of thing which you get from that. That leads to an interesting story that I can tell when we get to LINPACK.

HAIGH: I’ll make a note of that. At this point, were there any kinds of standards that were being set for the Oak Ridge Plant in terms of indentation or document or anything like that?

STEWART: No. No, this was completely anarchistic. That kind of thing belongs to a later era.

HAIGH: So were you doing basically the same kind of work there, from that first summer internship in the summer of 1959, in the summer of 1961, and in the summer of ’62?

STEWART: Well, part of the spring of 1962, because as I mentioned, I dropped out of the university in my junior year and they gave me a consulting contract. As a matter of fact, from then on, in addition to my regular summer things I had a consulting contract so I could go out to Oak Ridge one day a week if I wanted to and work out there.
HAIGH: So you dropped out in a “semester off” kind of way?

STEWART: Just a semester off. I was tired of school, and Oak Ridge seemed like a lot more fun.

HAIGH: Then after you graduated, you were full-time then, in 1963?

STEWART: Yes.

HAIGH: So were you basically doing the same job all the way through that period?

STEWART: Yes, but it grew along with me. You know, I mean that’s like, as my range increased, I did more sophisticated things. They gave us a lot of latitude to do personal projects or things like that. So for example, I wrote a very elementary contour plotter, which was widely used because there was none other at the time. But it was more just for fun, to write it. There was a group of us who were fairly close friends out there, and we were all given a reasonably wide latitude to do what we wanted. There were more problems than we could possibly work on, so we had a lot of choice. And as I say, if we wanted to work on utility routines or this sort of thing, it was just fine.

[Tape 1, Side B]

HAIGH: So you mentioned that you had a number of colleagues that you were working with. Is there anybody you feel that you should mention by name, or any that had an influence on your later career, or anything like that?

STEWART: Not really, except for the fact that we were good friends and did lots of things. We’d go occasionally for hikes in the Smokies and this kind of thing. And these people, by and large, stayed around at least 1963 or so. So it was good. It was a nice group, but not having any direct influence on my later career.

Oh, I can mention one. There was Bob Edwards, who had a Master’s degree from MIT, came there. And I was given a project to do. As a matter of fact, I walked in from lunch and my then boss put a book on Bessel functions on my desk and said, “Solve this,” and there was a problem there. So I worked and worked at it, and I did things much too elaborately, and really was getting nowhere, because I was trying to do too much. Then Bob Edwards, with good common engineering sense, just went straight to the problem, wrote the shortest program that would solve it, and I learned a lot from that. But as far as long-term influence, I wouldn’t say so.

HAIGH: So was your program too generalized? Were you trying to get too high an accuracy?

STEWART: No, not so much. It was just trying to do too much and not really seeing how to cut out the clutter from it. It’s hard to describe, and actually it would be very, I don’t remember the details that much, but definitely, what I was doing was complicated and unwieldy and what Bob Edwards did was short and elegant.

HAIGH: So then chronologically, we should probably mention your experiences at Princeton, at this point. And you also mentioned that you had a chance to go to Yale, as well.

STEWART: Yes. In my senior year in college, or actually it was a little beyond my senior year, because my time at Oak Ridge put me back somewhat…. Naturally you applied to go to graduate school, and it was fairly clearly in mathematics that I wanted to go. At this time I was taking a graduate course in topology and you know I was pretty much set up. I was nominated for a Woodrow Wilson fellowship by the department head. If I didn't know much about the Department, they evidently knew more about me. So they nominated me and I actually got one.
Then I got the right kind of recommendations and letters that would allow me to go to Yale or Princeton, and was accepted there. And of course, at Tennessee they wanted to keep me there. I decided to go off to Princeton, which of course was one of the prestige places to go. Now I won’t say that it was a mistake, but I definitely got in very nearly over my head there. I was competing against people from Harvard and other schools. I went to study algebraic topology, just because that was a very “in” thing in mathematics.

Let me back up a bit. During this whole period when I was at Oak Ridge and at Tennessee, when I was shuffling back and forth, the mathematics and the numerical analysis I was doing at Oak Ridge were compartmentalized. I mean I used the mathematics and the numerical analysis, there was no question of that, but it was a tool and they weren’t integrated in any sense. So when I went to be a mathematician, I wanted to be the kind of pure mathematician that was the thing that every good mathematician should strive for.

HAIGH: Right. You didn't have the idea of numerical analysis as a research area that might be respectable in it’s own right.

STEWART: Yes. Or even as a research area. That might be a better way of putting it.

HAIGH: And you’d also mentioned actually how much you loved programming. You also didn't have an idea that there might be graduate study possible in that area.

STEWART: Yes. Well, at the time, there really wasn’t an accepted thing in it, and I wasn’t aware of it. But I see that this would be 1962, and all…

HAIGH: That’s true. It was three years before the first Ph.D. would graduate in computer science.

STEWART: Yes, so that never would have occurred to me. So I went to Princeton in any case, to study algebraic topology, and then got absolutely fascinated with the foundations of mathematics. My German came to my help and I read through Gödel’s. You know, this kind of stuff. And I actually took a course from Alonzo Church, the logician which was boring—he just read his textbook.

HAIGH: Lambda calculus and the Church-Turing stuff.

STEWART: Well that wasn’t so much that. This was his introduction to logic.

HAIGH: And this wasn’t as interesting.

STEWART: No. That wasn’t what interested me. But it was exciting times, at that time. This was the time where [Paul J] Cohen at Stanford, had proved the independence of the continuum hypothesis and the axiom of choice. They’re two very controversial mathematical propositions from set theory. You don’t to worry what they were. But Kurt Gödel, who did more than just Gödel’s Theorem, had proven that they are what’s called a Relative Consistency theorem, that if set theory was consistent, then either of those were consistent with set theory; you could just add them on as an axiom and you wouldn’t get a contradiction. Gödel had done this, and Cohen came by in 1962, with a proof that you could add the opposite and not get a contradiction. In other words, that they were completely independent of the thing. So this was extremely exciting. There were also philosophical implications, because it was rumored that Gödel already had the independence proof but didn't want to publicize it, because it would lead people to think in the wrong direction. So this was very exciting and there were people there who were interested in it, and we had a seminar for it.
But ultimately, I just didn't take there, and didn't bother to renew my fellowship, and went back to Oak Ridge and sort of to be a happy programmer, basking in the sun.

HAIGH: So your original idea had been that you would do the Ph.D. at Princeton?

STEWART: Oh, yes. Yes, very definitely.

HAIGH: Then when you came back to Oak Ridge, was it to essentially the same job?

STEWART: Essentially the same job. Yes. My position at Oak Ridge has always been the same. As I say, with different scope and this sort of thing. But it’s essentially the same position. So I was doing much the same thing there at Oak Ridge, when I came back.

HAIGH: And how long was it after you came back, before you enrolled in the Ph.D. program?

STEWART: Well, it was a while, because before I did, I went to General Electric. My friend, Bob Edwards, had gone to work for General Electrics in Phoenix, Arizona. He was from Phoenix, and so that was a strong pull. So after a year he recruited me. I got married to my first wife and we went out to General Electric. That was an interesting time, because this was not really a scientific computing group. There were scientific problems and this sort of thing. The most memorable part of it was that both my wife and I were sent up to Alaska to a tracking station for the Nimbus Satellite. Somehow NASA had gotten it into its head that they needed “real” programmers to operate these machines. Presumably if something went wrong, we were going to take care of the problem in real time.

HAIGH: So that was a tracking station?

STEWART: Yes. It was a tracking station. And it was an orbital thing. And so it was right outside of Fairbanks, and that was a wonderful time. We were there in the midsummer when that area is just beautiful. It’s pretty warm, wildflowers all over the place. It’s just fantastic. We left in October when it was getting pretty nasty, except for the beautiful Aurora Borealis.

Then we came back and I did several different kinds of jobs. One very profitable one was programming a process control machine, you know a real-time machine. That’s quite an experience debugging programs with misbehavior you can’t reproduce. So that was useful. But finally, they gave me a Japanese tax form to program up, and I decided that that was not for me, and so I went back to Oak Ridge with my wife. She had not been able to get a job at General Electric, except for this one thing. So she was eager to get back to Oak Ridge.

HAIGH: All right. So was this with the General Electric Computer Division that GE put together decided to enter the computer business as a manufacturer?

STEWART: Yes.

HAIGH: What were your impressions of the general culture and management of the place?

STEWART: Well, it was the typical GE philosophy. You could take a light bulb manufacturer and put it into making computers. And you know, GE had sort of an interoperability philosophy for executives and managers. They believed that you could move, and it didn't really work very well, at least in the computer business. They didn't stay in it very long. The machines were okay, but nothing started like, nothing like what you’d be seeing Cray doing two or three years later, or Control Data. It was sort of unimaginative, you might say. The way of upgrading the machine and creating a higher level was to start turning the clock up. Running diagnostic programs, when
something broke, they would change the transistors around or the boards until it worked. And finally they’d have a higher clock speed, and they’d market this as a new computer.

HAIGH: So there was not much innovation in the architecture itself?

STEWART: No. No. One good thing, of course, that came of it is that before I left, a friend of mine, Bob Funderlic had given me a set of notes for the Michigan Summer Conferences by Jim Wilkinson, who as you no doubt know, is the leading figure in numerical linear algebra. And I had plenty of spare time. I mean this was a customer-oriented thing, and if there were no customers, you had spare time. So I would spend a lot of time…

HAIGH: And were these customers internal GE people who had something that they needed programmed?

STEWART: Some would be internal, but mostly it was external.

HAIGH: So it was running as a kind of service bureau?

STEWART: Yes. You know. So during that time, I would program up the algorithms from Wilkinson’s notes, and this was basically how I got really deeply into the area.

HAIGH: Do you want to say anything more about that experience of programming from Wilkinson’s notes?

STEWART: Oh, well. I mean it was wonderful fun. One of the best ways to learn an algorithm is to program it up yourself. You can sit there and read all kinds of descriptions of algorithms, but when you get down to the fine details, suddenly you really realize what’s going on, and so this was a major learning experience by the time I got back to Oak Ridge. I think I could say that I was very close to being a professional in that area.

HAIGH: So that was really very important in terms of your later career?

STEWART: Yes, very important.

HAIGH: And presumably, you’d done enough hands-on matrix programming previously that you could immediately appreciate the superiority of these matters.

STEWART: Oh, yes. Yes, certainly things like the QR algorithm, which was just very new at the time. That turned out to be fortunate, because it turned out that the QR algorithm was much more flexible than anyone had actually imagined. We’ll come to that story a little later. That was a very formative part of my career.

HAIGH: Then you returned to Oak Ridge, back to the gaseous diffusion plant in 1965-66.

STEWART: Yes.

HAIGH: And it was after you got back there that you enrolled in the Ph.D. program.

STEWART: Yes. What happened was, Bob Funderlic, whom I mentioned earlier, mentioned that Alston Householder was giving a course, and why don’t we go take it? Householder, at that time, was basically at the national lab running what originally had been called the Math Panel, but had become the Computation Center. It really had two arms: the service component and the research component. But he was also a Ford professor and taught a class at the University of Tennessee in Knoxville. It was on Wednesdays and Saturdays, and so I signed up. And I knew the requirements of Tennessee and I did a little calculating and figured that if I took a course in addition to Householder’s, and then cut back to half-time working the second year, I could get
enough knowledge and courses to take the comprehensive exams. Then the plan was that in my third year I’d write my thesis. And it worked out pretty much like that. So I took this course from Alston, and fortunately I was very much helped by the Department Chairman, John Barrett. He died, unfortunately, later of kidney disease, while he was still a productive person. I just simply walked into his office and said, “I want to get my Ph.D.” and he was very helpful. So I just adopted him as my unofficial adviser and would go in and chat occasionally with him. Really, he’s a very wonderful person. So it was almost painless, integrating myself back into the, or integrating myself into the Math Department.

The one difficulty that was always a bit of a nuisance was I never had an office there. At this time, I was commuting to Oak Ridge and coming back, so I had no place to park myself at the University, unlike other teaching assistants. There were no research assistants at this time. You were supported either by teaching, or not supported [chuckle].

HAIGH: And presumably, the attraction of Tennessee would be the familiarity and the fact that you could continue to work…?

STEWART: Exactly. Actually, we were rather well off for graduate students, because my wife was working full time. We had saved a great nest egg from the Alaska sojourn. We were allowed full expenses, with a foreign service allowance tacked on our salaries. And of course with both of us receiving full expenses, two can’t quite live as cheaply as one, but you can come close. And so we had that, and so were able to buy a house and this sort of thing. I was never a starving graduate student, at least as far as Tennessee was concerned. But that was the only disadvantage. I had a good time. The graduate students were a nice lot and friendly, and we all got along fairly well. I would say things just went pretty uneventfully, as far as the formal procedures of graduate school. The comprehensive exams were basically done in two days. They covered five subjects in mathematics. They did it in two six-hour days, I believe it was, or maybe a six-hour and a four-hour. This was followed by a oral exam afterwards, basically to see if your mistakes were really mistakes, or if you knew the subject. It was a very fair way of proceeding, I think, to follow up a written exam with an oral exam to see. The only difficulty was that the topology community who were all young and who I had made some wise-ass comments to, were gunning for me, all in good fun. But it made the oral rather interesting.

HAIGH: So during this period through when you graduated in 1968, was numerical analysis or any other aspect of computing beginning to appear in the program?

STEWART: Well, Alston was the only one. His course was the only one that you could call a numerical analysis course, and it was rather elevated even, that that. Alston was never a hands-on algorithmist or this sort of thing. I mean his contributions were significant because he had this amazing ability to see the general big picture and then set it down clearly. For example, when Alston came to the subject of numerical analysis, an interesting story in itself, there were tons of methods for solving linear equations that went under a wide variety of names, and Alston was the one who perceived that they were all Gaussian elimination, and gave a unified way of treating them so that they all appeared there. But he functioned at that level, even though he got two algorithms that have survived. But he was basically a higher-level person. He was not a very good lecturer. He mumbled and was not always completely prepared. He had a notorious lecture called his “B-D” lecture, in which he had two sets of vectors that he denoted by “B” as “D” so you’ve already got mispronunciations together. And basically, his “B’s” on the blackboard
looked like his “D’s” on the blackboard [chuckles], and so it went. But the material was inspiring and I really enjoyed those classes that I took from him.

HAIGH: So did he just teach one course in this?

STEWART: Well, he taught two different courses, but he only taught one at a time.

HAIGH: Right.

STEWART: One was on numerical linear algebra and the other was on a book that he was writing on, The Solution of Non-Linear Equations, and I took both courses. [Householder, A. S., (1970). The numerical treatment of a single non-linear equation. New York: McGraw-Hill].

HAIGH: As you were inspired to go back to the university after you took his course, was your idea always that he would be your advisor?

STEWART: I suppose, sort of. I really wasn’t thinking that far when I went back. You know I was just going to, I knew that I could get through the comprehensives, and I’d see what came of it. But by the time I was into his courses it was fairly clear. Also, by the time I had done this, I had attended the Michigan Summer Conferences, and that was the experience that confirmed everything, that brought together the two aspects, the numerical analysis and the mathematics.

HAIGH: I’ll ask you some more about that in a second. So how did your relationship with Householder develop after that first course?

STEWART: Well, mainly because of myself. I could be rather diffident. It’s not easy for me to go up and be a “hail fellow” while in a new circumstance. I’m quite relaxed with my friends and this sort of thing, but in a new situation…. So here was a guy who was very formidable, who had an international reputation—he was the only one at Tennessee that had an international reputation. So I was a little scared of him, and didn't really have much chance to work with him or talk with him. I just took the courses and worked with him. Actually a little later on, during one of the courses, I saw a much better way of proving one of the things that he was doing, and we ended up writing a joint paper on that that went into the proceedings of a conference. But before I started working on my thesis, my contact with him was really rather infrequent, of the classroom situation.

HAIGH: Oh, here we are. [(with A. S. Householder), Bigradients, Hankel Determinants and the Pade Table,” in Constructive Aspects of the Fundamental Theory of Algebra, Dejon and Henrici eds., John Wiley & Sons, New York, 1969]. Then did that relationship change over the years?

STEWART: Oh, yes. I mean we became very good friends. Probably while I was actually writing my thesis. Partly I wanted to keep distant because I was really, I wanted my dissertation to be my own work completely and this sort of thing. And of course, I was very naive about what relation an advisor has to a student. I really didn’t know what it was, so after I passed my comprehensive exams, Alston and I met. I guess it was in May or something like this, and this was out at Oak Ridge there, and we discussed some possible topics for a thesis. So after we had finished talking, I said, “Okay. Well, I’ll see you at the end of the summer.” And he looked at me, a little woebegone, and said, “It is customary to meet with your advisor a little more frequently than that.” So we agreed to meet once a month, on that. Of course, by that time, I was also publishing my own papers. As a matter of fact, the first paper I published was on the Davidon method was done at Oak Ridge before I had any really serious relationship with Alston. [“A Modification of Davidion’s Minimization Method to Accept Difference Approximations to
Derivatives,” *Journal of the ACM* (1967) 72-83]. But afterwards, you know, after that strain of student-professor relation, we became very good friends.

HAIGH: Well, I’ll be asking you later about the Gatlinburg meetings, so that might be a chance to pick up on that side of the story.

STEWART: Yes. Yes, that would be fine.

HAIGH: I will ask, though, while we’re on this period, do you think that there was anything about your own approach to research that you can identify that might have been influenced by your early experience working with Householder?

STEWART: Well, this business of trying to see the big picture had definitely been with me. I’ve always tried, throughout my career, to take time out to step back and see and write down what’s going on. So I enjoy writing survey papers and books that survey subjects, and this sort of thing. And I would say very much, that in doing this I’m Householder’s student.

HAIGH: All right. In order to keep this in chronological order, I’ll ask you about that first paper and about Michigan Summer School, and then we’ll come back and talk about your thesis. So let’s do the summer school first.

STEWART: Okay. That’s a good place to start.

HAIGH: So was it prior to attending the summer school, you’d already seen Wilkinson’s notes?

STEWART: Yes. Yes, so I was very primed for seeing Wilkinson himself. And he’s very much like his notes [chuckle], or was.

HAIGH: Okay. So what do you mean by that?

STEWART: Well, I mean that he was a voluble, very, I don’t want to say “smooth” because that implies slick, and there’s nothing slick about Wilkinson, but very articulate and enthusiastic, bouncy almost, in a way. And this showed through in his notes. It shows through in his writing. I mean you have to be attuned to it, but it did. So in any case, yes. This primed me for at least Wilkinson’s lectures.

At this time, it was sort of at the height. The early numerical analysts suffered from a lack of any venues to where they could meet and get together. When you get to the Gatlinburg conferences, I’m sure you’ve talked with enough people about that to know that was one of the venues. But the Michigan Summer Conferences were another, especially for the numerical linear algebra people. These people would come, drink large quantities of beer at the old German restaurant at Ann Arbor, and generally exchange ideas and things. It’s said that a theorem on things called Gershgorin discs was inspired by the rings made by the beer mugs on the table at the old German restaurant. So for them, it was an opportunity to get together in pleasant circumstances. For a person like me, when you have people like Jim Wilkinson, Alston Householder, John Todd, Richard Varga—you know, these people were minor gods to me and to a lot of other people.

HAIGH: Was this before you took Householder’s course or afterwards?

STEWART: It was while I was taking courses from him, and so I knew Alston at that time, too. But I was just glued to my seat. Anyone will tell you that that’s a very unusual thing to happen [chuckles]. And as I say, it’s at this point that I really knew that this was what I wanted to do with my life.
HAIGH: So did you feel that you were accepted by these people as part of that community?

STEWART: We didn't have any contact with them to speak of. The students were students, and they were there. So I had nothing there. I would have liked to have talked to Jim Wilkinson, and introduced him, but he was mobbed by everyone at the end. As I said, I can be rather diffident in those circumstances. So it was only after I got to the University of Texas that we first met.

HAIGH: So how large were the classes?

STEWART: Oh. I would guess, let’s see. The tables had seven or eight-- maybe 48, six by eight, something like that. Fifty. Let’s go to a round number and then put plus or minus ten.

HAIGH: So reasonably large, then.

STEWART: Yes. Oh, yes. Union Carbide used them as a perk you’d send these people to the conference. They didn't really care if you did anything after. There were a lot of other people in there in essentially that situation. It was a perk. They could go. And they all dutifully stayed there, but I think they were having more fun going out to eat afterwards.

HAIGH: Were there any other people that you did get to know more personally during the school?

STEWART: Well, actually yes. Bert Rust, who I mentioned earlier. We actually both went at the same time, so we drove up together and shared a room while we were there. So this was a nice way to become closer friends. But we had known each other for a long time. But the actual contact between the students and the faculty, or whatever you want to call them at the Michigan Summer Conference, was very little. But that didn't matter to me. It was the material that was really…

HAIGH: Did you have any impressions of Todd or Varga?

STEWART: Well, I admired Dick Varga’s elegant lectures in blackboard technique. Of course, I actually on my own studied his book, *Matrix Iterative Analysis*. So again, I was primed for this material. It was like ramming it in and seeing what it was like to be doing it yourself. That perhaps is maybe the amazing thing, an almost physical sensation of these people that they really were doing the research—that I wasn’t reading it on a page, mathematics or this sort of stuff. I realized that they were having great fun at it. That maybe as much as anything was there.

John Todd was probably the least of the lecturers. A wonderful man, but his lectures were not especially inspiring. There was Donald Greenspan who lectured on partial differential equations. He was quite a character, a big bear like person who would write on the blackboard with great vigor and break the chalk and then stomp around and grind the chalk into the podium or into the floor. It was very interesting. But never having really warmed up to PDEs, he didn’t have much influence on me either. They were wonderful conferences. I mean, people who had been there at other times said the same thing.

HAIGH: And did you just attend it that one summer?

STEWART: Yes. There wouldn’t have been much point in attending it more.

HAIGH: So it’s the same similar material every year?

STEWART: Yes. Oh, yes. It was the same material recycled or changed somewhat.
HAIGH: Anything else you want to say about any influence the Michigan summer conferences had any affect on your later career?

STEWART: Well, as I say, the technical material and the idea that this was a great, great area to do research in.

HAIGH: Do you want to talk then about that first paper on Davidion’s minimization methods?

STEWART: Yes. Let me digress a little and give some background here. Optimization at that time meant one of two things: linear programming or the solution of unconstrained minimization, you know, nonlinear minimization by various things. And the techniques were quite rudimentary at that time. One of the people working at it very hard was a major person in the field, Mike Powell, who is now a member of the Royal Society at Oxford. He may have retired; I’m not sure. So in 1962, I believe, a physicist at Argonne National Laboratory named Davidion produced what he called a variable metric method for optimization. I won’t try to explain it, but it basically was a form of Newton’s multivariate method that didn’t require the kinds of derivatives and work that Newton’s method did. It was the key insight. It developed in many, many different directions. Davidion would probably not recognize how it’s developed, but it was the key insight. And Powell also jumped on this. This was really, I would say, beginning of that area of optimization, Davidion’s paper. Well, of course, I got interested in it. I don’t know where I saw it, but I had an idea for even reducing the number of derivatives that you had to compute. I think I started this work, and most of it General Electric; again, a spare-time project. But when I came back to Oak Ridge I actually had written this program for doing this.

Now, at the time, the way people proved that they’ve gotten advanced in optimization was that they would write a program, and then they would assemble all the test cases that everyone else had used beforehand, add a couple of their own, and say, “Gee, look how much better I’m doing.” It’s not a very good way for a field to progress. Much of it was based on how well you could fine-tune your program. So I was very good at fine-tuning my program on an ad hoc basis, not on principles. So I beat out the competition until a year later or something. So I wrote up a paper on it, overlong and not very well written. I sent it to the Journal of the ACM, and it was accepted. The referees report was very good, and, of course, I could not possibly see how he could say these nasty things about my paper, but I followed all the suggestions, and they published it; it came out in 1965, I believe. As a matter of fact, I got the galley proofs when I was at Michigan summer conferences. I had gotten them just before.

HAIGH: Actually, it shows it was in 1967 on your resume that it came out.

STEWART: Oh, okay. Maybe it came out then. Maybe it was published then. I believe you. I believe that’s right. I can’t remember the details of that. Yes, it was a long time going through the process. But in any case, that was my first paper. It was, in many ways, not a paper to be proud of, but it was a paper, and I had done it completely without advice. As a matter of fact, that’s where I should have gotten advice on how to write things up better.

HAIGH: When that finally arrived in print in your mailbox, was that an important moment for you?

STEWART: Yes. Although actually, the second paper I wrote on the generalized inverse was the one that I really loved. That comes a little later. Although we could discuss it now.

HAIGH: Let’s wrap up with your thesis first. So that paper came out. Now, at this point, when you enrolled in the PhD program, sent your first paper up for review, did you have the idea then
that you wanted to be a career academic researcher, or did you think this might coexist with continuing to work as a programmer?

STEWART: Well, actually I always wanted to be in academics. I mean, this was something-- The University of Tennessee sat on a hill. At the time that I was there as an undergraduate, and even as a graduate, most of the departments were concentrated up there: the education department, and the business department. And this is a fairly formidable hill. I would walk up and watch the faculty members zip by in their cars to park on the top of the hill. And I thought that this was just wonderful, that I would someday be a faculty member myself and be able to zip up through the thing. It’s a little like the mathematician [G.H.] Hardy when he was a young person wanting to go into an academic career because he could drink port and eat walnuts in the commons room! But no, I had always wanted to. Although, I had never taught, I always felt that I wanted to. Actually, most people at the university thought when I graduated that I was going to stay at Oak Ridge. They were very surprised when I went for an academic position. They thought I would just go back. By that time, I was with the laboratory, and I could have had a very nice job just doing research, a research position.

HAIGH: Do you want to say something about the fellowship that you got for 1967 and your experiences at the laboratory?

STEWART: Yes. The experiences weren’t much. This was after I passed my comprehensive exams. I applied for a dissertation year fellowship, and got it with no problem. I’m sure it was Alston that I made sure that I got it. He would have certainly had a great deal of weight in deciding it, which was very nice, because I had planned, of course, to drop out completely and work on my dissertation for the last year, so this provides some income in addition to wife’s. And so I had an office out at Oak Ridge. I still had to take one more course to finish up at the university. Actually, the Orel people gave me some trouble with that, because I was supposed to be working on my dissertation and not taking classes. But that was smoothed over, and I went back, which was very nice, because I really liked the people back at Tennessee, the graduate students there. They weren’t friends for life, but they were friends. It was always nice to get back there and chat with them.

At the laboratory itself, I was working on my thesis, I did have a number of friends. Amongst them, was Bill Gragg. He’s a person who has done some really fundamental work in the subject of what is called fast numerical algorithms. He really is a very interesting person. This November, they’re having a celebration for his 70th birthday. But he was associated with the university at Oak Ridge. There were other people who worked there. It was just a very pleasant environment with the people there.

HAIGH: So how would you describe the general culture at the lab at that point?

STEWART: That’s hard to say, because I never really was deeply integrated into it.

HAIGH: What was the name of the piece that you were in?

STEWART: Well, even that’s hard to say. I mean, it was in Alston’s division or whatever the organization was called. At that time, what had been the math panel, as I mentioned earlier, had grown quite enormously. They finally got computers other than home grown ones. It was a Control Data computer that they got, and so Alston’s research oriented thing acquired a big service component there. Alston, at that time, was approaching retirement. Basically he did the research end of the thing, and another person managed all of the service part, and I had nothing
to do whatsoever with that other than I used their computers. So as far as the other part was concerned, I would say it was not an especially active place. As I say, Alston was approaching retirement and definitely slowing down in some sense, so the main activity seemed to come mostly from Alston himself, in the sense that he was still an active researcher, and from the people who came through, people like Fritz Bauer and, although not at my time, Jim Wilkinson. And during the time that I was there, Gene Golub came, and so I got to talk with them.

[Tape 2, Side A]

HAIGH: You were describing the lab, Alston slowing down, and then you mentioned the series of visitors: Gene Golub, Bauer.

STEWART: Yes. So that certainly was stimulating, but the institution itself, I would say, was definitely in decline, whereas I think Argonne was coming into an ascendancy at that time. This was the time when they were contemplating EISPACK and things like that. So I think there was a changing of the guard here going on. But as I say, mostly, except for talking with people who I had made friends with and this sort of thing, I was doing my own research there.

HAIGH: You mentioned Gene Golub. Was that the first time that you met him?

STEWART: Yes. He came through. I get very stimulated by these conversations, so this leads in naturally into my thesis. After talking with him I went home in such a furor that I got the major ideas for the biggest part of my thesis, basically overnight or so. And, of course, there was a lot of work to do there. Mostly because of the stimulation, there was nothing that he said or that we talked about that had anything to do with my thesis.

So if you want to talk about my thesis, let’s back up to where Alston said it’s customary to meet with your advisor a little bit more often. The one thing that we had agreed on was that I would do some work on Lehmer’s method, which was a method for finding zeros with polynomials. I won’t bore you with the technical details, but it was a method of some promise, and I was going to work on that. And I did, and got, actually, a good algorithm based on it that basically combined Lehmer’s method and Newton’s method. It wasn’t the algorithm that people ended up using. It was too expensive and this sort of thing. But it was a nice journeyman’s piece of work. And that was part of my thesis.

The other part was going to be this paper on the perturbation of the generalized inverse. This isn’t really a good time to discuss it, I think, because it was done at Oak Ridge. Actually, it was done partly at the Gaseous Diffusion Plant and partly at the laboratory. So this paper was going to be part of my thesis. It arose when I decided to try to find out a perturbation analysis of least squares. This has been done by Gene Golub and Jim Wilkinson, but they hadn’t done it rigorously in the sense of actually producing bounds. They had produced order estimates, as they’re called, so I decided that just for fun I would try to produce the rigorous bounds. And I did, and I wrote it up. I think it’s a pretty nice piece of work.

I gave it to Alston after I’d written it up, and he sent it out for review for SIAM Journal on Numerical Analysis. And I got back a report from a person named Adi-Ben Israel, who was the doyen of a thing called generalized inverses, which are closely related to the topic, saying he had done it all in his paper. In point of fact, he hadn’t done it all in his paper because he had left out the really hard cases that were difficult to choose. And so I said, “Well, look. I’ll not let this mistake happen again.” And so I changed it from least squares to generalized inverses and made very clear what had not been done. And then the paper went through, and I had intended to
include that in my dissertation. But after I developed the stuff that I’ll talk about in a moment, Alston suggested that I drop that and just publish it as a paper, which is how it came out.


STEWART: Is that where it was published? Okay. Yes. Yes, there is quite a gap, incidentally, in time between when these things are submitted and when they’re published at that time. It’s not like physics or other areas. So you can count on a year, year and a half in the appearance of an article.

So in any case, I had this Lehmer’s Method and I had this paper, but I really felt that I didn’t have a thesis. It was I guess probably in October some time, and I said, “You know, if I’m going to get an academic position next year I better get a thesis.” And so Gene visited. Oh, at the time, I had been primed by reading some of Fritz Bauer work on ‘treppeniteration’, which is a method sometimes now called simultaneous iteration or subspace iteration for finding groups of eigenvalues and is a generalization of the Power Method. In fact, I had actually translated some of his papers from German on that.

HAIGH: Just for your own use?

STEWART: Just for fun! I’ve always enjoyed translating. So I got to thinking about this, and one of the problems is slow convergence when they’re close eigenvalues. It suddenly occurred to me how you could speed it up. The technique is now known as the Rayleigh-Ritz Method, but I didn’t have enough background to know that at the time. And so I immediately sat down and programmed the method to see if it worked, and it worked beautifully, so I decided that I would develop this new algorithm and analyze it. And it went remarkably fast—I think a month, basically, or so to do it. And my main interest was in algorithm. But in the course of it, I needed some perturbation theory for what are called invariant subspaces, in order to prove that the algorithm worked. And so not knowing that this was a hard problem, I tackled it…or I found a paper whose approach for doing it were just simply eigenvectors, which are one-dimensional variants of spaces. It could be generalized in a very neat way, and so I just did it and wrote it up. And it appeared as theorems within my thesis.

And so I had this new algorithm, and I had this perturbation theory. Alston, actually, when he was writing back after I had turned it in… Oh, by the time that I had done that, I had decided that the smart thing to do was to write this up as if it were a paper rather than as a thesis and that I could just get it typed up and send it off to a journal. So I got back. And he said something that mystified me about these theorems. He said, “Well, you’re very close to Kahan.” And I had no idea what this meant. This is Vel Kahan. And again, I was really a little too shy to ask. I commented, “Well, I just want to prove that this program works.” And there it was.

So it turns out (and this was completely without my knowledge) that I really basically hit right on the nose the research of two other people. Well, first, there were Davis and Kahan who were working on perturbation theory for invariant subspaces. Now, what I did was in essence the stuff that they did, but they did much more. I don’t want to claim that I came out with this glorious theory or everything, but the basic idea was there, and it was used effectively in an application or in the mathematics of analyzing a program. But the other thing that I didn’t know was that Heinz Rutishauser, who is considered one of the best algorithmists in numerical analysis, he was at, I
believe, the ETH, the Technical High School at Zurich. He had come up with the same algorithm for speeding things up, this sort of thing, and he had his own analysis of the thing. And as it turned out, these appeared back to back in the journal *Numerische Mathematik*—our two papers.

So this became the major part of my thesis. And basically, it was a blockbuster, but I never realized it until 20 years later. [Chuckles] I just never put two and two together. I knew that I seemed to be progressing professionally very well, but it was only later when I began to think back on it that I realized suddenly that I had, as a graduate student and largely without help (I mean, Alston had very little to do with this part), had come up and basically reproduced the research of some of the bigger people in the field.

HAIGH: Okay. So which of these papers is that you’re talking about now.

STEWART: This: “Accelerating the Orthogonal.”


STEWART: Yes. So in any case, with that, my thesis was written. And I wrote up Lehmer’s method and that, so it was called, “Some Topics in Numerical Analysis.” But two would be more exact. And I passed my comprehensives and started looking for a job and ended up at the University of Texas.

HAIGH: All right. So just while we’re talking about those topics, you talked about the publications that came directly from the thesis.

STEWART: Well, there would be two. Actually, there would be three. Okay, one would have been on Lehmer’s Method for the solution of polynomial equations or something like that. I believed that appeared in *Mathematics of Computation*.


STEWART: Yes. And then there was #22, which I wrote up while I was at Texas and sent in. It was a technical part of the Lehmer thesis. [On a Companion Operator for Analytic Functions,” *Numerische Mathematik* 18 (1971)].

Actually, this illustrates the problem that I’ve sometimes had. I’ve worked very hard to make my stuff easy to see. I really don’t believe in trying to impress people with pages of equations, so if ever I can find a short way, I’ll rewrite and this sort of thing. And occasionally, I’ve gotten back referee’s reports that say, “Oh, this is obvious!” You know, it wasn’t obvious. It wasn’t, to anyone or to myself. And this was a case in point, except the referee knew that it wasn’t obvious. I suspect it was Jim Wilkinson. But it said, “These are interesting results and deserve to be published. They are not as obvious as Stewart makes them appear.” So that was essentially the referee’s report. There were no suggestions for any changes or anything. So those were the two papers. And then, the Orthogonal Iteration paper. So those three appeared from my thesis. And the latter was, as they say, written up in the form of a paper already. It’s in my thesis, and I just shipped it off.

HAIGH: So beyond those papers that came directly from the thesis, did the experience of working on these topics lead you later on to return to any of the same questions or have an influence on how you approach things?
STEWART: Both in a good way and a bad way, but maybe this is a good place to stop and come back to Texas because that’s where that happened.

HAIGH: I’ll flag that question for later then. Just one more question on the thesis itself then. You had mentioned that you were, you know, it was tied in with implementing test versions, right, that there was some programming involved with it. Was that taking place at the Oak Ridge Laboratory doing the fellowship?

STEWART: Yes. And I might add that on Lehmer’s method I actually produced a formal program that appeared in the microfiche pocket of Mathematics of Computation along with the article. That may have been the first program to appear in that way in Mathematics of Computation. Usually that was reserved for tables and things like that. But I said, “Can you put the program in there?” Unfortunately it was written in PL/1.

HAIGH: Did you include any programs in the thesis as an appendix?

STEWART: No.

HAIGH: So in those days— I guess the bigger question here is how novel was this? Was an implementation anything that you could get academic credit for at that point?

STEWART: It would’ve been difficult. This was a time when these very ideas, things were being debated. Remember that still, at this time, a large part of the numerical analysis and algorithmic development, at least numericals, was being done in math departments or in engineering departments often. For example, the history of the finite element method: it was an engineer’s creation. It only later got taken over by the numerical analysts. So engineers have made substantial property contributions. But I think in a math department just producing a new program, even if it had great originality, would not have been sufficient to get a Ph.D.

Begin Session 2, taking place in the afternoon after lunch.

HAIGH: So before we took the break, we talked through your work at Oak Ridge Plant, your fellowship at the Oak Ridge National Laboratory, your thesis work. So it would seem the logical thing to discuss next would be your future after graduation. As you were finishing up the thesis, did you have a clear plan of what you wanted to do next?

STEWART: Yes, very clear in the sense that I wanted to get a university position. I had only the vaguest idea of what that was. I was extremely naïve about it, but when I let Alston know he pulled out the stops and got me interviews. He actually talked to George Forsyth about a possibility, but at that point they just had nothing at Stanford in their department. So I ended up interviewing at the University of Texas, the University of Virginia, and Florida State University where actually Orville Harold, who I had mentioned earlier, was now chairman of the department and brought me in. But it was very clear from the very first that Texas was the place. Texas was extremely attractive. I was offered a joint appointment in the math and the computer science department. The computer science department was just being started up, and they had done it quite in the right way. The same at Maryland, incidentally, when they started up. Rather than just trying to hire a few people, what they got was very senior researchers who were at least on the periphery of computer science, people like Jim Brown, who was a physicist, but had become interested in computer networks and such things as that. He actually was the chairman when I arrived, although I was hired by Bob Gregory, who was the first chairman of the department. So they got these senior people who knew what research was and then they brought junior people in and made sure that they learned how to do it, but in the more core areas. This is
an excellent way to build up a department. And the math department also had been given a large amount of money to expand. Although I didn’t know this when I accepted the job, I was moving into a milieu with a lot of people just my age, new Ph.D.s and this sort of thing. This is a nice thing to have to have companionship as you go along.

HAIGH: You arrived there in 1968 as assistant professor; that was a regular tenure track position?

STEWART: Yes, regular tenure track position.

HAIGH: Now on your resume this shows as “mathematics and computer science.” So at that point was that one department or was that a joint appointment in two departments?

STEWART: No, it was a joint appointment in two departments. It’s not a position I would generally advise someone to take, although it worked out quite well enough for me. So I came there. We bought a house within reasonable distance, commuting distance from the university. As a matter of fact I could ride a bike there, but the temperature was not very conducive to doing that and started in. This was the first teaching that I had done, and I was teaching two courses a semester, something I swore I would never do again. I’ve managed to not do that. But it was an enormous load, you know, having to learn how to teach, how to prepare classes, spinning your wheels doing unnecessary things. I was learning as I went. So it was extremely busy, extremely timely, and at the same time trying to do a full program of research in this time. It was an exhausting year. I think the only reason I got through it was that there was a program called Laugh In every Monday night that I would watch on the television. It was quite the thing at the time, and that provided me with at least an hour of humor a week.

But otherwise it was very nice. We made friends there. The department was very understanding. They actually gave me for a while a little office in the math department (they couldn’t, later space stopped them) but my main office was in the computer science department. At this time, David Young took me under his wing. He ran the Center for Numerical Analysis there. I had said that the two aspects of my thesis, one was unfortunate and one was fortunate. The unfortunate one was that I tried to continue my work into the solution of polynomial equations, and this was sort of a dead end. I mean, there really wasn’t much research in this sort of thing, to do with that. Why look at it any more? So it really was a dead end. No one has done much with it. As a matter of fact, you can almost recognize a crank paper when it says, “A Method for Solving Polynomial Equations”—they’re not really going to improve much on what’s in there. So I did some work in that, more on the theoretical side than on the algorithmic side, and tried to get an NSF grant on that subject and it failed completely; I didn’t get anything. So we had a sort of lean summer in getting through.

Then a rather puzzling thing. Again, I was so naive, I didn't really recognize the significance of it, but Bob Simmons, who was in artificial intelligence, came by my office and said (this was at the end of my first year), “I hope you won’t hold it against us, but we just can’t find it possible to offer you tenure after only one year.”

HAIGH: The tenure was less formalized in those days?
STEWART: Well, you could always offer…. I mean, in all universities, it’s gotten very much bureaucratized now. But as far as offering early tenure, or whatever, that would be a department’s decision. They would just have to force it up through the thing. They did offer me tenure the second year, and I got it with no difficulty. As a matter of fact, one of the professors whose name I won’t say, but a bit of a gossip, followed it all the way up and would give me weekly reports as to what was happening.

HAIGH: So that related to what you said earlier, about not relating until later, how soon you could work on your thesis?

STEWART: Yes, exactly. I’m sure this had to do with my thesis work, plus the fact that I was being very productive even during that first year. The second year was much easier. David Young gave me a half-time position in the Center for Numerical Analysis, so my teaching load was cut in half. I mean, I really enjoyed teaching. It’s not that I dislike that. I very much liked teaching, and any time that I don’t teach for a semester or two, I get very antsy to get back in front of the class. So it wasn’t that, but everything in moderation. So there was that. I had support. Then I got grant support, actually. I realized then that I needed to concentrate on my strengths, and someone told me, and very wise advice; “Well, call up these people at the NSF and talk to them.” And of course, as any assistant professor, these people are seven feet tall and have fangs, dripping blood and everything else, and I was totally scared to call these people up. Finally, I locked myself in my office, and the resolve that I couldn’t go out— or I didn’t lock myself; I closed the doors, such that I couldn’t go out until I made these phone calls. Then I began drinking coffee, and pretty soon hydraulic pressure forced me to make the calls. I found out, of course, that these people are really nice, concerned about their programs and eager to help and everything else. So I then was, from there on, reasonably successful at getting support.

HAIGH: How much of an expectation was there, in those days, that someone researching applied mathematics would be able to find external funding?

STEWART: Oh, very much. Things like numerical analysis, and this sort of thing were well supported by NSF and this sort of thing. They, for example, funded a lot of the earlier Householder meetings. So you could certainly expect support. Also for a long time I had support from the Office of Naval Research. They had a long tradition of supporting all kinds of things.

HAIGH: And that’s back to being in the very early days of the computer?

STEWART: Yes, and even outside computing. They supported people who invented plate tectonics, you know, all the trips across the ocean to map the ocean. They had geologists along.

HAIGH: So you’ve mentioned the Center for Numerical Analysis a couple of times. What was that?

STEWART: Well, it was David Young’s creation. Dave was, I’m sure you’ve heard him mentioned before, but wrote a very fundamental thesis at Harvard, and then came by degrees, but I think fairly soon after his thesis, to the University of Texas where he actually became head of the Computation Center.

HAIGH: And that was the campus computer center, was it?

STEWART: Yes, yes. It was right outside the building that we were currently in, in sort of a bunker-like arrangement, just below the big Tower in Texas. And so he came. And eventually it became clear that running the Computation Center really wasn’t his métier. So I don’t know
whether they eased him out or just negotiated it, but he was given money for a Center for Numerical Analysis, for secretarial support and money to support people and this sort of thing. That was split off from the Computation Center, and that’s where it came from. So he supported me and some others.

HAIGH: I saw it referenced in your resume that you also had a role as a research mathematician in the computer center, so that was in fact in the Center for Numerical Analysis?

STEWART: Yes, that would be the Center for Numerical Analysis. I’m not exactly sure as to where the organization chart lines go, but it was essentially a separate group.

HAIGH: Did you know David Young before you arrived at Texas.

STEWART: No. No, I had never met him. I met him first when I was there. As a matter of fact, I was a little surprised that I was hired at Texas, because when I’m doing research it’s usually very incoherent until it’s at an advanced stage. And I tried to talk to him about some of my ideas that I had, and basically it was just unintelligible, or at least, I thought I was.

HAIGH: So you didn't have a good elevator summary of what your work was all about.

STEWART: Well, I was just talking about what I had been thinking of doing. I’ve never really had a research program. I’d just go on from interesting problem to interesting problem, and don’t usually look back very much. I don’t try to redo the thing.

But maybe we should return to the other thing that was good, and that was the algorithm for orthogonal iterations and the perturbation analysis that went with that.

HAIGH: All right. Let’s just talk for a second about the institutional side of things. So you had mentioned that the mathematics department and the newly formed computer science department brought forth interesting young faculty people. Is there anything more that you want to say about the culture there and the work that that was going on with your colleagues?

STEWART: Oh, yes. Well, it was, in the math department, of course, the math department was the one that the great big hiring, and I made a number of good friends like Jim Daniel, and Larry Schumaker. These are not names that you would know because they were not in a computational area as such. But it was just a nice bunch of young people and there was a lot of socialization and parties and this sort of thing. It was very nice situation to be in. In the computer science department, there were no others, but I was good friends with Richard Bartels, who is now at Waterloo. We socialized with them a lot. The research in both departments was excellent, especially in the math department. There were just a lot of young people who had gone on to make full professor and this sort of thing, so it was just a very nice place to be.

HAIGH: In computer science, were they offering undergraduate degrees?

STEWART: Yes, undergraduate and graduate degrees. As a matter of fact, when I came, although I had had a lot of computing experience, I had not had any formal training in some of the areas like programming languages and this sort of thing, and so I decided that to be a good citizen of the department that I ought to really learn some of this stuff. So I started systematically working through the advanced/junior/senior level undergraduate courses in programming languages, and the like. And to practice, I continued up to now, and just taught some. One of the best ways to learn something is to teach it, and this was a more or less deliberate program to broaden myself in computer science.
HAIGH: What was the position of the numerical analysis in the curriculum at that point, in computer science?

STEWART: Oh. There were standard undergraduate and graduate courses in them. I can’t remember exactly what the names of them were, but they were well attended. Numerical analysis was still very important in the computer sciences at that point. A large number of people who had come to computer sciences were also numerical analysts, so it was very well respected.

HAIGH: When you were teaching the numerical analysis courses, would you get students coming from Engineering and other disciplines?

STEWART: Oh, yes. Yes, definitely. Nowadays, a lot of engineering departments teach sort of their own flavor of numerical analysis: numerical analysis and civil engineering, or numerical analysis and chemical engineering. And so you see much less of that nowadays, or at least at Maryland. I haven’t done any surveys to see what’s going on in other schools, but I have no reason to believe that it’s different. But at this time, and even up to the late 1970s and later, when I was at Maryland, we still got a lot of engineers. And we still do get some in our courses, but not the number that we got previously.

HAIGH: So that covers the questions I had about the institutional side of Texas. So unless you have anything else to say about your colleagues or the program there or the university, then we can move to discuss the research that you did over there. I think I cut you off when you were beginning to talk about perturbation.

STEWART: Well, I’ve always thought that it-- or from my experience with polynomials, I’ve come to the conclusion that one should not try to rewrite one’s dissertation, that it’s not necessarily good. It’s good to branch out, but in this case, in the orthogonal iteration and the perturbation theory that it went on to, were done for symmetric matrixes. Symmetric matrixes have the property that they’re much easier to handle than non-symmetric matrixes. But it turns out that both of the perturbation theory and the algorithm had very nice generalizations to non-symmetric matrixes. And I pursued that in the thing. In this case, although I knew pretty much the kind of algorithm that I wanted for the non-symmetric case, I actually worked on the perturbation theory first, because I knew that I needed to understand the algorithm. And this resulted in something of a breakthrough in itself, because it generalized in a very nice way, the stuff that Davis and Kahan had done and that I had sort of, included in my thesis there. So that was a paper that appeared probably about in ’71 or so. There. It’s 1971. That was the original paper.


STEWART: Yes, a nice unassuming type of paper. So the basic idea of the thing is much simpler than matrixes, and I decided for fun-- and I was very deficient in functional analysis-- to do these things in Hilbert spaces, just to see what I could learn about it. So I wrote this paper, but it’s very hard to read because of all the paraphernalia that you have to put in it. And some time later, I discovered that this approach generalized other kinds of eigenvalue problems, and when I realized that, I realized that I should write a general paper that surveyed the whole thing, and I kept it in terms of matrixes and that’s the “Error in Perturbation Bounds for Subspaces Associated with Certain Eigenvalue Problems,” that appeared in SIAM Review. That was actually
my *SIAM Review* paper. I’ve written several since, all in perturbation theory; I’ve always seemed to put that in.

HAIGH: I talked to Ed Block about the history of SIAM publications, and his idea was that *SIAM Review* would hold material that was interesting, that talked to the broadest kind of audience, and so would be of interest to people who weren’t just specialists in one corner.

STEWART: Yes. Well, it never really worked out that way. I think the new *SIAM Review* is a little more oriented to that way, but basically the papers were intended to be survey papers, but not really accessible even to very technically competent people who weren’t in the immediate area, or in a closely related area. All you have to do is just go back and look at them and ask to take the paper out of your area, and then ask, “Can I read this without any difficulty?” and the answer is pretty much, “No.” Beyond a certain point in these things, there’s just too much you have to learn before you can appreciate the niceties or even follow the mathematical language of these papers. People talk about mathematics as being a universal language, but it’s more like Chinese characters. It’s a common way of writing a whole bunch of different languages or dialects. Physicists have their own version of mathematics; statisticians, another; and electrical engineers single processors, have another. And it’s not easy to break into one that you have not grown up with.

HAIGH: What kind of reception did you get for the 1973 paper?

STEWART: Oh, very favorable. I mean, the results were new. It took time to percolate through, because they’re not easy to use in some sense. And people were not thinking very much in those days, in terms of invariant subspaces rather than eigenvalues. The reason to think about invariant subspaces is, if you have a matrix, say, with two eigenvalues, and they have things called eigenvectors, which are really the things of interest, what happens is as eigenvalues approach each other, these eigenvectors become more and more sensitive to the perturbations in the matrix, and they sort of spin around. But if two eigenvalues that are approaching each other are separated from the other eigenvalues, then when you look at the thing you find that those two vectors are spinning all over the place, but they’re spinning in a very well-defined plane; they’re not moving out of that plane. That plane is the thing that you want to put the perturbation bound on, not on the eigenvectors, which are hopeless, but on where they’re lying. So that’s the fundamental contribution. I mean that was in the Davis-Kahan paper, but it was for symmetric matrixes, and it’s more difficult to do it for non-symmetric matrixes.

HAIGH: Did the results from this paper translate into mathematical software in terms of improving the accuracy of the performance?

STEWART: Well, when I went on to look at the algorithmic aspect of the generalization, let’s call it subspace iteration of non-symmetric matrixes—as I mentioned before, it has several names. I had really developed this perturbation theory because I didn’t feel that I could write the algorithm without understanding-- and the algorithm itself is very difficult. To get it right, it’s not at all simple, and to prove that it converges is not at all right. So I needed that perturbation theory, even to write the algorithm itself. But once I had it, it was done, and then I wrote programs for it. Eventually many years later, I had actually published it in *Transactions of Mathematical Software*, or a version of it.

HAIGH: But in that case, once you had the theory, there was still a lot of work to be done on the algorithms.
STEWART: Oh, yes. Yes. I mean the point of it is that the theory was general. The application to
the algorithm was very specific, and that required further analysis.

HAIGH: Do you want to say anything about ways the theory might have been used by other
people in different areas?

STEWART: Well it’s my understanding from Margaret Wright that it’s been widely used, for
example, in optimization. Now when these interior point methods came in, I believe this is what
she said. It’s you know, been very informal, but my book with [J.G.] Sun, which recapitulates a
lot of this, was found to be very useful in handling their algorithms. So it’s useful.

HAIGH: Actually you’d also mentioned that this paper was in a sense, a spin-off.

STEWART: Oh, yes, “Perturbation Bounds for a QR Factorization.” Yes, I had to prove a
perturbation bound for the QR factorization as part of the application-- Well, that’s the
interesting thing. That’s for the algorithm, I had to prove auxiliary theorems, this general theory
to do it, and this was one of them. And I realized that this looked rather pretty in its own right, so
I just yanked it out of the paper and wrote it up and sent it in as a separate paper, and that paper
has spawned a lot of different papers. So it was the first paper, essentially, that was a
perturbation theory for intermediate matrix decomposition like the QR Factorization. And there
have been quite a few papers along these lines, for various decompositions.

HAIGH: No while we’re on this topic of theory and application; as you’ve worked on both, how
would you compare the kinds of satisfactions and challenges involved in working on a proof
from working on a piece of software?

[Tape 2, Side B.]

STEWART: Well, that’s hard to say. They’re all fun. So is writing up the research. I mean, there
are many aspects to doing research, and I like most of them. In some instances, there really isn’t
much distinction. I would say generally that most of my perturbation theory has proceeded from
trying to analyze algorithms rather than just inventing a perturbation theory because it’s a pretty
theory. But aside from that, I wouldn’t want to say that one is more fun than the other. They’re
all just great fun.

HAIGH: Is there any other work that you’ve conducted or started while at Texas that you’d like
to discuss?

STEWART: Oh, yes. Well there are a couple of algorithms that have proven really pretty
fabulously successful. I guess the first was the work that I did with Richard Bartels was on, I call
it the Lyapunov equations, but now it would be more generally called Sylvester equations and
their variants.

*Communications* was publishing algorithms then, and then that function was spun off to
Transactions, as mathematical software. It took me back for a moment, because I would have
sworn it was TOMS, but it’s not. [(with R.H. Bartels), “Algorithm 432: The Solution of the

This particular problem was an important one in a lot of control theory. Someone had come and
asked Richard about it. And Richard brought the problem to me and we began talking about it.
The approach that people had been using was through a decomposition called the Jordan
Canonical form, which is practically uncomputable. Jordan is the 19th century mathematician. So
we fell into talking about it, and it quickly became evident that the real way to proceed with this
thing was to work through a much more computable form called the Schur form. The thing is that very few people were aware of the Schur form’s existence, so we got in on the ground floor, sort of. The form itself goes back to the early 19th century I think Schur paper appeared in 1906 but it was part of some very abstract mathematics. I was only the early ’60s that people began to use that notably in the Francis version of the QR algorithm. So we saw this, and the algorithm just sort of fell out. There were a lot of things to do to take care of special cases and this sort of thing, but the basic outline of algorithm came out very quickly. And this was published, and the algorithm is widely used and widely cited.

HAIGH: That’s interesting, the form being very obscure. Was that because until you had a large, digital computer the computability of it would not have been an issue in the first place?

STEWART: Well, partly. But that’s true also of eigenvalues and eigenvectors, too, in general. They only came into play, at least for reasonably large problems, when we got computers. To illustrate this, a lot of eigenvalues problems can be recast as root-finding problems with polynomials. Now Wilkinson, in his early days, or in the days of the national laboratory, had routines for solving polynomials, and routines for solving eigenvalue problems, eigen systems. Now he noticed, that as the routine for eigen systems came in, the routines for solving polynomial problems started being unused—that their use declined. The Schur form is a type of eigenvalue decomposition, so yes, it couldn’t have been used on any large problems before. But I don’t think it was even seen as a very interesting form until the Francis version of the QR algorithm came out, which actually computes the form. And then I don’t know, maybe Richard and I were the first to realize that this was a form that could be useful independently of the fact that it was a step to solving the eigenvalue problem; that you didn't need to go all the way to eigenvalues and eigenvectors to solve certain problems, if you just had the Schur form. There may be other people who had done that, but that certainly was one of the earliest.

HAIGH: In that case, you were publishing the program to carry it out as an algorithm in Communications of the ACM. So in those days, the algorithms had to be in ALGOL, didn't they?

STEWART: No. Ours was written in FORTRAN.

HAIGH: Oh, they’d loosened up by then.

STEWART: Yes. It was Numerische Mathematik that really insisted on ALGOL.

HAIGH: Oh. And how much effort did people put into making those algorithms reasonably efficient, portable, usable implementations versus just being a proof of principle?

STEWART: Oh. Well if you published it in the ACM or a place like that, then you had to make it do a professional quality implementation, and this was.

HAIGH: So those algorithms wouldn’t just be seen as an adjunct to the paper. They would be seen as software that should be really usable industrially.

STEWART: Yes. Yes, oh, yes, very definitely. It was intended to be used, and it was used. I’m sure people have coded their own versions since then. And there have been improvements of it, but the code was definitely used and was intended to be used.

HAIGH: Okay. I understand that you also, in this period, want to talk about this paper with Cleve Moler.

STEWART: Oh, yes.

STEWART: Yes, that was it. That, in many respects, is probably our best [chuckle]. So the way it happened was… let’s not worry about what a generalized eigenvalue problem is, but I had resolved to work on it. It looked like an interesting problem.

HAIGH: All right. Well without learning too much about what the problem was, did you decide to work on it because you had a sense that this was a problem that people out there, who actually had matrices to crunch, would be able to do their jobs better if it was solved? Or was it more attractive on a purely intellectual level?

STEWART: Well, no. It was definitely because it was practical. I very seldom look at an algorithm, just because it’s, you know, that there’s a problem out here, but no one is using it, but there’s a pretty algorithm. I tend to call this an algorithm in search of a problem, and I’ve never started that way. No, I mean the problem’s been around for a long time, and there were ways of solving it, but not entirely satisfactory ways. I don’t want to go into the technical details as to why, but the algorithms that had appeared before, well just say that they all involved the inversion of a matrix or the product of a matrix with it’s inverse in some form or another. This can generate very nasty instabilities.

HAIGH: Now that you mention it, I did see a passage, I think in your textbook, where you were trying to talk people out of inverting matrixes, and telling them that they shouldn’t be, but that they somehow thought that they needed to.

STEWART: Yes. By the time I wrote that textbook, that was the common lore of the people who knew about matrix computations, but it had not been around enough; only about a decade or more that it really spread out into the other communities. So I did that. But in any case, I had gotten interested in solving this algorithm I didn’t quite know how I was going to do it. And I was, at the same time as I often do, working on the perturbation theory while I was doing it. I was thinking about algorithms. Then Cleve came to visit. This was the first time that I had met him. I think we may have seen each other in passing at meetings or this sort of thing.

HAIGH: Was he in New Mexico, at that point?

STEWART: No, he was visiting in Stanford I know, because I went out to write the paper with him and we stayed at his home in Stanford. But he was visiting. He was formally at Michigan at that time, I think, and was visiting at Stanford. But in any case, he came to give a talk, and then he came to my office after the talk. And the night before, not being able to resist doing some algorithms, I had been thinking about the eigenvalue problem and got a special reduction that’s equivalent to the reduction that you get in the Francis QR algorithm for ordinary eigenvalue problems. It was a nice clever reduction, and so I showed it to Cleve, and then we started discussing and wondered if the trick that makes the Francis QR algorithm work would work in this case. It became pretty clear, after some discussion—not too much—that it had a damned good chance of working. You know, this is what I meant when I said the QR algorithm turned out to have more ramifications than anyone expected there. So Cleve cancelled his plane flight and came home with me and stayed at night, charming the hell out of my little son—Cleve’s got a really nice way with children. Then we came back the next morning and worked out the details...
of the algorithm. Then I did a preliminary program of it and sent it to Cleve at Stanford and he ran it.

There’s a sort of interesting story there, because I got a telephone call from Cleve, saying, “There’s something wrong with the algorithm. It isn’t working.” And I said to him, “What’s wrong?” He said, “This intermediate form…” the one that I had thought of the night before, he said, “…it’s all different. It’s not the same.” And I said, “Have you looked at the eigenvalues and see how they are?” He looked. He says, “Oh, they’re fine.” It turns out that this was a classic example of something that’s known to people who knew numerical linear algebra. In matrix computations, you can often have algorithms that work through intermediate forms that are completely inaccurate. The slightest change in the thing will completely change that intermediate form, but the final answer is perfectly accurate. In other words, this is a violation of the idea that a chain is only as strong as its weakest link. I can’t go into the technical reasons why that’s true, except that anyone who knows backward error analysis would understand it. But this was a wonderful example of it. And in fact, when he moved back to Michigan, with the same deck of cards and ran it on the machine, it was the same IBM machine, same model, same operating system, and everything else, he got a complete different set of intermediate answers. And we tracked it down. He found that they had made a change in the square root routine, so that the last bit in the answer was changed slightly every once in a while. That was enough to throw it off.

But in any case, the algorithm was…we did it. We didn’t publish a code because Cleve wanted to do it in EISPACK. It was put into EISPACK, so we didn't formally publish a code, and the code that I wrote was proof of, you know, was basically to show that it worked; it wasn’t of quality software. But this, I would say it’s been certainly the most successful algorithm that I’ve had my name on.

HAIGH: Ah. So it would be an example of one of the things that made its way into EISPACK, that hadn’t been in the earlier Wilkinson publications and the Wilkinson-Reinsch handbook?

STEWART: Yes, one of the few things. EISPACK was pretty much a literal translation of the Algol [from the Handbook].

HAIGH: Okay. Now I know that the QR algorithm has a very high profile now. When people think about the QR algorithm now, how much of it is a result of this generalization? At the time, was it apparent how important the algorithm was, or did that only become apparent some years later with this work in order to generalize it.

STEWART: No, no. Francis’ QR algorithm was a major breakthrough. I mean, the QZ algorithm, that’s the algorithm that Cleve and I did, is a nice piece of work, but it would have never been done without the original breakthroughs in the QR algorithm. That was immediately perceived by everyone who knows.

I can illustrate it with an anecdote that Wilkinson used to tell. When he learned about this, he coded it up Francis’ version of the QR algorithm and started to run it with a matrix whose eigenvalues he knew. At the time, the output for the computer was a card punch. So the algorithm started it out, and it went “Chunk!” and out came an eigenvalue. “Chunk! Chunk! Chunk!” [Said very fast] “Chunk, chunk, chunk, chunk,” driving the card punch as fast as it could go. Well, this was unheard of, and Jim claims that he looked over for two days for the error in the program before he looked at the answers and found out they were correct. So that’s how stunning the algorithm was, that even Jim, who would have certainly had a theoretical
appreciation of what’s it’s properties must be, couldn’t believe it when he actually saw it in action.

HAIGH: Had other people had this idea of trying to generalize this aspect of the QR algorithm?

STEWART: Well, this has gone in two ways. Actually, Gene Golub’s method for computing the singular value decomposition can also be regarded as that.

HAIGH: Did Gene Golub look at it in that way?

STEWART: No he did not. One night I was lying in bed, and I said, “Gee, can we do this with the singular value decomposition?” As a matter of fact, don’t hold me to it, but I think it was before the QR algorithm, we did the QZ algorithm, so I may have had the pump primed for this kind of thing. But I was lying in bed, and I said, “Gee. You can do this wonderful thing and get the singular value decomposition algorithm for it in the generalization.” And then the next day, I went down and I looked at Golub’s algorithm as implemented by Wilkinson and Reinsch in the handbook series. It turned out to be exactly the same. So I was very excited for a time, but there’s nothing new under the sun.

Well, in any case, it was not derived regarding that, but it turns out that that’s the way people explain it now, as marrying it with the QR algorithm. Remember, we want to call it the “Francis QR algorithm” because there were precursors that weren’t very successful for one reason or another.

HAIGH: And you mentioned that that was the first time you had met Cleve Moler.

STEWART: Yes. Well, as I say, we may have met in passing, at meetings.

HAIGH: But that you’d really interacted.

STEWART: Yes.

HAIGH: So aside from his ability to charm your son, how would you describe Cleve and your interactions with him?

STEWART: Well, Cleve’s an old friend, and I think the hell of him. He’s just got a lot of virtues. He’s a bit of a bear-like person at first sight. He’s as opinionated as I am. But he’s got great good will, and is as sharp as a tack. He’s, for example, a far better programmer than I am. I’ll tell you an anecdote about that at some point, or when we come to talk about LINPACK. He’s an excellent, wonderful writer. He was also editor of the school newspaper, I believe, at Caltech, I think, when he was there. So he’s got this nice rapport to his style—direct, very direct. My writing tends to be more… well you can call him Handel and you can call me Bach. Neither of us are in that category, but Handel works in square, nice rhythms, and Bach is more sinuous and their styles are very different. But I always enjoy reading something that Cleve does. So generally, he’s been wonderful for the profession and everything. And I’ve been delighted to be his friend.

HAIGH: Okay. So we’ll obviously return to that relationship with the LINPACK Project. Does that conclude the research that you did while at Texas that you would like to discuss?

STEWART: Yes. I think that’s the major.

HAIGH: Okay. So how did you come to move on from Texas?
STEWART: My wife at that time, developed an allergy called Cedar Fever. There’s a cedar tree that grows, I gather, only around Austin that people develop violent allergies to, and nothing really helped her. So we decided that I needed to move. I was very happy at Texas. They were treating me wonderfully—there was no reason. So I was flying home from a trip from somewhere, and I was in Pittsburgh. And Joe Traub was at the time Chairman of the department there. So I called him up on the phone and said, “Joe, do you have a job?”

HAIGH: Did you know him, prior to that?
STEWART: Yes, I had met him.

HAIGH: So he was active, at that point, in numerical analysis?
STEWART: Oh, yes, he was quite active, then. I think he had had me visit at Bell Labs, and this sort of thing.

HAIGH: To give a paper?
STEWART: To give a talk. So yes, I knew him reasonably well. I’d stayed a night in his apartment in New York. I’m not sure if at that time they were married, but with his now wife. And so you know, we were known. So I called him up and asked him if he had a job at Carnegie-Mellon, and he said yes, and so we moved to Pittsburgh. That was sort of interesting, because two years later, my wife and I were divorced, and at that time, I decided that I really didn’t want to stay around Pittsburgh and so I moved, once again, to Maryland, which was the last academic move that I’d made. So Carnegie-Mellon was sort of a, you might say, a side journey.

HAIGH: So, other than it’s lack of allergenic cedars, what drew you to Carnegie-Mellon?
STEWART: Well, it had a good reputation. Joe was there, who I liked. It just seemed like a reasonable place to move. And there was a job. So I did there. Actually, it’s probably one of the rare cases, but they would hire me as an associate professor but they wouldn’t hire me with tenure, so I actually gave up tenure to go to Carnegie-Mellon. It was a certain naïveté that I didn't really understand what tenure was in those days. But with no regrets. There were very nice people there. Bill Wolfe, who is now the president of the National Academy of Engineering as was there at that time. His now wife, Anita Jones, was also a member of the National Academy of Engineering, was there at that time. And there were people... Herb Simon was there, and Allan Newell. They were both live wires. I wasn’t much interested in their area, but it was nice to be around.

HAIGH: Now was that another joint appointment?
STEWART: Yes. That was the last joint appointment in math that I had.

HAIGH: So how was the mathematics department in its relationship with computer science?
STEWART: Well, it practically had none. The math department did its thing and the computer science department did its thing, and the computer science department was a far more exciting place. But you know, I taught my courses in the math department. They were all in the same building. At Texas, I had always had a problem that the math department was in one building, far away enough to make it an inconvenience just to drop over. But at Carnegie-Mellon, there was none of that problem. My main objection, I guess, to Carnegie-Mellon was that Joe was not doing as much in numerical analysis at that time, and I really was the only person in that area. I really wanted a place where there were more people to talk with.
HAIGH: So is that why you left?
STEWART: By and large, but also I liked Washington. I was born here. I was raised my first five years. My uncle who had come home from the war who had married my babysitter was here. So I frequently, ever since high school, had come up here for vacations and things like that. When I was in college, my parents would give me the car to take a vacation at the end of the summer, and Washington was always a stopping place. So the whole area was very attractive to me. And you know, I didn't want to stay around the scene of my divorce, and that had unpleasant associations. So I interviewed at Maryland and I liked what I saw.

HAIGH: Did you consider any other possible destinations?
STEWART: Well, I considered Texas again, because again, they had been very good to me. I can’t say the decision was entirely rational, but Maryland won out. Instead of getting joint appointments in math and computer science, I took an appointment in computer science which was joint with what was called the Institute for Fluid Dynamics at this time. It later became known as the Institute for Physical Sciences and Technology. So I had a joint, with reduced teaching load there. That worked out very nicely. Again, all the departments, the IPST and the CS department were in the same building, and so it was a very nice arrangement.

HAIGH: So before we talk any more about Maryland, chronologically we’ve reached your Introduction to Matrix Computations. [G.W. Stewart, Introduction to Matrix Computations, Academic Press, New York, 1973].

STEWART: Yes. I actually started that in Texas. There was an undergraduate course in numerical linear algebra, and there was really no textbook for it. The one textbook was Leslie Fox’s *Numerical Linear Algebra*, and it was dated [L. Fox (1964) An Introduction to Numerical Linear Algebra, Oxford University Press, Oxford]. That’s all—it didn't have the modern stuff, the things like the Francis QR algorithm. So first I began making impromptu notes for my class. As I said, I like to write, and so producing notes is no difficulty for me. But then I realized that I probably should write a textbook for this. So I basically sat down and started writing. On Tuesdays and Thursdays, I didn't have anything in the morning, so I stayed home and wrote for four hours, two days a week, and then came in, had lunch and came in to the University and did my usual thing. The book was largely finished before I got to Carnegie-Mellon. There were lots of things to do to get a book published, as I’m sure you know.

HAIGH: Did you talk to publishers early in the process?
STEWART: No. As a matter of fact, I’ve never done that. I mean, now that I’ve been having such a long relationship with SIAM. But usually I’ll get my books mostly written before I even consider talking with someone. I don’t really like to have an unfinished chunk of thing and an obligation to a publisher. So this was largely written... But again, it was at Texas that I taught. And now I’m not sure of my memories here, but it may well have been Werner Rheinboldt who put me in touch with Academic Press. It wouldn’t be acknowledged in there.

HAIGH: No, but there’s a list of books, and it says in here that he was the series editor.
STEWART: Okay. That was it, yes.

HAIGH: And so this series in Computer Science and Applied Mathematics.
STEWART: Yes. He was the series editor, so that must be, and of course, Werner Rheinboldt was instrumental in getting me to Maryland, too. He was in the computer science and math
department. So in any case, I started it. I put a poor graduate student of mine through all the exercises in the book, Margaret McNamee, now Blevins, which she shortly became—she married while I was there. I got an email from her, fairly recently saying “Hi.” She’s still at Texas, working for some programming institute, along with another woman who was a graduate student at Texas that I knew very well, and whose husband I knew. So it’s sort of a small-world phenomenon. But in any case, the book was very well received, if only because there was no other. I was very pleased when I was told that it was stolen from the Harvard Library.

HAIGH: You say in the preface, I think, basically, that the experience has been accumulated since digital computers became available but hadn’t found it’s way yet into linear algebra textbooks; that the methods were all still…

STEWART: In tech reports and things, and journal articles.

HAIGH: And the book basically had this niche to itself.

STEWART: Yes.

HAIGH: Now, did the areas that you were covering in the book, were those basically the same that you would have covered in the class that you were teaching?

STEWART: Yes. Probably with some selection. You always have to slightly overwrite or expand the textbook to give the people who are using it some scope to tailor it to their course. So there’s probably more than you could cover in a semester’s course there. As Werner Rheinboldt pointed out, the gradient gets much greater as you get toward the end. The stuff is more difficult material. But it was there. I can say I really enjoyed writing it.

HAIGH: You have a whole chapter at the end, on the QR algorithm, and another chapter on the eigenvalues and eigenvectors. So obviously the QR algorithm chapter wouldn’t have been there in an earlier textbook.

STEWART: No, it wouldn’t.

HAIGH: With respect to the other sections, are those topics that would have been covered that you’re just covering a different way from earlier texts? Or are there entire topics here that just would not have been covered in a course that wasn’t oriented towards the realities of digital computation?

STEWART: Well, they would have been taught in some form or another. I don’t know that the business of the linear least squares problem, the way I approached it, would have been taught that way. The world was still pretty much immersed in the normal equations approach, rather than using the QR decomposition to solve it. So that would have probably been different. I mean the breakthrough paper on that was Gene Golub’s 1965 *Numerische Mathematik* paper on using Householder transformations to solve least squares. [Businger, Peter and Golub, Gene H., "Linear Least Squares Solutions by Householder Transformations," Numerische Mathematik, 7 (1965), 269-76].

I guess it’s sort of interesting, and I’ve told Gene this and we laugh over it. But when I was over at General Electric, I also wrote a very badly written paper on using Householder transformations to solve least squares. This was probably my first real attempt at beginning to write a paper, but I never did anything with it, and Gene certainly deserves the credit for showing how to use Householder transformations. My work went nowhere. I was very immature as an
algorithmist and mathematician at that time, but I did have the same ideas, or some of them. But Gene’s paper, it was truly groundbreaking, and that’s 1965.

I started working on this maybe in the early 1970s, so there was not much time for this to percolate through the other communities. So that would have been another aspect of the book. I think, in it’s overall organization it was different. No one would have spent as much time on how you would actually code matrix algorithms, the art of writing them. That later developments have made completely obsolescent. Now you have to talk about interaction of algorithms with cache and this sort of thing, and it’s a deeper subject than it was when I presented it. But you wouldn’t have typically seen a chapter on how to program matrix algorithms in a book. So there were some new features, other than just the technical content.

HAIGH: You mentioned that you had enjoyed writing it.

STEWART: Oh, yes.

HAIGH: Why was that?

STEWART: Well, first I enjoy writing. But I had been learning all of this stuff since I had started programming Wilkinson’s notes. And Wilkinson, you know his tome *The Algebraic Eigenvalue Problem*—that was my Christmas present when I was in graduate school [Wilkinson, James H. *The Algebraic Eigenvalue Problem*. Oxford: Clarendon Press, 1965]. It cost $22, if you can imagine. Just look up the price now. And I studied it in great detail. I had earlier studied his rounding errors and algebraic processes in detail, and I had read papers and this sort of thing. So I had basically immersed myself in this computational linear algebra, and the chance to just let it flow out in a natural way, it was almost pre-organized by then in my head. The only part that I hadn’t really scoped out was how to explain the QR algorithm and experience. So I knew it all. It was at my fingertips. I didn't have to go to references or anything; I could just write. And that’s one of the best feelings, to be able to write from a secure knowledge of what you’re writing about and only concentrate on, “Am I presenting it right,” or “Is this the right way?”

I did discover the virtues of a great big wastebasket then. For the chapter on least squares, I actually I got involved in pseudo-inverses and wrote a highly technical sort of unnecessarily mathematical treatment of the subject. Then I came to my desk on a Thursday morning, I said, “This chapter will never fly,” and I just tossed it in the trash can and the garbage man take it out so that I wouldn’t be tempted to try to pick up pieces of it, and I rewrote it completely in the way that it should have been written.

HAIGH: Now one of the things that I don’t see in the book is chunks of example code.

STEWART: No. Well, no actually, there are; they’re there in a pseudo language, see? See, we’re now operating at the matrix level here, so that’s…

HAIGH: So this is in an imaginary high-level…

STEWART: Yes. INFL, which means “informal language”.

HAIGH: …matrix language?

STEWART: No, not necessarily. It’s sort of a combination of matrixes and language. Here you’ll see a loop and this sort of thing. It’s basically a combination of mathematics and coding constructs.

HAIGH: Was that a common approach, or was that something you came up with, yourself?
STEWART: Well, for example, this way of writing loops, with the little "l" there, was some, actually and who was it? It was a thin book on numerical analysis, very nice. I used it in one of my classes once, and I can’t remember the name. So that part, yes. I mean pseudo code had been used widely, so I wouldn’t say that I was the first to do it. I think that I may have been the first to do it systematically.

HAIGH: I know, for example, that early programming textbooks would describe an imaginary machine and then present assembly language for this imaginary machine.

STEWART: Yes. No, this is not an imaginary machine. Yes, you see in this section, there is lots of it. And they even do a FORTRAN in a problem. You can see there’s a great deal of algorithmic stuff there. It’s easy to miss. More or less deliberately, the book divides sort of into theory, practice, theory, practice, theory, practice. I start off in the basic linear algebraic theory and then move into coding matrix operations, so that people can get a breather from the mathematics, and so on. It goes on like that, and I try to alternate that stuff to get a nice rhythm. So if you get open to the wrong part of the book, you’re not going to see it in programming.

HAIGH: So would your expectation be that if students had this as a class textbook, that some of their assignments would be to write FORTRAN programs to implement…?

STEWART: Oh yes, definitely. Well, when I taught—I’ve never taught from my book, but when I’ve taught these courses, very definitely we had projects in which the students would implement these things. MATLAB has, by and large, made this unnecessary, because you can manipulate things at higher levels. So I always felt that these projects, to some extent, were wasting the students’ time with working at too low a level. Some of it’s good, but you could spend an inordinate amount of time just on the mechanics of it. Remember, you had to walk down to the computing center and submit your cards and everything to get a run; in those days, you couldn’t type it in, at a terminal and do 50 compilations until you got rid of all your mistakes. So a coding project took a lot more time for a student than it would today.

HAIGH: So at this point, you’re presenting some pseudo code and the exercises would be to write in FORTRAN.

STEWART: No, no, no. That, sometimes I’ll say, “Write in pseudo code.” I just noticed some FORTRAN code there.

HAIGH: But if the students actually wanted to write a program code and run it, they would have to do it in FORTRAN, wouldn’t they?

STEWART: Yes, they would have. That is, in language, yeah.

HAIGH: But if they were doing it today, they could do it at the high level in MATLAB, closer to the level of your pseudo code.

STEWART: Yes.

HAIGH: Was there any period in between, where the exercises might allow them to use the software library, but they would have to know what to call it or to use routines?

STEWART: We have two courses at Maryland (and I’m suspecting that other schools have this distinction) that were originally considered numerical analysis. CMS 460, at the undergraduate level has some analysis of the algorithms’ derivation (you can’t do too much with the level of mathematics that the people have) and then projects to push it in. Our other course, 660, could be
called Scientific Computing, although that really wasn’t a subject then. In that course, people generally used packages for solving ordinary differential equations. It takes a much broader scope, and you learn that there are algorithms for solving these problems and here’s software to do it, but you don’t learn how the software works. So both approaches are perfectly legitimate. This was before you would have distinguished between those approaches.

HAIGH: Right, because the packages weren’t really there to use.

STEWART: No.

HAIGH: So you’ve mentioned the book was successful in the sense that presumably it sold a lot of copies. Do you have any awareness of any specific ways in which it was used, or different kinds of courses it might have been assigned for? Do you have any stories that you’ve got back from professors who taught with it, or anything?

STEWART: Well, from the people have used it, I’ve gotten practically nothing but very nice comments on it. There’s only one course you can really teach and that’s an introductory course in matrix computations, and it’s been used for that. There are other books out there that people use now. As I say, a lot of this is somewhat obsolescent; you know, it was written about 30 years ago. But the thing has been favorable. I mean, the woman who lives next door to my father-in-law, who she and her family live there, I found out took a course from my book.

HAIGH: Was there a reason you didn't do a second edition, say, five or ten years later?

STEWART: Well, actually I seldom look back on books. I think that you can get mired down doing second editions. But I did want to consider revising it, and then making a second edition with large systems; this is basically focused on dense, smaller systems. By that time, Academic Press had been sold and the people who had control of the book, the editors, didn't really know what they doing. That was the General Electric light bulb, a manufacturer type philosophy, and I realized that I'd get nowhere with Academic Press on doing this, and that’s what started me off on my matrix algorithms series. I said, “Yes. Let’s not look back and try to revise it.”

[Tape 3, Side A.]

HAIGH: Well, that probably concludes the discussion on that first textbook, unless you have anything else to say.

STEWART: No.

HAIGH: So should we then pick up the story of your career with your arrival at Maryland?

STEWART: Sure.

HAIGH: Okay. So you’ve mentioned reasons that you wanted to move on from Carnegie-Mellon, in terms of many things that drew you here. You’ve mentioned the appeal of the D.C. area itself, and also of getting away from having a joint appointment.

STEWART: Well, that didn't actually draw me here; that ended up being the arrangement.

HAIGH: So what was the reputation of the UMD Computer Science department at that point?

STEWART: Well, it was a starting department, but it had a good reputation because they had done exactly what Texas had done, and that is they had gotten senior people to start the department and to conduct themselves actually being in the department and living in it. Notably, you had people like Werner Rheinboldt, who was very well known, and of course his SIAM
career is well known. But in addition, we had people like Azriel Rosenfeld, who was one of the top people in computer vision. We had Jack Minker who was very strong in AI, artificial intelligence. All of these people were coming in, not from computer science originally. And then we had some others, perhaps not quite as strong, but all of them seasoned researchers. So just before I got there, the computer science department hadn’t been a department. It had been the Education Grant of the Computer Center. The intention was always to make it a separate department, and just before I came, they really had, and Jack Minker was the chairman. I interviewed-- No, no, he wasn’t. It was Bill Atchison who was chairman when I interviewed.

So the department was just forming, and they had another nice group of assistant professors who really built the department. I mean they were the ones who knew how to make the courses run and everything. They’re sort of heroes, in a sense. One of them was actually a graduate student at Texas when I was there; Vic Basili, and he has made quite a name for himself in software engineering. So the department felt nice, if you know what I mean, and then there was this appointment in the Institute for Fluid Dynamics and Applied Mathematics, IFDAM at the time, that a couple of years later became the Institute for Physical Science and Technology, and there were nice people. Also, many people in the math department who were applied mathematicians, had appointments in IFDAM. So this was a double plus in the sense that I got the best of all possible worlds, in the sense that I did have the contacts with the math department with the people that counted, to me. So as far as I was concerned, it was a very nice appointment, and I’ve never considered leaving Maryland since then. So I could say it’s partly a two-body problem, because my wife had a nice job at the Office of Naval Research during all of this time. Actually, we married afterwards, after I met her here, and we married afterwards. So in some sense, I was stuck here, but I’ve never felt stuck. I just don’t, the department and the situation has just been too good.

HAIGH: Now we have a quote from you that “It’s easier to do math in a CS department than CS in the math department.” That would tie in with what you were just saying, but do you have a sense of why that is that’s how things evolved?

STEWART: Well, yes. There’s a long tradition of “pure mathematics”, and this has a great hold on the math departments. The math departments, no matter how good they are in applied mathematics, are not going to get their ratings from it. They may be highly respected, they may have lots of students who go to it for that, and can attract very good faculty in applied mathematics, but they’re going to be rated on their abilities in pure mathematics. So this, to some extent, creates a standard in which the applied mathematicians can be less than appreciated. This is far less true today than it was say when I arrived at Maryland or before. Just to give you an example, do you remember about the four-color problem and when it was solved by a computer program. There was great debate with the purists lining up against this being an acceptable proof. Of course now everyone would accept it as an acceptable proof without any difficulty. We understand computers, and they’re just another form of pencil and paper for that kind of thing. It’s just a matter of data manipulation and if we can extend our things, but that was not the truth at that time. So it’s very difficult for a person to feel really appreciated within a math department, working in the more applied areas, unless there’s a strong group to support them. And there is at the University of Maryland. I can say a few more words about that in a moment.
But in a computer science department, the theory of computation is well recognized. Mathematics is always useful in a large number of areas; computer systems, where queuing theory is important, even in compiler construction where there’s a great deal of algebra languages that prove useful. This is sort of a superannuated topic because compilers are good now, and you know, people aren’t even taking the classes very much anymore. It’s a very specialist topic. I know it does, but at the time, there was the theory of languages and this sort of thing. So any person with a mathematical bent has ample opportunity to exercise it within the computer science department, and they’ll be generally appreciated for it. So that’s how I see it.

HAIGH: And leading on from that, what was the place of numerical analysis in the teaching of computer science there?

STEWART: Well, this is the unusual thing. I said that the applied mathematicians are well provided for in the math department. In fact, what happened, what I came into was this situation, because we had the mathematicians that were half time in IFDAM and half in the math department. All of the numerical analysis became the joint property of computer science and the mathematics department. So instead of having it divided up, we had a joint field committee to administer it across both departments. It was jointly listed between the departments. Later there was an applied math program that came up, and then the listing shifted toward, to the applied math program which had a different nickname, but it was still more or less within the math department. So we had this very nice arrangement where numerical analysis was more or less respected in both departments, and we could have the staff to have a wide variety of interesting courses. Particularly, we were fortunate to have Ivo Babuska who was just fantastic at rounding up students, at least for his area in partial differential equations. And Werner Rheinboldt, again, I mentioned was a leader. And this has continued to this very day. The only downside I can think of it is, probably our strength is not quite as appreciated as it should be, because people look at us as individuals in computer science or in mathematics and have a hard time seeing the thing as a whole.

HAIGH: Does that nature of the expertise make it harder to attract graduate students to do Ph.D.s in the area?

STEWART: No. Actually, my colleagues Dianne O’Leary and Howard Elman and I actually tend to get our Ph.D. students from the applied math program. Now, it’s now called the Applied Math and Scientific Computing Program. It’s had a refurbishment. But we tend to get our students from that, although I have a student right now who’s in the computer science department.

HAIGH: Okay. I know where Dianne O’Leary has been in the department, and is also very active in the area. So was she already there when you arrived?

STEWART: No. No. We hired her a few years after I came. She had gotten her degree under Gene Golub at Stanford, and then spent a couple of years at Michigan while her husband got his medical degree. He had gotten a Ph.D. in biochemistry or something like that, and then got a degree and then they moved here. Being associated with Dianne has been one of the best parts of my professional life. We’ve worked together. We actually don’t do a great deal of actual research together, but we sort of support each other in other ways, you know, just knowing each other and we’ve always had joint grants of some form or another. And she’s down the hall from me and we chat about our families, etcetera. She performed the rather remarkable feat of going from assistant to full professor while at the same time having three children. So she’s very well
organized, and her husband Tim is great with the family. So she’s been great. And later, we had
Howard Elman join us from Yale. And he’s been a wonderful colleague. So those are the three in
the computer science that do this kind of scientific computing, and it’s been, as I say, a very nice
and satisfactory thing.

Start of Session 3.

HAIGH: So we’d talked in general terms about your arrival at the University here and the place
of numerical analysis in the work at the department, the relationships with the mathematical
department, Dianne O’Leary, and some of your other colleagues. I think you’d also mentioned
the Institute for Physical Science and Technology. What kind of form did your involvement with
that take?

STEWART: Well, it was basically a research position. At the time that I got there, as I had said,
it was called IFDAM, the Institute for Fluid Dynamics, and it was basically a collection of
physical scientists and mathematicians, some in meteorology and various other fields. And
several mathematicians had permanent half-time appointments in the Institute. Mine was just
another there, and basically I did my thing by interacting with the mathematicians in the
department. It was later changed. We got a provost who was an engineer, and wanted to change
its direction so that it included engineering, and so that’s when it became Institute for Physical
Science and Technology. It didn't change much, except that some of the appointments in
mathematics were reduced—there were no more full time appointments in mathematics. But
otherwise, it was pretty much business as usual. We moved at some point, from where we were
with IPST to where I am now, the AV Williams Building. And roughly about that time,
UMIACS was formed, and so I applied for a permanent appointment there and got it, and
dropped my appointment in IPST.

HAIGH: That was a personal move by you, but the Institute for Physical Science and
Technology still exists.

STEWART: Yes.

HAIGH: And it’s still doing basically the same thing.

STEWART: Yes, much the same thing. Well I haven’t kept up with it, but it’s essentially that.
It’s a very good institute. It has Michael Fisher, who’s a National Academy member, and some
other very good scientists there.

HAIGH: How about the evolution of the computer science department itself, over time?

STEWART: Well, when I came, it was still rather small. We had this compliment of assistant
professors. I came in as an associate professor and was promoted to full professor the next year
after I came. And the senior faculty, who some of them could be called straight computer
scientists, but most of whom had come in from other areas. And basically, the department grew
by having young people promoted. I was perhaps, the last outside hire for many years at the
senior level. So it’s a homegrown department, and that has worked out very well with us. Very
few people have left us. We’ve had, for example, a database person stolen from us. You can also
look at the caliber of where people go to. But it’s a happy department, and most people seem to
be very happy to stay. I know I am. It’s developed. It has the usual academic squabbles, which
are not very serious, mostly over things like how we’ll organize the comprehensive exams, and
this sort of thing. But generally, there have been few growing pains.
One growing pain was when UMIACS was founded, that we were suddenly given money to bring in 15 or 16 new assistant professors, and we did this in two years. The department was a little overwhelmed, there, and I don’t think they got quite the mentoring that they should. And when that bulge hit, five years later, it hit the promotion cycle, it was a very difficult time. I mean we survived and no one was permanently mad at each other, but it was a lot of people to consider in one time. But other than that, it’s been a good; it’s developed. It’s stayed near the top, especially in public universities. We tend to be in the top ten. We actually had been very well, even from the earliest days, very well thought of in the sense that, for instance, the National Academy of Sciences rated computer science departments, and basically, because we had a very strong publication record, we ranked rather high in those. So it’s a well thought of department. It’s not in the Stanford, Carnegie-Mellon, MIT bracket, it’s down a tier, but that still means very good, indeed. So otherwise, it’s developed. Just as you would expect, as different subjects like networking come in we hire people. And other people sometimes change their areas as they go.

We’ve had remarkably few people who have dropped out at the associate professor level. That’s a constant danger in a department; you promote people and then they take a permanent vacation. And we do have a very few, but as a percentage, it’s negligible. The Department can easily support these people.

HAIGH: Would you say that there’s anything distinct about the way that the program is structured, or about the research areas, compared with other leading departments?

STEWART: Not really. We are very strong in computer vision and things, and that’s a little out of the main core. With Dianne, Howard and I, we probably have more in scientific computing than you would see in a typical university. Otherwise, we cover databases and networking and computer systems and programming languages, some of the more central areas, if you want to call it. The university has founded a center for bio informatics and computational biology, like everyone else is doing, but we’ve gotten a very good guy in the name of Steve Salsburg, who’s directing it and is half in the department. And he has three appointments to make jointly with the department and his institute. That, of course, there are going to be some growing pains there, because to outside eyes, it looks like these people are just doing genomes and assembling genes and this sort of thing.

So if you look at it from the view of many of the people in the department, the people that they are trying—well, there’s one person’s who’s there and Steve wants to bring him into the department, and it looks like he might just be just sort of a technician in one of these large groups. Actually, I went over Thursday. This had all come up in meetings and this sort of thing, and I went over Thursday to Steve’s office, and we had about an hour’s chat on this. And I think things are going to work out very nicely with this. The one thing I had not been aware of is that it isn’t just doing these massive projects with the 50-author type things. They have journals where these people get out and take their individual contributions to these projects and book them in, and that will probably diffuse any difficulty that it can be. Because all of the other areas of the computer sciences are exactly like that. You know, I don’t publish in database journals and this sort of thing. But with the existence of things where people can publish with two or one, or five authors there, it should be nice. So that looks like it’s going to be a very nice expansion of the department.

HAIGH: While we’re discussing the department as a whole, I know that in different ways many departments have formed ties with local industry or produced start-ups. Have there been any companies or government agencies which the department has formed particular ties?
STEWART: I don’t pay an awful lot of attention to that. There have been some and one of our professors had some associations with a large corporation in Japan. And so there is some of that going on. I don’t think that it’s to the extent that it is in some other departments, where it’s almost unseemly commercial. But definitely, there are ties among the people. I’d certainly not been involved in any of them.

HAIGH: Has the department realized any benefits from having all these governmental agencies and institutions at various kinds in the region?

STEWART: Oh yes, undeniably. I’ve never broken them down by department, but if you look at the figures of what the government grant money is going to, it’s going to, with the exception maybe of California, it’s inversely proportional to the distance from Washington. So we’ve benefited enormously in that way.

HAIGH: So I’ll shift and ask you a few questions about your personal role, and then we shall be talking a bit more about the Institute for Advanced Computer Studies as well. So I think you’d implied that you had continued this pattern that you had first developed at Texas of attempting to teach courses in different areas of computer science.

STEWART: Yes, yes. So I mean I’ve taught courses in operating systems, and computer languages and compiler construction, networking. Oh, and then computer architecture and computer systems. I haven’t taught data bases, which I’d like to. So these are outside my main area. In some sense, they’ve all been useful, some more than others. I would say the most has been the computer architecture, but the networking is improving to be increasingly useful as numerical computations spread out over the net and understanding some of the interconnections there. I find it very worthwhile. I must confess that as I get older, my energy is running down, and it gets harder and harder to do this. It takes a lot of energy to just get and take a subject that you have never really studied and then teach an upper division course in it. So I probably will not be doing much of that any more.

HAIGH: Is there anything else that you think has been distinct about your approach to teaching?

STEWART: Not really. I considered myself a pretty average teacher. I think I’m very good at not coming in and stumbling over things in class. I prepare my lectures well, and I give clean assignments. I always make sure that I’ve written out the answers before I give the assignments out, the mechanics. But I’m not a charismatic teacher, as some people are, and I don’t mean that in a bad sense. I’m just sort of a “meat and potatoes” professor that serves it up.

HAIGH: I think you had mentioned that you had never taught from your own textbook.

STEWART: Yes.

HAIGH: Why was that?

STEWART: I may have used it a couple of times. I’m just trying to think. It’s not from any ethical principles. You don’t make enough money off of these things. I mean, the amount of money that professors make off of textbooks is highly overrated, unless it’s a calculus textbook or something like that. So I would have no compunctions about it. Right now, the course that it was written for, we’re not teaching it in Maryland. But other reasons are that even when we were teaching such a course, the parts that were in the book were only part of the course. So it would not have been really suitable. But when we had these courses was a long time ago, and I can’t really remember the details.
HAIGH: Your role as an advisor of graduate students?

STEWART: Generally, I’m not very good with graduate students. I’ve had a few Ph.D.s, and three of them have done very well. Robert van de Geign is a full professor at the University of Texas, my old alma mater. Xiaobai Sun at Duke University, tenured, and Misha Kilmer (actually Dianne and I jointly supervised her dissertation) is at Tufts and was just recently jumped over associate professor straight to full professor. So I’ve had some very good students. But by and large, my problem is, as I mentioned earlier, when I don’t know the answer to a problem, my thinking is very fuzzy. So I have a very hard time advising students what to do. By the time I can help them, I’ve already solved the problem, and that’s not very good for a graduate advisor. I guess you can’t be good at everything. But with good students, I do well. As I say, Robert, Xiaobai and Misha are no trouble. It’s not the necessarily bad students, but the students who can’t just jump at an idea when it’s very poorly formulated.

HAIGH: So do you have anything else to say about teaching or advising?

STEWART: No, not really. You mentioned that large grant for parallel processing. [NSF Coordinated Experimental Research Program for a Testbed for Parallel Algorithms, 1983-89, $4,547,577].

HAIGH: Yes.

STEWART: That should be discussed in context to the department, because it was really an infrastructure grant from the NSF for the department rather than a personal grant for myself. So I think that this would be the best place. At that time the National Science Foundation had what they called infrastructure grants. I think they exist under other names at this time, but the idea was to infuse lots of money into specific departments to advance their infrastructure and their ability to do computing and this sort of thing. So evidently while I was on sabbatical some people had tried to prepare a proposal, and I got back from sabbatical and looked at it, and it was just awful. It was everyone’s piece of the pie type thing. Everyone puts in their own thing. Disorganized. It was that thick, and this sort of thing. So I volunteered to take it on, and basically what I did was I turned every bit of that stuff into an appendix and then just wrote a very general proposal on the parallel computing. We were building this machine called the ZMOB, which was a ring of Z80 chips for parallel computing. And that was the central part of the grant, to build support around the ZMOB, so that there could be things. So low and behold it worked. No one read the appendices. They read the thing that I wrote, and we got through the reviews or the site visits. We got through the first stage. And then, I spent a lot of time organizing the site visits and everything, and we got the grant, and then I bowed out.

HAIGH: So when the money actually came, you weren’t much involved with…?

STEWART: No, I was nominally chairman of the laboratory for parallel computation, but my heart wasn’t in it and it went away. The main problem I had initially was that there were a couple of professors who really considered this to be a pie to be divided up, and I made it absolutely clear that it was going to be used for the department as a whole, not to contribute to any individual professor’s domain. There were some hard feelings there, but they didn’t last. So that was my role, and that’s the one time I’ve done that, and I don’t think I’d ever try it again.

HAIGH: Well, it sounds like a fairly good experience from what you’ve said, so why wouldn’t you?
STEWART: Well, it was a good experience in retrospect, but the actual doing of it takes an enormous amount of energy from the things that I really want to do. I’ve ordinarily just been content with nice small grants that will pay for summer salary in a graduate student or something like that. One of the nice things about the department is that that makes no difference. I really like that. You can organize a large research program with lots of students and lots of publicity, or you can have a small program. As long as the research is published and is good, it doesn’t matter in our department.

HAIGH: So the date on the grant is from 1983 to ’89, so would you have been preparing in 1982?

STEWART: Yes, something like that. It would probably be 1981. The time it takes, I would have been on sabbatical there. Yeah, ’82. It would be something like that.

HAIGH: And had the department already built this ZMOB experimental…?

STEWART: It was building it at the time. There’s an interesting story associated with that. When we were site visited, we had the ZMOBs working in cabinets, but the ring that connected them was not working yet. So needed a very impressive parallel demonstration, and we didn’t have any parallels, so someone (not myself, I’m sorry to say) got the great idea of putting on a Bach fugue on each of these things. They had reeds that they could do it in putting parts in the Bach fugue. Well now, the clocks were accurate enough, so that if you started them off at exactly the same time they would play the fugue without any synchronization, at least for a while, before it would drop out. So this was really very impressive. Of course, we immediately told them what we had done, but the first impressions are important in that kind of thing. So they came in with these three ZMOB boards playing a Bach fugue. I’m sure it helped us, and everyone had a good laugh when we told them, but really it wasn’t what was happening.

HAIGH: So that was ongoing anyway. Now, did you personally have any interest in parallel algorithms?

STEWART: For a while, yes. I got in fairly early. Diane and I worked on some of it together. And then, I basically got out of it. The reason was that at the time I was principally interested in eigenvalue problems and eigenvalue algorithms, and still am. It was becoming clear to me as I did my own research and looked at what other people were doing and trying to do that parallelism never really worked effectively with eigenvalue problems, at least general ones. You always have special cases that you can do this. So I sort of lost interest in it. Incidentally, I think time proven that I was correct. The basic problem is that with a lot of the matrix algorithms where parallelism has been most successful have been algorithms in which you only transform the matrix from one side by multiplying it by other matrices from one side, because usually these things can pipeline. You start hitting it with one transformation and you can bring in the next transformation all the way through. With eigenvalue problems, you’re hitting it with both sides, and the first transformation has to get a substantial way through before you can start the second transformation, before you have the information to compute the second transformation. So they don’t pipeline and they don’t work very well. So as I said, I did some papers, both theoretical and some practical on this, but by and large I grew discouraged. I didn’t feel that I could do the things that I really wanted to do with it and basically turned my attentions elsewhere. So that’s the story of me in parallelism.
HAIGH: As so much of the focus on high-performance scientific computing has turned to massively parallel systems, have there been many areas like that where people just had to come to the conclusion that that class of algorithm can’t effectively be parallelized?

STEWART: Well, I’m sure that there are plenty of them around. You’d have to ask various people in different applications. Cleve coined the expression “embarrassingly parallel”—that the parallelism smacks you in the face. There are others that are more subtly parallel, and if you care to take the trouble, you can get enormous benefits from it. Large eigenvalue problems in which you only can compute a few of the eigenvectors can certainly benefit from parallelism, but it’s the one-sided parallelism, and it is by now well known. But I’m sure if you go around to various engineering and scientific applications, you’ll find that they have things that, unfortunately, just don’t benefit very much by parallelism, and other things that benefit immensely by it.

HAIGH: So, switching back to the Institute for Advanced Computer Studies, or UMIACS, so you joined that in 1990. That was when it was created, was it?

STEWART: Yes. It was created a little earlier, and then they sent out invitations for permanent appointments, and they made four permanent appointments, and I was one of them.

HAIGH: And that’s one of those centers that’s bringing together people from the computer science area itself and from different related disciplines and applications?

STEWART: Yes, it’s been very successful at that, thanks to two very able leaders. Larry Davis took it over first. He’s also, then, our department chairman, and he was actually a graduate student at Maryland when I came to Maryland. We played tennis together. Not very good tennis—neither of us were very good. And then when he graduated, got his Ph.D., he went to Texas, got promoted to associate professor, and then came back here and eventually became my boss in two senses. Once as head of UMIACS, and the other as chairman of the department, which is where he is now. But he and Joseph JaJa from the electrical engineering department, who was chairman. Both of these people have been extremely good. Larry was good at getting the place started up and Joseph was very good at encouraging interdisciplinary works.

For example, we have very strong connections with the linguistics department in computational linguistics, and Bonnie Dorr, who has just been promoted to full professor, is becoming president of a computational linguistic society, but I don’t know its exact name. So this has been a highly successful collaboration, and there have been others. Basically, the Institute has rotating appointments that it can give out. A number of them are earmarked for the computer science department, and these are generally three-year renewable, or they can be more for people with more substantial reputations. They’re often used to help bring in people, you know, to bring in a new person; that would give them a five-year appointment in UMIACS, and it’s generally been quite successful. As a matter of fact, too successful, because the programs have been expanding and the needs for support and everything have expanded. The budget has gone down in the various crises that have hit the state universities, and that’s a serious difficulty. But it’s a fine, well-working institution.

HAIGH: So at that point, in 1990, was this kind of institute a novel idea, or were there examples that people were looking to emulate that were already functioning in other universities?

STEWART: I don’t think it was really founded in that way. What happened was that the NSA wanted to form an institute, basically, to bring in outsiders, a computational institute that they could have outsiders working in. At that time, NSA didn’t exist. People went to the meetings;
they’d ask and you’d say, “Oh, I’m a government employee.” Now they have tee-shirts and everything. So they needed some vehicle of interacting with expertise from outside NSA, and they were planning an institute for that, and in order to get it, Maryland said, “Well, we’ll build a sister institute at the University of Maryland, and this will be UMIAC.” So I think the interdisciplinary thing just sort of emerged. I’m sure they put the words into the thing, but the original idea was to get this institute in Maryland, and they did.

HAIGH: So this would be a sister institute for the NSA’s internal group to correspond with?

STEWART: Yes, well, they’ve got something outside of NSA, outside the beltway in a science park there, and that’s the institute that the university would have been sister to. There isn’t any interaction to speak of, and it’s gone its own way, and one of the ways is sponsoring this interdisciplinary research. For example, the university makes a distinction between institutes’ centers. The institute is the bigger thing, and then the center.

HAIGH: Does an institute have to span several departments to include participation from more than one department?

STEWART: No, not necessarily. They’re each their own thing, but an institute can have several centers. For example, for our bioinformatics, the Computational Biology Center is a center and it’s under UMIACS, actually. It’s a little Byzantine, and I don’t pretend to understand it all, but it has a number of centers. It has the center for automation research in it and the computer vision laboratory, I think, is under UMIACS’ threshold or under its auspices.

[Tape 3, Side B]

HAIGH: So are there any other aspects of the evolution of the computer science department here and your relationship with it that you think we should cover?

STEWART: Not really.

HAIGH: So it seems that we can now rewind the clock and talk about all the research that you’ve been involved with since you arrived here. And unless there are some of your papers that you think we should do next, perhaps it would be appropriate to talk about the LINPACK project. There’s nothing in other areas of your research that logically proceeds that?

STEWART: No.

HAIGH: All right. Well, I think you’ve mentioned EISPACK in passing.

STEWART: Yes. I had nothing to do with that.

HAIGH: Presumably you were aware of it.

STEWART: Vaguely, yes, and became more so as time went on; but initially not very aware of it.

HAIGH: It seems that the Argonne people saw LINPACK very much as a sequel to EISPACK.

STEWART: Yes.

HAIGH: Did you have the same perception?

HAIGH: Well, no I didn’t. Let me tell you how it got started. Right after I got to Maryland, Jim Pool called and asked me would I be willing to spend a summer at Argonne, and I said, “Sure.” I was footloose and fancy free, and it sounded like a wonderful idea. So I arrived there. Jim’s very
G.W. “Pete” Stewart, Oral History with Thomas Haigh - 52

good at getting people to do things and he said to me, “We’re thinking of trying to build a package for solving linear systems, and we want to hold a meeting for everyone. Would you mind chairing this meeting?” I now know Jim well enough to know that what he meant was, “Can I get you involved in it?” And so I said, “Sure I will.” So there was a rather large meeting. I would say maybe 50 people attending, mostly from around the Chicago area and people who happened to be in there. Not much was decided right then, but the people who stayed around to work on it, we sort of decided that it was a good idea to produce a linear package.

It was not a successor to EISPACK, in the sense that we were not going to do translations. We were going to do the algorithms straight from the beginning. And this was actually on the advice of Jim Wilkinson, who said that the linear systems algorithms in the handbook series preceded the eigenvalue algorithms. We didn’t understand as much as we did by the time we were doing the eigenvalue algorithms, and it would not be a good idea to translate those algorithms. So it was immediately-- well, it wasn’t so much decided. The talk just sort of merged around people who happened to be at Argonne at the time, and among them were myself, Cleve Moler, Jim Bunch were there, and we began to discuss the ideas. Finally it sort of settled down into broad outlines of what LINPACK became. I’m sure that Jim sort of thought of it as a successor to EISPACK, but there isn’t really that strong a connection between the two. I was in a sort of unofficial leadership capacity, so after the summer I came back to Argonne and stayed in the lodging facility that they had there and spent two days writing up the grant proposal.

HAIGH: Had you been visiting Argonne before that, or was that your first involvement?

STEWART: No. That was the first time I was there, and so I wrote the grant proposal. We really had to weasel-word it.

HAIGH: You talked about the wording. In your biographical fragment, you did phrase this as, “Jim Pool invited me to Argonne for a summer. As soon as I arrived, he asked me to MC a public meeting on the possibility of producing ‘a successor to EISPACK.’”

STEWART: Yes, but it’s only in the loose sense.

HAIGH: That’s in quotes; it’s how he thought of it; that’s not how you came to know it.

STEWART: Yeah, or how it came out. There’s a certain amorphism to this, and what came out was definitely not a successor to EISPACK except in the sense that Argonne is in the business of producing this quality linear algebra software. In that sense, it’s definitely a successor to EISPACK.

HAIGH: I think you’d said earlier that during the early 1970s Oak Ridge was in decline in terms of its numerical analysis research and that Argonne was more ascendant. So when you arrived at Argonne, was there a kind of tangible difference in the atmosphere there, the way the people were approaching things?

STEWART: Oh, yes. Argonne just had this nonstop summer program bringing in people like myself and Cleve and Jim Wilkinson to spend some time there. So there were lots of people coming through and lots staying for the summer. They had a wonderful program for bringing in graduate students, which always sparks up a place. So they had a summer program for graduate students at the same time. Of course, this wasn’t Argonne-wide, the graduate student program, but many of them were associated with Jim Pool’s group. So it was just a very exciting place to be. A lot of things were happening, a lot of socializing, a lot of going into Chicago for enormous free dinners, and just generally and technically exciting, too. It was just the place to spawn
LINPACK. So, in any case, LINPACK had to be sold to the NSF, which was against software development at the time. EISPACK had had something of the same difficulties, and so the grant was written in such a way that the main purpose of LINPACK was to do research in how you develop this kind of software.

HAIGH: Yes, the National Activity to Test Software

STEWART: I don’t get the reference.

HAIGH: It may have been the name of the EISPACK grant, but I know that Jim Cody was one of the authors of it, and the rationale for one of these pieces of software to getting the grant was called, I think, the “National Activity to Test Software”, and that’s an NATS project. I think that was EISPACK. But probably they already had got one grant with that storyline and--

STEWART: Ours was research and how to develop software. It’s somewhat different emphasis than testing it. In other words, we were going to show how you got people together.

HAIGH: That’s true. It probably was EISPACK that was called testing. The rationale was that the software would be “certified,” I think. It was supposed to be a kind of quality control.

STEWART: Yes, and here the emphasis was basically, at least as I wrote the grant, was on, “Let’s do this project and see how it’s done, and we will report our experiences,” and this sort of thing. I and I think everyone else considered it a dodge. We wanted the money to develop LINPACK. But it’s not that an awful lot wasn’t learned and fused through the community. In some sense, we did accomplish that goal, but it wasn’t the primary one in any of our lives.

HAIGH: What did you claim would be novel about the way in which the software was developed?

STEWART: I forget exactly what I put down in the boilerplate, but as it turned out, there were a number of novel features. Perhaps most novel was the use of software to format our programs, so that we could have one master complex version of the programs and then have them automatically generate the various versions for various precisions and this sort of thing. Another thing was that the BLAS and LINPACK fed off each other. The BLAS were being proposed at that time, and then the question was whether LINPACK would use the BLAS. My first reaction to almost anything is no. I know enough not to listen to it, so when I first heard of the BLAS I decided that I’d better code up something using them and see how I liked it. So I wrote a Householder least squares type program in it, and I found that, once you got the rhythm of the writing in this style, that it was easier and that in some cases algorithms were more efficient. So I became a convert, and so we decided to use the BLAS.

HAIGH: And do you know how you first became aware of them?

STEWART: I think it was flying with Virginia Klema on an airplane. I think she had mentioned them. But I didn’t really see what they were until they began to impinge on LINPACK and we had to make a decision on it. The decision was to do it, and I think it was very much the right decision, but it also meant that the BLAS had a major package that could support them or help them that we’ve been used in this major package.

HAIGH: So I understand that Richard Hanson and Charles Lawson were responsible for the Level 1 BLAS.
STEWARD: Yes, and others. The papers have several authors on them. I wasn’t part of this loop at all.

HAIGH: Would any of the core LINPACK people have been dealing directly with Hanson and Lawson?

STEWARD: No, not in the sense of working on the BLAS themselves. I mean, we all knew each other and this sort of thing. But our contacts were, basically, “We’ve got this thing that’s coming up out there. Should we use it?”

HAIGH: So you would read their papers describing it but you wouldn’t invite them to come to a meeting or have a joint session or anything like that?

STEWARD: No, no. It’s a little amusing. I don’t know that it has really affected anything, but at the time they were arguing over whether it should be the BLAS or the BLAMs, Basic Linear Algebra Subprograms or Basic Linear Algebra Modules. At the time, Alka-Seltzer was having ads for curing the blahs, you know, “I have the blahs, and take Alka-Seltzer,” that sort of thing, and so this was very popular. So I think one of us said something like, “Well, we’ll do it, but let’s bargain that they call it the BLAS so we can say LINPACK has the BLAS.” [Chuckles] I don’t know whether that had anything to do with whether they decided to choose the name, but we certainly were joking about it.

HAIGH: And are you aware of any tangible difference that the LINPACK project and its requirements had on the design of the BLAS itself?

STEWARD: No, I don’t think so. By the time we came in, they were pretty much fixed.

HAIGH: Okay, so let’s move back to LINPACK itself, then. So you’ve mentioned Cleve Moler and Jim Bunch and Jack Dongarra as the main people.

STEWARD: Well, Jack initially was basically a systems support person. He was not involved in the design of the package or this sort of thing, and only later did he begin to take an important role in it. But in the first part of it, he was not part of it. In fact, reading Jack’s comments on Cleve’s and our go-arounds… [Jack Dongarra, Oral history interview by Thomas Haigh, 26 April, 2005, University of Tennessee, Knoxville TN. Society for Industrial and Applied Mathematics, Philadelphia, PA]. Jack was somewhat out of the loop in this stage. It’s a legend at Argonne that Cleve and I had some real, I would call them shouting matches, but, you know, loud words. But you have to understand that, as I say, we’re both persons of strong opinions and this sort of thing, and it was not about mountains or molehills. These were serious issues, even if some of them don’t sound like it.

For example, the nomenclature—that sounds like a silly thing to argue about how you’re going to name your routines. But in point of fact, in those days, remember, FORTRAN only had six characters in a subroutine name available, so we had to figure out some kind of way of coding the maximum amount of information into this. You can have very different opinions on that. For example, just to give a difference between Cleve and me, I’ve always believed that you should maximize consonants in a name because it contains more information than the vowels, and Cleve is a euphonious person who likes to put in vowels to make it pronounceable. So here we had things like this were there. So sure, we communicated by being loud, but we communicated and the compromises were reached, and in this particular case, it was good enough so that it was taken over wholesale by LAPACK later.
HAIGH: The naming the conventions?

STEWART: The naming conventions. So we did end up doing something right. One thing that was not controversial, but originally, as we were designing it, there would be an error return which would be IERR, typically, in almost any program—the “I” because it would be an integer, and then the “ERR” for error. I suggested that in many cases these things were not really errors but information on how the program had done, and so we changed it to “INFO” and I think that was a positive thing. So the arguments that we had were actually real, about important things, and they were not counterproductive in the sense of two people clashing and not giving in. Another thing that I pushed very hard for was to make the LINPACK users’ guide have an educational function and not only to just describe the usage of the algorithm, but to derive the algorithms and explain the coding details that went into it. So there are two ways of building—the minimal user’s guide just gives you a calling sequence and explains how to use the program. The way it finally turned out, we have an intro in which background is sketched, then usage. The intro should contain enough information so that the calling parameters do this, and then there were programming details. The introduction would also have the mathematical derivation of what you were doing, and then a detail thing on programming details so someone who wanted to get into the algorithms didn’t have to read raw code to do it, and finally give numerical examples. I think this turned out to be a great success. At least the LINPACK user’s guide was a SIAM bestseller for some years. [With J. J. Dongarra, J. R. Bunch, and C. B. Moler], LINPACK Users Guide, SIAM, Philadelphia (1979).

HAIGH: Then would you say that you were the main author of the user guide?

STEWART: No. I wrote the introduction, but each of us wrote our own sections. I just pushed to have a very full discussion of each algorithm as it came up. By no means am I the author of the LINPACK user’s guide.

HAIGH: So the work was fairly evenly distributed?

STEWART: Yes.

HAIGH: Okay, so let’s pull back again to the big picture. Now, while you’re on the topic of arguing with Cleve, you got onto that in the context of explaining that there were some issues that it was really important to come up with answers for. You’ve mentioned that the changing errors to information was not controversial but the naming conventions were; and you pushed through the idea that the user guide should be more of a tutorial and less of a test reference. Were there any other things that were controversial about the design or implementation of LINPACK?

STEWART: Not really. I don’t think there was an awful lot. Oh, I’m sure that we went round about some minor details, but I completely forget them. Once the overall design was decided each of us had our responsibility for certain areas.

HAIGH: All right, so how did the responsibility break down between you?

STEWART: Well, I was responsible for things having to do with the QR decomposition, least squares, and singular value decomposition. Cleve worked on the things relating to Gaussian elimination, and Jim Bunch worked on stuff relating to symmetric indefinite systems. Basically, each of us went back and coded up our routines, and then I believe it was in 1976 we all brought them to Argonne and started testing. During that time some of the conventions were also decided. We didn’t have uniform conventions in naming at that time, but that was easy to do.
HAIGH: So that would have been one year after the beginning of the project that project?
STEWART: I think, yeah.
HAIGH: Was there ever any controversy over what the scope of the package should be?
STEWART: Well, only at the very first. It was more a matter of did we want to include such things as sparse matrix algorithms and this sort of thing. It generally came down to who was at Argonne at the time. Actually, it would have been premature, probably, to include sparse matrix algorithms in LINPACK at that time. The subject was not mature enough. So in that sense, one could obviously have proposed other things that you might want to include in LINPACK, but what finally came in there was pretty much the core of dense linear computations and pretty natural.
HAIGH: Now, how about Argonne’s institutional role in the project? You had mentioned that Jim Pool had the original idea to invite you there and propose this meeting, and you were physically going to Argonne in the summer. Was Cleve Moler also coming to Argonne for the whole summer?
STEWART: Yes, and so was Jim Bunch, and Jack was, of course, there.
HAIGH: So what was Jim Bunch’s institution affiliation?
STEWART: I think UC San Diego.
HAIGH: So it was an Argonne project, but none of the main people were Argonne--?
STEWART: None of the people were with Argonne, no. EISPACK also had a large number of non-Argonne people. Cleve, for example, was not Argonne.
HAIGH: So this was just a continuation in the same direction?
STEWART: Yes, except perhaps more so in the sense that they were more dependent on external things. Of course, Jim Wilkinson came every summer and was sort of an advisor.
HAIGH: So what was Wilkinson’s involvement with the LINPACK project?
STEWART: Well, as I say, as an advisor.
HAIGH: So he would be around, but mostly he would be doing other things.
STEWART: Oh, yes. No, he had nothing to do with the actual coding or anything.
HAIGH: But if you had a question, you could go and see him?
STEWART: Oh, yes. And as I say, I think his greatest contribution was to tell us to forget about the handbook series and strike out on our own.
HAIGH: And other than physically being a place where you came to work, was there any involvement from the Argonne management? Did you have to write progress reports for them, that kind of thing?
STEWART: No, there was nothing of that kind of thing. We were rather free in our researches. I mean, we all became part of the Argonne crew in Jim Pool’s division. I made good friends there. Hans Kaper, Gary Leaf, Jim Cody and others. People that I’ve been very proud to know. And the support was wonderful. We had no difficulty. As I say, Jack’s original role was systems support for us.
HAIGH: And how did that change over the course of the project?

STEWART: Well, he became more involved. As I say, he only wrote one rather small code for it, but when it came time to actually decide on things like sequence of authors and this sort of thing, we just sort of decided that Jack was showing exceptional promise and that we might as well give him a leg up and put him as the lead author, and then alphabetically after that, and I think we all feel that that was a very good investment. And, of course, he immediately then turned it into the LINPACK Benchmark. So that’s the story there.

HAIGH: How do you feel about the LINPACK Benchmark?

STEWART: Mixed, but I think that it at least made people think about comparing these machines, at least, on something that was real. Whether that is the right thing to compare it for someone doing, say, quantum chromodynamics or some other application is not clear, but at least it provided food for thought for people who wanted to try to compare these various machines.

HAIGH: So you’d said that the project started in 1975 and you bought together the first code in 1976. When did you first start letting people attempt to use it?

STEWART: Well, what we did was I first had to get the code in shape. This meant using the Jim Boyles TAMPR system to reformat the code. This was nice because it would take in any code that you gave it in FORTRAN 66—that’s what we were working in at the time—and it would automatically reform it, trying to reveal its structure. This is one of the reasons why I say I think Cleve is a better programmer than me. As I mentioned earlier, I took very seriously the structured programming thing, and it already had very structured code with explicit indentations, et cetera. Cleve came in with his code and it looked like pure spaghetti code—the FORTRAN numbers were not in order, you’d have a 10 and a 100 and a 5. It just looked awful, and little meaningful indentation. The Tamper system took it and it came out looking clean as could be. In other words, Cleve, without having to think about it, naturally coded in a structured manner, and it was just real interesting to me. Now, I’m not at that level. I have to think very hard about how to structure my codes, but Cleve is a natural.

HAIGH: So you had this tool that would format the code nicely?

STEWART: Yes.

HAIGH: Now, were there other aspects of documentation or organization of code where you had to define standards within the group?

STEWART: Well, that was part of the things that we were talking about with the naming and this sort of thing. There were many minor decisions which I’ve completely forgotten about, but in the end what we ended up with was master complex code and complex arithmetic that would automatically be changed into real single, real double, complex double, and complex single.

HAIGH: And was that done by TAMPR too?

STEWART: That was by TAMPR, yes. So we had that automated. Then, as far as testing, we created test suites.

HAIGH: And Jim Boyle was responsible for that?

STEWART: Yes.
HAIGH: So did TAMPR already exist, or was that something else that was evolving in conjunction with LINPACK?

STEWART: Well, it existed well enough to take our code when we came in 1976, and TAMPR, I think that there may have been improvements; I’m not sure. I didn’t follow that aspect of it that closely. As long as it was working, I didn’t inquire.

HAIGH: Okay, so that was something that you took advantage of, but that wasn’t specifically produced to support this project?

STEWART: No, no. This was part of Jim Boyle’s own research into reformatting code.

HAIGH: Okay. Sorry. I cut you off there as you were launching into another topic.

STEWART: Well, you actually had brought up the business of getting it out to people for testing. So we did build comprehensive test suites and then distribute them to people. Here’s another case where the Argonne administrative stuff handled things like that for us and we didn’t have to worry about actually the physical mailing of code or sending of tapes and this sort of thing. This was all done for us. We sent it out to a number of test sites, and actually there was a mix-up here because Cleve had originally had the conception of actually distributing the test code with the package, and I had seen it as a thing that we were using to get other people to test it. He wrote beautiful test code and I wrote code that was perfectly good, but it was sloppy because it was not supposed to go out with the package and I was a little nonplussed when I discovered that it was. But it was entirely my fault, and it was the right thing to do, to do this, and LAPACK has followed suit in doing that and having extensive test packages going out with the code when you download LAPACK, at least in its entirety. You get a whole bunch of test code.

HAIGH: That test suite would let people test that they’d installed it correctly and it’s working with that particular machine, configuration, architecture, and so on?

STEWART: That, I think, was the ultimate idea. In my conception, which was somewhat more limited, this was to circulate around so we could find out what machines it wasn’t working on and make sure that they were, but that was the extent of it.

HAIGH: Now, before you started worrying about machines that it would and wouldn’t work on, was there an earlier stage where you would release it internally within Argonne so you could see if it worked on their machines?

STEWART: Occasional people would try it, and we found some bugs that way, but no, not wholesale. It could only be done in the summer because the people around to fix it were only there in the summer.

HAIGH: So there was no point where a test version of the whole package would have been made available within Argonne prior to it being made available externally?

STEWART: No, no. It went immediately external. And I don’t think we would have gained that much. An occasional person might have tried to use an occasional program, but by sending it to various other installations and institutions and asking the people to test the whole package, we got a lot more information out of it than we would have gotten releasing it.

HAIGH: And when you got that feedback, how many bugs did they manage to find?

STEWART: Not an awful lot. There was a bug that no one found in my SVD code, mainly because it didn’t make much difference, but we had to correct it later. I think maybe there was a
computing error, a shift in an iteration that had the wrong formula. But, as I say, it didn’t make any difference because things were converging at effectively the same thing.

There was one interesting thing. Some of the least-squares code failed on VAXes because the floating point exponent and double precision was too small. This was one of the big arguments going on between VAX and Vel Kahan, the thing about the exponent size, but in point of fact, there was nothing we could do about it. We were pushing the extremities, and it turned out that anyone with a decent-sized exponent would work, but it wouldn’t work on VAX machines in these extreme cases. So we discovered something, but I wouldn’t say that it was a flaw in the LINPACK code.

HAIGH: I asked this to other people, but I’ll ask it to you as well just for completeness. With today’s open-source projects, one of the theories is that users will not just spot bugs, but, in some cases, may be able to fix them themselves. Did you ever have any cases where anyone would write in with a suggestion for a better way to code something or would include a possible fix for something together with an error report?

STEWART: Not really. As I say, for example, the error in the shift was detected, but by and large, LINPACK was released on a take-it-or-leave-it basis. We all went on to our various things. Cleve, at the end of LINPACK, was actually preoccupied with MATLAB at that time, and so it was very much like EISPACK, too. EISPACK was not supported in the sense of detecting bugs.

Regarding the Open Source idea, no, it’s not really the same thing. I mean, these are highly technical algorithms that it would require a real expert to dig into. Some of the simpler ones don’t, but some of the decisions are really critical, and, unlike some kinds of systems program, you can’t just get in and read it and say, “Oh, I can do this better,” like that. You would need a good deal of background to be able to do it, and the code can be quite difficult. I’ve learned later some of the code that I coded in LINPACK ended up in LAPACK just in exactly the same form and I asked someone about it, and they said, “Oh, we just took it over because no one wanted to tamper with what you’d done.” They could have pieced together what I was doing, but it was fairly delicate. So this kind of stuff doesn’t fit the Open Source model very well.

HAIGH: Which brings us to a related topic. So, obviously one of the reasons for that would be that you were already trying to gather the best methods available and do a quick implementation of them. Within your areas of the package, where did you go to find the methods and algorithms that you were implementing?

STEWART: Well, it wasn’t difficult; we just were conservative. We’ve looked around and we knew what algorithms were good. There were rounding error analyses for them and this sort of thing, so we’re conservative. We do not try to push the envelope with algorithms. In that sense, we were really doing research and developing software, because we did not try to develop new algorithms. The nearest thing, I would say, that came to it was that the indefinite systems stuff has been improved on later. It was still a little researchy, but these are perfectly adequate programs and will do for the job. Well, I coded a program for downdating QRD compositions or Cholesky decompositions, which is a very difficult problem. And that was definitely a researchy type of thing, but it was necessary to have one, and I got help from Michael Saunders at Stanford, who’s one of the few people who does rounding error analysis in this world, and later on there’s a paper that discusses that that came right out of LINPACK.
HAIGH: So that was an example where the needs of the package made you realize that there was this little gap in the research, and then you had to fill that before you could responsibly incorporate a method.

STEWART: That’s an extreme case. Mostly we stuck with the tried and true. That doesn’t say it wasn’t fairly new. I mean, all of the least squares was developed from that Householder QR that Gene Golub had published in 1965, so we’re talking about 10 years old. That is not old when you consider that Gaussian elimination came in 1809.

HAIGH: Were there any cases where there were important methods that you didn’t include but with hindsight you could have incorporated, that had already been published but that were too new, not proven, you hadn’t done the analysis of?

STEWART: Not really. I mean, LAPACK has expanded and done a lot more, but I think we covered the basics rather well. The one thing that I regret—and there’s no way we could have known, we would have had to have been blessed with hindsight—but we were in a unique position to invent the BLAS II, there. As a Monday morning quarterback, we should have seen a lot of times that we were subtracting rank one matrices from our matrices and put that in the BLAS. If we had LINPACK would have run on Crays very fast, because that operation could be fine-tuned for the Crays in machine language because they’d be in a subroutine in a BLAS. But we didn’t see it. The Crays were just coming out after we had finished our code. We were hearing rumors from Los Alamos that they were solving these 500x500 linear systems in incredible speeds, but we didn’t know, really, the structures of the things. So in retrospect that was a failing, and, of course, LAPACK does have all of this stuff.

HAIGH: All right, but at this time the Level II BLAS had not yet been proposed?

STEWART: No, no.

HAIGH: So your regret, in a sense is that you didn’t--

STEWART: Monday-morning quarterbacking, yeah. But I mean the point is, if we’d had a good enough aesthetic sense, we would have--

HAIGH: You mean if you’d taken this structured programming idea--

STEWART: No, it’s not structure programming. If we had looked at the operations that we were doing, we would have suddenly seen that this operation was repeating itself constantly throughout large parts of the package, and if we’d put that in a BLAS, it would have been part of BLAS II, and we would have had it and it would have been aesthetically the right thing to do. Aesthetics plays an awful lot in these things.

HAIGH: Wouldn’t imposing an additional level of abstraction and having something be a subroutine instead of being repeated many times in the code be a kind of structured programming?

STEWART: Yes and no, but not in the original sense of it. It does say modularize, but if you modularize it to too fine a level, you have the inefficiencies of calling a subroutine to do too little, and so there’s a fine point there. I mean, you could say that, yes, you should modularize, but we just never really made the association that we were doing this common operation so frequently.
HAIGH: So you were returning every summer until 1979, and at that point, did you stop going there because LINPACK was finished?

STEWART: Essentially because LINPACK was finished. I was married and we were starting to spend summers in Maine. I mean, it helped not to be married. Cleve had been recently divorced, I believe, or was not married at the time, and so just getting away and spending all summer was not a thing I could do for all my life.

HAIGH: Yeah. And according to your resume, you continued as a consultant until 1983.

STEWART: Yes, I would come back on occasions.

HAIGH: Was your consulting work on different topics?

STEWART: Not really. Basically, I’d come back and they’d allow me to do what I wanted to. I’d interact or see people, but it wasn’t serious or directed.

HAIGH: And since then, have you had much interaction with Argonne?

STEWART: No, not really.

HAIGH: Now, you’ve already talked a bit about the LINPACK user guide; you’ve talked about your insistence that it should be more tutorial. Just a follow-up question on that would be, have you had any feedback from the users themselves? Do you have a sense of whether it was widely taken up for that purpose? Do you ever hear from any happy readers?

STEWART: That’s actually one of the more interesting things I’ve heard. I can only judge, really, by its success, and people have told me that the LINPACK is a nice thing, but I don’t know that they’re referring to the guide or not. But when LAPACK came out, someone sent over a message. “Well, it’s a good package and everything, but hang on to your LINPACK user’s guide if you want to know what’s going on.”

HAIGH: Any other stories or comments on that?

STEWART: Not really, I guess.

Beginning of Session 4, being held on the second day of interviewing, May 6, 2006, again at Dr. Stewart's house in Washington, D.C.

HAIGH: Yesterday we had discussed the LINPACK project itself, the work that was carried out, the process that you used, the areas covered by the various contributors. So it might be appropriate to begin now with a general question about the impact and significance of the LINPACK project.

STEWART: Well, I think it had very great significance within the area of numerical linear algebra. I also think that it had great impact in the sense that the package was actually used. I mean, I don’t have any statistics to support that. Things weren’t downloaded the way they were in those days, so many people must have either gotten the programs from other people and then through channels. There was some sort of distribution channel at Argonne, but I’m not sure the details of that.

HAIGH: Is that the Argonne Code Center?
STEWART: That I don’t know, but it was distributed through Argonne in some way or another. But I’m sure just lots of the programs simply passed as card decks originally and then later through the network and this sort of thing. It was only when NA-Net was set up and Netlib was set up that these things became trivially easy to distribute. I’m sure you’ve talked with Cleve about NA-Net and Netlib.

HAIGH: Yes, and Jack Dongarra.

STEWART: Yes. So I think that it’s had a pretty big impact. It certainly had an impact within numerical linear algebra, because as I mentioned, a lot of our conventions and our basic formatting and this sort of thing was adopted later by LAPACK, which was a complete remake of both areas, the EISPACK and the LINPACK. I had no part of that.

HAIGH: Why was that?

STEWART: I suppose that I could have said, “I want to be a part of it,” and they would have, but I just felt that I had done my thing and it was best to let other people have their chance at it without having the dead hand of the past on them, so I just sort of stepped aside and let it go. I think Cleve must have done much the same thing. He was not actively associated with it. On the other hand, Jack very much was and, along with Jim Demmel, was a leader in the development of the package. They certainly have produced far more code than we did, much of it needed and very good. I’m a little sorry that they just produced man pages for their user’s guide, but once Jim quoted to me, I said that and he said, “Well, what you rather do? Write a book or put more programs out in the world?” There are arguments on both sides. I happen to think that maybe taking some time out to writing the book might have been a better thing, but I can certainly see Jim’s point.

HAIGH: And do you think your view would be a minority position within the numerical software community?

STEWART: Well, I think so, but not for reason of just simply one view or another as to how one should do these projects. But I just don’t think that the others shared quite my enthusiasm and Cleve’s enthusiasm for writing. If the writing is not an important part of a project to you, then you tend to skimp on it, and this certainly increases the amount of writing that you have to do. So you have two perfectly defensible points of view, and I think that one follows one’s inclination when writing a project. You follow the point of view that suits you best.

HAIGH: I think you also mentioned that you had had a conference paper about the LINPACK project.

STEWART: Yes, and it’s called “Research, Development and LINPACK”. It appeared in a book that John Rice edited.

HAIGH: Oh, yes. I see. It was Mathematical Software III.

STEWART: Yes. Really it was a continuation of our argument that software development not only was a research tool, but I extended it to say that it also suggested research topics in its own right. Quite frankly, I haven’t reviewed the book myself, and don’t want to try to remember all of the topics that I mentioned there. But one important one was scaling. It turns out that algorithms like Gaussian elimination, and even methods based on orthogonal transformations like Householder’s and Givens’ transformations, can be very sensitive to the scaling of the matrix of its columns and rows. This was then and still is, a very imperfectly understood area. It certainly
affected some of my further research, because I became much more sensitive to these problems and the behavior of scaled matrices, and I’ve written some papers on it as a result. So that’s an immediate impact on me.

HAIGH: And was the closest that the LINPACK project came to generating a publication on what was its ostensible research area for the grant purposes, this software development technique?

STEWART: Yes, essentially. Because I think that by the time LINPACK actually came out, it was pretty well accepted that high quality people were doing software work, and it was worth funding. And we didn't have to jump through these hoops any more. LAPACK had no difficulty getting funded. It was just a turnaround. It was like finally mathematicians, some dragged kicking and screaming, agreed that the computer could form a legitimate part of a profitable exchange, and by the time LINPACK had appeared, I think the tide was going in the direction of agreeing this is being a legitimate area to be funded by federal funds.

HAIGH: You mentioned also that the influence of the LINPACK experience on your research in terms of an increased appreciation for scaling. Are there any particular papers that you had mentioned as examples of that?

STEWART: Well, yes. There was one paper on downdating.

HAIGH: Would you have a reference for that?

STEWART: Yes, “The Effects of Rounding Error on Downdating the Choelsky Factorization” [“The Effects of Rounding Error on Downdating the Choelsky Factorization”, *Journal of the Institute for Mathematics and its Applications* 23 (1979) 203-213].

One thing we haven’t talked about much, actually, is rounding error analysis. That’s another area that I had worked in fairly extensively. It’s not an easy area to work in. In other words, doing a good rounding error analysis involves a great deal of meticulous detail. I often tell people that the way you do a rounding error analysis is that you stare at an algorithm until you understand everything about it, and then you quickly do the rounding error analysis, get it written down, and then you never look at it again because you won’t understand it when you see it the second time. It really requires real concentration on every detail of algorithm. At it’s best, though, it shows you exactly where the algorithm is weak and where it is strong and how robust it is under certain conditions. But for a significant algorithm, it’s not awfully easy to do.

In any case, the particular algorithm that’s the subject of this paper, is what its called: downdating a Choelsky decomposition. The word “updating” has been in the literature for a long time. I mean it’s a natural English word, but has plenty of non-technical meanings as well as technical. But technically it means making some small, or not-too-great, modification to a matrix and then recomputing something associated with the matrix, say a decomposition or some other thing. So for example, linear programming people proceeded by updating. From the very first, they would find a new point on the polytope over which they were minimizing their function, and then they would have to update bases for the thing and usually, in this case, an inverse matrix that they were using to calculate the next step of the linear program. So updating goes back quite some time. In fact, you can even trace it back to Gauss if you wanted to.

But this particular problem was something I put in LINPACK, and it’s something of the exception of not doing “researchy” stuff for that package. I mean, of not pushing the envelope technically. If you wrote down a table of what programs belonged in this area, this particular one
almost had to be there. So I devised a program. It turns out that actually Michael Saunders had also devised the same algorithm before me, and it’s come to be known as the “LINPACK Algorithm.” It’s an algorithm that’s known not to work under certain circumstances. We always believed that the reasons it didn't work were intrinsic to the problem, not an artifact of the algorithm itself. If this algorithm didn't work, then any algorithm would fail to work, no matter how it was done. “Downdating” was actually my word, and my neologism had spread throughout the community. Essentially, this rounding error analysis showed that, that process of downdating the Cholesky decomposition has a different kind of stability than numerical analysts had been used to dealing with before. What that basically meant was that it was intrinsic to the problem itself, rather than the thing. And this area has gone on.

Actually, the algorithm in LINPACK was not the best. There was a better one that’s somewhat more stable, but still in this unusual sense, and that’s the one that’s currently used by most people.


STEWART: Yes. Yes. I’d almost forgotten about that. Thanks for reminding me. Yeah, that was a really novel thing that LINPACK did. It had its origins in a paper that I wrote with Bill Gragg. We were doing an algorithm for solving non-linear equations.

HAIGH: Do you know roughly what year that would be?

STEWART: Yes, I think I can find it. [(with W.B. Gragg, “A Stable Variant of the Secant Method for Solving Non-Linear Equations,” SIAM Journal on Numerical Analysis 14 (1976) 889-903]. So that pretty much says it. The “stable” part came from the fact that we were able to approximate what’s called a null vector of a matrix called the Jacobian matrix within the algorithm. The trick was sort of nice. We were very happy with it, and it worked. It was one of those things that you can’t prove works, but does work almost all the time, and we did it. I realized when we got to doing LINPACK, that this same technique approach could be used to tell when matrixes were ill-conditioned; that is, when the linear systems that we were solving, would almost certainly be solved inaccurately, no matter how good your algorithm (another example of an intrinsic limitation of computing with rounding error).

HAIGH: Would the idea be that the user would then be warned that they couldn’t get a good answer.

STEWART: Yes, exactly, to warn that they were dealing with an ill-conditioned system, and that they really needed to go back to the drawing board and look harder at how they were defining their problem.

HAIGH: Yes. Presumably, that would be a significant advantage of using this software, rather than trying to go to the textbook and code their own; that they would not gain that knowledge with the kind of amateurish implementation?

STEWART: No, they would not. In principle, the condition number is easy to compute, but if your matrix is order N, then the naive way of doing it requires order N cubed operations. The algorithms that we proposed, while as I say, not guaranteed to work every time, in practice do, and make the required order N squared operations. So over the operations that were already done, this was insignificant amount of time to get this information out. And in any case, I
suggested it, that we do this. Then Jim Wilkinson and Cleve Moler joined it. We found counter examples, or rather naive algorithm and slowly over a course of roughly a year, and proved it, during which time, incidentally, Allen Kline also found another counter example and we sort of incorporated that. So that’s why he’s also an author in the paper, and published it. Then the topic took off for a few years afterwards, people producing different kinds of condition estimators. And it’s settled down now, but there are still a number of very good condition estimators. I actually prefer to call them “null vector estimators,” but the “condition estimators” name really stuck, but they all essentially produce a vector that is almost mapped to zero, like a matrix in question. And that often is what you really want.

HAIGH: So then those two papers you think would be the main aspects of your own research output that could be linked to the LINPACK project.

STEWART: Yes, indirectly. Obviously, I must have been influenced in many ways by that, because it was an intense, very satisfying experience.

HAIGH: Now, do you think we should talk now some more about your other research, or would it be appropriate now to talk about your involvement with professional associations such as ACM and SIGNUM?

STEWART: Why don’t we go through the professional stuff, because I think it can be handled in fairly short order.

HAIGH: . So do you think it would make sense to start with the Gatlinburg meetings?

STEWART: I’m sure. Actually, that won’t be in such short order [chuckles]. I had forgotten about that.

Well, you’ve had plenty of the history of the Gatlinburg meetings, and I don’t really need to say a lot more. Previously before I even came involved with them, they were of course, started by Alston Householder. Actually, once they were started, he held the first one in the town of Gatlinburg, which is called The Gateway to the Smokies, and even then, was a glorious tourist trap that you had to go through to get into the Smokies, at least from one exit. But it was a nice place to have a meeting.

HAIGH: And that would be about an hour’s drive from Knoxville, wouldn’t it?

STEWART: Yes. Well, depending on how you came. It was somewhat longer from the airport, but a much more scenic and pretty drive if you came up through Marigold from the airport. But yes, it was reasonably accessible to Knoxville, but not accessible to much of anything else. But there wasn’t really much of anything else around for it to be accessible to, at that time. And he started these. I forget the year, but they were started. After the fact, people talked about other meetings as Gatlinburg zeros, and even Gatlinburg minus-ones. But I think probably it’s best to say that the first one was the one that Alston organized. Alston was very good at that and had the resources of Oak Ridge National Laboratory behind him. So he had all the secretarial and administrative support he needed to organize these meetings, and that’s one of the reasons I think they became so popular. Originally, they were organized actually just as research meetings, and only the people who were actively researching in the area were invited. And it was by invitation only, by a rather small panel of people who Householder called and said, “Who should we invite?” This was the subject of problems later. The first Gatlinburg I attended was in 1969, which was also Alston’s retirement, Gatlinburg.
And actually, there had been a fuss in the previous one, because George Forsythe wanted to bring Cleve Moler to it, and bringing a student to this meeting of researchers was not to be done. This was, of course, not the way of the future, and now we try to encourage as many graduate students as we can to come to the meetings. But at that time, it was something quite new and quite unexpected. And I don’t think you should blame these people as being old foggies. They really had set it up as research meetings where the best people in the field could communicate with one another; they genuinely felt that there wasn’t a place for students who were still learning there. Now times change, and it’s worked out. And it must be admitted that the small committee approach eventually had to go.

So I came there in 1969, after I had gotten my degree. I was basically a passive participant. Originally, people just gave talks, but it was beginning to split off into evening sessions and people would have special sessions that weren’t plenary to discuss certain areas. That developed quite significantly as time went on. So I forget when it was; it was pretty quickly. I think it may have been at the Asilomar meeting, I don’t know. But two or three meetings afterwards, I was put on the Organization Committee.

HAIGH: According to your resume, you were a member of the Executive Committee from 1978. Is there another committee as well, that does organization, or is it the same one?

STEWART: No, no, no. But after Alston retired, the meetings started floating free. It was still called the Gatlinburg Meeting, but the next meeting was at Los Alamos, and one person of the committee would take charge of it, but they would usually find a local person to do arrangements, since it was not necessarily held in the same place as the person’s home institution. That was certain the case in both of the ones that I organized.

HAIGH: Right. So that was for the 1987 and 1999 meetings?

STEWART: Yes. I was at Maryland, and then one was in Tennessee and the other was in British Columbia. Each meeting was its own individual case, but a general pattern began to emerge of one member of the committee would take on the responsibility for organizing the next meeting, that would be announced, and the executive committee would generally decide where the meeting was going to be held at that meeting. So on the last day, or during the banquet and celebration, they would announce who was handling it and what was going on. Usually, there would be a person who was vocal on the site to handle the details of actually doing the meeting. So it was a division of responsibility. The person on the executive committee supervised balloting on who would come and this sort of thing. We would want to get into that, I think, a little later. But then the local organization would worry about hotels and to some extent schedules. That was usually hacked out between them, according to the facilities and the rooms, between the local and the executive committee member. This has been, for me it was always a very nice relation. In the Fairfield Glade meeting in Tennessee, Bob Ward, at Oak Ridge National Laboratory organized it and did a wonderful job. And in British Columbia, Jim Varah organized the meeting, and again, did a very nice job. So that was how things were organized.

HAIGH: So going back to that first meeting in 1969. It said that you were very quiet. Other than that, what were your impressions at the meeting?

STEWART: Very exciting. These were people who were doing really good work. Attending it was a little like when I attended the first Michigan summer conference. But now I was interacting, and talking with people.
HAIGH: So were there any people that you really talked to for the very first time at that meeting with whom you had continued the relationship with later on?

STEWART: No. I’m sure I must have met Cleve at that meeting. But I felt myself, at least, very small deer. You know, I kept a little to the sidelines. It was at later meetings, that I began to have more interaction with people and this sort of thing. It was a gradual thing. I always look forward to the meetings and almost invariably benefit technically from something that I hear or learn there.

HAIGH: Now at that point in 1969, had you been to many larger meetings?

STEWART: Well, I’d been to some, but not an awful lot. Remember, this is one year after my Ph.D. so I had not had an opportunity to go to a lot of meetings. Later on, of course, I went to my share of them. I’ve never been one to go to a meeting a month or anything. I get very tired from that too much meeting hopping. To tell you the truth, for many years, they stimulated me so much that I really had no need to. I won’t say every meeting, but was almost tantamount to my coming back with an idea for a paper or something, you know, or for some line of research or this sort of thing. So too much could have just been overload. Still, I went to a lot of the SIAM meetings and especially I enjoyed when the national meetings were smaller, when SIAM was smaller and the national meetings were not specialized, so that you could talk to a lot of people over a wide variety of areas. I’ve always felt a little unhappy about the fact that SIAM is split off into these special groups and you don’t get to see as much of people. I don’t know. I mean, again this is one of those things that you can argue it both ways, but something was certainly lost as things like the linear algebra group and the optimization group split off and began having their separate meetings.

But back to Gatlinburg.

HAIGH: Yes. So did Householder himself continue to attend the meetings after his retirement?

STEWART: Oh yes, he attended regularly.

HAIGH: And how did your personal relationship develop? I think you said that as a graduate student you were a little bit shy and…

STEWART: Diffident, yes. Well, I mean, it developed very nicely. I mean we never really worked together, after, but we had a couple of papers together basically on polynomial research, which was his interest at the time, and which I realized was something of a dead end. But, you know, socially, we were fine. As a matter of fact, he came to my wedding. I actually spent time at his house in Malibu, when he moved out there. So you know, it was a very comfortable relation, and he is a wonderful person, I both respected and I liked him very much once I got over my own diffidence. So it was very nice. The last I saw of him was when he was very feeble and he came to the Gatlinburg meeting at Lake Arrowhead. As I say, he was extremely feeble at that time, and I had to get him through the airport to transportation and this sort of thing. Then he died a few months later after that.

HAIGH: Other than the shift you’ve already mentioned where the meeting started moving around, and that you alluded to that more graduate students were attending…

STEWART: Well, what happened was, by the time…it must have been by the time I got to Maryland, because I remember reading a review of an application for support of the next Gatlinburg program. So I was definitely at Maryland at the time, so this would be 1974 or later.
The meeting came under considerable attack from outsiders as being elitist, and controlled by a small group of people who were not allowing new people in and this kind of thing.

HAIGH: By “outsiders,” do you mean people outside the field, or people inside the field, who weren’t invited?

STEWART: Well, some outside the field and I think, some inside, who were not invited. I read a review for an NSF proposal which basically stated these terms in a rather cutting manner. I think they may have been funded by the NSF that time, but they were told, basically, “No more.” So something had to be changed, it was very clear, if we wanted the meetings to continue. I was not at that time in any position of power or this sort of thing. And it was Gene Golub. I’m pretty sure—again, I wasn’t privy—who instituted a system of balloting. So what happened was that the meetings, although restricted in size, were where everyone was allowed to apply. Anyone could apply.

The committee then began to expand, so by 1978, I had been placed on it. It just wasn’t the same old people. Then the committee would ballot on them, and the totals would be added up, and then adjustments would be made on the basis of, “You’re overlooking Joe Doe here. He really should come because he’s doing such good work in this.” “Well, maybe. John Black’s work isn’t really quite that lustrous.” “Maybe.” You know. But they would tinker with it around the borderlines, around the edges, not interfering with that. Then people would be written and told that they were invited to attend. It could do no good about the hurt feelings—if people weren’t allowed to attend, they’re going to have hurt feelings. But no one complained very much about the fairness of the process.

HAIGH: Did the change in the process have a significant impact on the character of the meetings, or who was coming?

STEWART: Oh, yes, definitely, because one thing, the student argument was out. A student could apply, although not at first. Rather quickly, they began to ask for people to submit extended abstracts of the talk they would give, if they were invited to give a talk. You might be invited to a meeting and not invited to give a talk. So the committee had an ability to look at a person’s research or their proposed research and what they were interested in, and had something to go on to evaluate them, other than just, “This is a good old boy,” and various word of mouth, good old boy establishment, or “He’s my student.” When that was done, it even increased the fairness, but it made much more probable that a person who was up and coming but relatively unknown could get to the meeting.

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

STEWART: Between mid 1970s and 1980s. It was not just a sudden shift, and details had to be worked out over time. Like the addition of the extended abstract for the talk. Other meetings in numerical and linear algebra adopted the same format, “We may have to limit the number of people of asking for these abstracts.” It’s not quite like conference papers, since these are just abstracts. But it serves a purpose of lending some objectivity to it. Actually, the way it’s done now, or was when I was last on the committee— I resigned at the meeting before last; I had resolved to resign the first meeting after I reached 60, figuring that that was a good time to get off and let other people get on. At that time, we were actually having a ballot and then a discussion, and then a second ballot, so that people would have a chance to say, to make these adjustments, but then we would ballot. Then we would cut off the lines. And they had to be cut
off pretty arbitrarily. When you actually looked at these ballots, what they came out with, it was the top, which was clearly evident, and then the curve of the number of ballots would slide down to be very nearly horizontal. At the size of the meeting, you would be cutting it off, but it would be to some extent arbitrary, which was another reason for re-balloting to make sure. But at least to the extent that the opinions of the executive committee reflected things out toward the 125th person or so, the scoring was rather flat there. But I certainly thought it was a fair procedure.

HAIGH: So after those changes were fully implemented in the early 1980s, did it stay pretty much the same for the next 20 years?

STEWART: Yes, pretty much, and I think it’s pretty much the same now.

HAIGH: Gene Golub had actually mentioned you as an advocate for keeping things small.

STEWART: Yes, very definitely. We disagreed on this. Now when I say “small”, I say relatively small. A meeting of 125 for an intensive research meeting over a period of week, it’s not entirely a small meeting; it’s quite a bit larger than the original Gatlinburg meetings. So I was quite willing to see that size. But at one point, Gene wanted to certainly expand it quite a bit beyond that, and I just didn’t agree. I thought that expanding it would have a detrimental affect on it.

HAIGH: Okay. Has the overall size of the field been increasing in the last 20 years?

STEWART: It’s increased. Well, I mean, if you look at journals like SIMAX you can see that at least by the number of papers that it’s increased considerably. But once again, these are hard judgments to make, but I still don’t think that the Gatlinburg committee would be improved by trying to follow that increase. What we’d end up with is so much noise. As I say, the top people really stand out, and it peters off fairly much before the 125 that has been sort of traditional in the last few meetings there. It’s this flat curve that makes me think that there wouldn’t be a large benefit of increasing the size of the meeting. But happily, I’m completely out of that loop, and I’m very happy to be. So that will be the judgment of the executive committee, which is as it should be.

HAIGH: Aside from Gene Golub, who were the other long-time members of the executive committee?

STEWART: Well, of course, Alston Householder, but he dropped out pretty quickly. He had a heart attack shortly after his retirement, which he recovered from fully. As a matter of fact, he spent a couple of years as Chairman of the Math Department at the University of Tennessee, then he moved out to Malibu with his wife, who died out there. And I think at about that time, he was attending the meetings regularly, but he was not part of the thing. So you had Jim Wilkinson, Fritz Bauer, Gene Golub, myself, after 1978, and things began to blur at that point, over time. I mean you can see that by the Whistler meetings, you look at my web page, that none of those are present, except for me. And that was my next to last meeting, I think.

HAIGH: Let’s see. I think Whistler was 1999.

STEWART: Yes.

HAIGH: After you submitted your resignation, you remained on the committee until 2003.

STEWART: Yes, that’s right. In any case, we kept adding. We tried to keep a balance between European and American things, and also a balance in the meetings. It generally was one meeting in Europe, and then two in North America, which could include Canada and the United States. It
basically meant Canada or the United States. We tried to keep a nice balance between that. We brought on Angelica Bunse-Gerstner. Also, Diana O’Leary was brought on to the committee so that we have a good representation of women on it. Of course, we tried to keep it broad in the sense of people who covered the subject. And it continues to evolve.

HAIGH: So are there any other aspects of those meetings that we should cover?

STEWART: I think we’ve covered most of it. I would just conclude that I think they’ve been very much to the good of the field, that they’ve had a positive influence over the years. It’s, as I say, unfortunate that you have to make decisions that are exclusive, but I don’t see how that can be avoided. And I think that when you balance the gains over the deficits, that the gains win.

HAIGH: Let me ask you about John Rice’s series of meetings in the early to mid 1970s. We know that you attended at least the third one of those, because you had the article in the book. Did you go to the other ones?

STEWART: No, I didn’t. So this was, they had invited me to go there. But other than LINPACK, I was not really a specialist in software development, for that matter. Obviously, I was very much involved in it with LINPACK, and I have published a number of codes and this sort of thing, and had been very interested in seeing the algorithms get out and distributed. But my primary area wasn’t software as such. So I can’t tell you very much about these meetings.

HAIGH: Did you remember anything from the one that you did attend?

STEWART: Not all that much [chuckles], to tell the truth. Some of these meetings tend to blur with the years. I’d rather stick to what I can say definitely.

HAIGH: Okay. I know that some of the people who were more extensively involved with software, see them as being the important developments in the cohesion of that community. As a semi-outsider, would that reflect your own perception?

STEWART: Yes, I think so. I mean the people who did the various BLAS this was not an effort that I was a part of, at all were, I think would have attended these meetings. So not being able to say anything about the meetings themselves, let me just say this. There was a lot of activity, and these meetings were an indication of the fact that this was considered an important topic and that people were working actively in it. Certainly, my own impressions are that most of this activity was to the good.

HAIGH: From 1980 to 1984, you were Chair of the ACM SIGNUM Group.

STEWART: Yes. I’m very embarrassed about that, because I was King Log—I really basically did nothing, and I shouldn’t have— Well, I’m not really executive material and should have realized it.

HAIGH: So was that your only elected office in that association?

STEWART: Yes. And after that, I just never tried again. I’ve never tried to be a Department Chairman for anything else. My real problem is that as I mentioned before, I don’t really have a research program in any sense. I go from problem to problem as I see it, and that takes precedence over almost everything except my teaching. When I see a really good line of research, I follow it very quickly and very heavily.

The other thing, though, that I have done, I’ve done a lot of editorial work, and I take that very seriously, too.
HAIGH: All right. Just to review that in slightly more detail, had you been active within SIGNUM during the 1970s?

STEWART: Yes. Well, I was on the board. That was the thing, and that made me a natural to do it. I had been involved with LINPACK, so that also tended to make me a natural. I guess my desire for the honor overcame my good sense.

HAIGH: Can you remember when you first got involved with SIGNUM?

STEWART: No. I can’t really, which probably shows how much I really did. I would publish the occasional thing in the SIGNUM newsletter. I first published one when hand calculators were out. It was doing numerical analysis on the hand calculator.

HAIGH: I saw that, actually. I found it in the ACM digital library. [“Matrix Computations on a Hand Held Calculator,” SIGNUM Newsletter, 1981].

STEWART: Yes. It was a fun thing. I really was, in those days, doing calculations on airplanes.

HAIGH: So it looked like you’d had a little grant to try out three different calculators.

STEWART: No, no. I just bought one, I think.

HAIGH: Well, that’s one that I can check. In acknowledgements, you say that this work was supported in part by the Office of Naval Research and a contract number. Then there’s a string of meaningless characters. “The author is particularly grateful to the National Bureau of Science in providing him with an HP41C.

STEWART: Oh, okay. The National Bureau of Standards provided me with one. I had started going out there once a week at that time. Yes, now I do remember. But no one supported me by buying me a bunch of calculators. It was just that I got that, and then started playing with it, and then wrote the article.

HAIGH: So the ONR funding you mentioned was just a general…?

STEWART: It was just general. I put that on anything that I did. But at that time, ONR was very flexible—I could write very general proposals and then do pretty much what I wanted, as long as I produced enough output. So it was a very, very convenient relationship. That went on for some years.

Incidentally, I should mention that I also published in SIGNUM News a little article on inverse perturbation for least squares.

HAIGH: Yes. [“An Inverse Perturbation Theorem for Linear Least Squares Problems,” SIGNUM Newsletter, 1975].

STEWART: Yes, yes. That also set off a bunch of research, because I had proposed two theorems for backward perturbation. One is interested in this because one would like to assess how good a solution is, in the sense of: does it nearly solve the exact problem, or the original problem? You know, is it a slight perturbation of the original problem? So one way of testing this is to try to compute some kind of a residual that, if you exactly solve the problem, should be zero. And then you want to throw that residual back on to the original problem and say that a small residual means a small perturbation of the original problem. And this was trivial for linear systems and was non-trivial for least squares problems. I proposed two ways of doing it, but
neither one of them was necessarily optimal. It was only several years later that Sun, my co-worker on the book on Matrix perturbation theory, cracked the problem and gave the definitive solution to it. But this started another line of research, and not necessarily just for least squares problems.

HAIGH: Let me ask you a little bit more about SIGNUM itself. During that period in the 1970s, what would you say it’s main contributions and areas of activity were as an organization?

STEWART: Well, it would organize an occasional meeting, and the SIGNUM Newsletter was actually for me, the most interesting part. I wouldn’t say “gossipy”, but a very informal organ, and people could place things, just like I placed this little least squares thing, or I thought it was little at the time, or the desk calculator thing. They also made an award named after George Forsythe. But I don’t think it was a major influence in numerical analysis. In general, it was sort of one of the smaller SIGs, in the ACM. You know, things like SIGGRAPH really were big powerhouses. So numerical analysis played an increasingly smaller role a time went on in the ACM’s funding. I mean, when ACM first came out, it was loomed very large. Alston was president of the ACM, I believe, and Jim Wilkinson published his fundamental paper on rounding error analysis for linear systems in the Journal of the ACM. So the presence of the numerical analysis was very great at that time. But the people, I think tended to move over toward SIAM in later years, and it just sort of faded.

HAIGH: So do you think that shift in emphasis within the ACM as a whole limited the effectiveness of the SIGNUM group?

STEWART: Yes, I think so. In other words, I think that there were fewer people interested in SIGNUM, and as the special interest groups and SIAM developed people found a more natural home there. Of course, optimization is always the big exception to this, because optimization, by historical accident, is not considered part of numerical analysis, even though it could have been easily have been otherwise.

HAIGH: Is that because of its associations with operations research?

STEWART: Exactly. Certainly, the unconstrained minimization part tended to be more connected with numerical analysis than the operations linear programming and constrained optimization. You’d also find that the latter is being done now, though, very largely in business schools. I mean, Gene’s protégés, Walter Murray and Phil Gill, and Michael Saunders, who was his student, I believe was associated with the business school at Stanford, not with the computer science department, and certainly not with mathematics.

HAIGH: When you were chair of SIGNUM, did you interact with the ACM headquarters?

STEWART: Yes. And especially in one rather frustrating thing. The ACM basically wanted to get a hold of some of the money that SIGS had, so they had this long meeting in which we were to discuss the future of SIGS. It was held in place, I believe out in Virginia, a sort of pyramid like bunker—ugly building. I forget which corporation had it. They leased it out for meetings. This is my first and only experience with a facilitator who they brought in. So we spent two days, basically coming up with nothing. I found it very frustrating. Oh, it was nice meeting people and drinking beer afterward, after the day. But it was basically not a very satisfactory thing, and I kick myself about it. Because of that I had declined a chance to go to Moscow.

HAIGH: So did you feel, in general, that the headquarters and the parent organization were doing a good job in supporting the SIG?
STEWART: Well, I wasn’t a member of a very big SIG. And as I say, I think that there was some conflict in there, that the central part of the ACM felt that they had too much autonomy. But this was many years ago, and I’m trying to recall impressions from that. So I would suggest that if someone really wanted to know what was going on, you’d quiz some people who were chairmen of the larger SIGs at that time.

HAIGH: You implied that as chairman, you didn't really do very much.

STEWART: No, I really didn't. And as I say, I’m very embarrassed by it. Well, that’s all that needs to be said.

HAIGH: Oh, okay. From some point in the 1970s onward, the group seemed to be in their decline, which culminates with it being wound up in the 1990s.

STEWART: Yes. Well, I didn't follow it afterwards.

HAIGH: So you don’t have a sense of what happened to it, after 1984?

STEWART: No. But as I say, I do think the driving force behind was basically the exit of numerical analysis from ACM. Perhaps the real, the group that was at present, the people who were the real successor of SIGNUM and this sort of thing was really TOMS (Transactions on Mathematical Software). That’s the part of my area, and this area that ended up in prominence in the ACM. And of course, TOMS is just really just an illustrious journal.

HAIGH: Did you have any personal involvement with it?

STEWART: With TOMS? No, I submitted articles to them and Ron Boisvert, who was the editor-in-chief for many, many years, is a good friend. He’s actually my boss out at NIST. So I know this journal was edited with considerable integrity and care.

HAIGH: I think this would be a natural transition to SIAM then. So do you remember when you first went to a SIAM meeting?

STEWART: Let’s see. You know, I can’t. The first one that I really remember is the Denver meeting, which I believe was in… [looking through paperwork] Was it in 1974? I must have gone to others. At that time I was also going to American Mathematical Society Meetings. So they sort of merged. For example, I went to an AMS meeting in San Antonio, fairly early on, when I was in Texas.

HAIGH: Well, it says that you were co-chairman at the SIAM meeting in Austin, Texas, in 1972. You must have been a reason to be involved, by then.

STEWART: Yes. Yes, I was. Okay. But this was a very special case, because this was…

HAIGH: In your hometown.

STEWART: In my hometown, yes. I was more of a local organizer, as much as anything. And actually because that was the year I left, I started the organization but I was not the one who actually carried through with it.

HAIGH: So over time, did your interest shift more towards SIAM?

STEWART: Oh, yes, very definitely. The AMS
meetings, they were very big. And I think maybe the last one that I went to was the one in Las Vegas, and who could resist Las Vegas, or watching the mathematicians with methods go to their deaths at the gaining tables? [Chuckles]

HAIGH: Well, actually we’re having a Society of The History of Technology meeting there in the fall, so I’ll also be able to see the historians of technology at their wildest.

STEWART: But SIAM basically became my home, as far as a professional society.

HAIGH: Did you have a sense of how large the meetings were in the early 1970s?

STEWART: Oh, they were larger than a typical SIG, or the LA group or say the Optimization Group, but not all that much larger. It was much cozier.

HAIGH: So would that be several hundred people?

STEWART: Probably, yes. But I would be hard-pressed to say. I never looked at any statistics and this sort of thing.

HAIGH: But it was small enough that you could talk to all the people that you wanted to talk to and go see all the papers you wanted to see.

STEWART: Yes, yes. Go to all the papers I wanted to see, and I guess just important, socialize with people who weren’t dead in my area. That was very nice. And I missed that in our meetings. It’s not that the meetings are not good. But for example, John Dennis, who was at Rice—he’s retired now, and a bright and very important in the optimization area—was also a close friend, and we later saw each other at the SIAM meetings, but as they split off, I got to see less and less of him.

HAIGH: Now other than, of course, going to the meetings and presenting papers, would you say that your main contribution to SIAM has been through editorial boards?

STEWART: Yes. I certainly have.

HAIGH: So there’s the list of editorial boards. You may as well talk about any of those that you think are significant, not just the SIAM ones.

STEWART: Okay. Well, very early, I was given a SIAM Journal on Numerical Analysis. At that time Werner Rheinboldt was the editor. I’m sure you know how SIAM generally handles things, basically. The associate editors, after a brief break-in period, are given pretty much free reign. Some of it depends on what the editor-in-chief is doing, but basically the associate editors get the papers, send them out for review. There must have been some central accounting system, but no central editorial control. And then they would get two referee reports usually, on the same paper. Or they might decide that this is a circle-square and reject it out-of-hand. And then they would get their recommendation to the editor-in-chief, who generally would, unless they saw something very wrong, would just rubber stamp it. Oh, as a matter of fact, in SIGNUM we communicated directly with the author, so I would write a copy to the editor-in-chief, and Werner, with my decisions. For a year he asked me to send them to him and he communicated. And then after that, I did it, so it’s a fairly loose editorial system.

HAIGH: Would you receive papers in a particular specialized area?

STEWART: Yes. I mean they would not be general papers. SINUM was much broader. SIAM Journal on Scientific Computing, or Scientific and Statistical Computing, as it was started off,
hadn’t come into existence at the time, nor had the *SIAM Journal on Matrix Computation*. So virtually all kinds of numerical analysis were included in there. So for example, I certainly would not have handled a paper on Finite Element Techniques, or this sort of thing. So yes, the editor-in-chief was responsible for finding an editor that could handle this sort of thing. As a matter of fact, this was another disadvantage of my early involvement with polynomial routines, is that I got all the bad papers on polynomials, and there were a lot of them. But in any case, so at that time, now the *SINUM* is much more specialized and very seldom publishes anything in numerical linear algebra, for example. That’s all gravitated toward *SIMAX* in some sense, to *SISC*.

HAIGH: So that’s the parallel process to what’s happened with meetings and the interest groups.

STEWART: Yes.

HAIGH: But do you feel that with the journals, it’s a happier development?

STEWART: Yes. I mean it had to come, and I mean, you know, nothing is keeping you from looking at the other journals, and I do. Whereas, when you’re at a separate meeting, you can’t go to all of the meetings, and that. So this was a necessary change. You had to have venues for these. The only reservations I have are that by doing this, you have a chance of watering down the field by making it too easy to publish, in the sense that the quality of papers can go down just because you have more venues to spread them out. And I think that to some extent, this has happened. I don’t call it a crisis or anything, but I do think that if you look at the earlier SIAM journals or *Numerische Mathematik* and others, you can barely open one of those early ones without finding in it some sort of a classic paper, or a very good paper in it. I don’t think that’s true today. There are certain very good classic papers being published, but the proportion is much lower than it was in the earlier editions of the journals. So that’s a negative side, but I don’t see that it could be helped. But on the whole, the gain is there of being able to represent these various fields.

HAIGH: So back to your personal involvement. In terms of your style, do you think that you would put more effort into it, write a more careful response, compared to a typical editorial board member?

STEWART: Well, I often would get hands-on with the thing and grab it, and actually try to help a paper or this sort of thing. Also, if there was a very bad paper and a lot of editors do this, I’d simply write a referee’s report myself, and then see the saddest faces for rejection when it was very obviously a bad paper, and this is not an uncommon practice. But it was only when these were obviously flawed and just pointing out the flaws was sufficient. There’s no point in going out and spending six months waiting for a report from someone else that would say essentially the same thing. But also, especially in my work with *SIAM Review*, I tried to take papers that were flawed in some way and actually help them along. This is in the education section, I’d try to help people get them published. More like editing a book, even a book of fiction, although I don’t want to imply that these papers were fiction. So I’d generally taken my responsibilities as editor, on editorial boards. I’ve never been tempted to be an editor-in-chief again. I’m just not cut out for that kind of thing.

HAIGH: Do you think you would be trying to do it too hands-on?
STEWART: Well, there was probably some of that. But just as I say, above a certain level of administration, I’m just not good at it. It’s the Peter Principle. You’re familiar with the Peter Principle?

HAIGH: Well, if it’s the Peter Principle, you’d already be doing that, and you’d be doing it badly, and you’d keep doing it.

STEWART: No, no, I’m trying to avoid the Peter Principle. If I would to take over one of these jobs, I would be at my level of incompetence. So I’ve not, but I’ve certainly enjoyed working as an editor of things. I’m winding down now.

HAIGH: And so with those specific journals, is there anything that stands out in terms of places where you’ve made a particular contribution or where you’ve had some kind of influence in the development of the editorial policy of the journal?

STEWART: I don’t think, really. Basically, as just as an ordinary editor. With the possible exception of Electronic Transactions on Numerical Analysis, which I was in on from the birth. But all of the other journals were generally fairly well established. I mean Mathematics and Computation goes way, way back to the days when there were mathematical tables and other aides to computation. So it’s got a long history. Numerische Mathematik was very much a going concern when I became a member of its editorial board. Again, Linear Algebra and its Applications was a long-going concern before I arrived there. And these journals get their momentum, and you can really do very little. Perhaps as editor-in-chief you can do something, but as an ordinary editor, there is very little you can do to change the course of the journal.

HAIGH: You were also involved with the Institute for Mathematics and its occupations.

STEWART: Yes. This is basically a pro-forma thing. I really have not done much with that. So I’ve not been active in.

HAIGH: So what are your general impressions of that documentation?

STEWART: Well, very little, because I’ve not been very active in it.

HAIGH: But are you not active in it because of the impression part of it?

STEWART: No. It has nothing to do with that. It’s just that somehow, I’ve never really connected with it and been active in it. And they’re happy to keep me on it.

HAIGH: There are also a number of meetings that you were involved in organizing. There may be some that you don’t have anything in particular to say about, but if you’ll just review the list. And if there is anything that you do want to talk about, at that specific time.

STEWART: Some of these meetings refer to part of my career that we really haven’t spoken of. It’s moderately important, but probably we don’t really have time. But it’s when I did interact with statisticians for a good time, in statistical computing. I actually was going to try to write a book on linear algebra and statistical computing, and this type of thing. So I interacted with them. During part of that time, I was on the technical advisory board for the biomedical data package, which at the time was the package used by the medical profession and the biological scientists for doing statistical analysis.

HAIGH: What time period was that?
STEWART: That would be in the late 1970s and early ‘80s, or early to mid-‘80s. Also, some of these others reflect an interest that I had in queueing theory and computer systems, that I did some work in that. It never really was very significant, but it did clean up some areas that people had been working on. So I was on a number of program committees for those.

HAIGH: Why was it that you didn't follow through with the plan to write a book on…?

STEWART: Because it just turned into a numerical linear algebra book. You know, the statistics became minimal; it would have sort of been like rewriting my textbook with examples from statistics. With that, I made a couple of starts on it, and it just never panned out to my satisfaction.

HAIGH: So the more that you worked on it, the more that you realized that the numerical linear algebra in statistics was the same as the numerical linear algebra in other areas?

STEWART: Well, one thing was, I mentioned earlier that mathematics can be a different language or can be a common format for people speaking a different language. In fact, what I had wanted to communicate to the statisticians in the book was the use of matrix decompositions. If you look at statistics in books, certainly at the time and to some extent now, they’re full of involved formulas involving matrix inverses and this sort of thing. A lot of this stuff can be rephrased in the language of projections and the QR decomposition and this kind of thing. So I wanted to communicate this to them. But again, when you start writing down and everything, it comes out just linear algebra, and “Hey, it would be a good idea to consider doing it this way.” That’s not really what I wanted to spend my time doing. The involvement was real. I knew these people. As a matter of fact, I spent two quarters at the University of Minnesota in the Statistics Department. This was partly because Astrid [Schmidt-Nielsen] had a post-doc there. We weren’t married yet, and were planning to get married, and so I walked into Steve Fienberg’s office—he was at the applied math group at the St. Paul campus—and asked if I could come visit for a couple of quarters. I also did the computer science department, so I had a joint appointment there. So I’ve actually been a professor of statistics, in some sense, at least a visiting professor of statistics. It was a wonderful time, but it just didn't have the research payoff, and I gradually dropped out of it.

HAIGH: In terms of edited volumes and meetings, I also noticed some references to sparse matrices. I hadn’t seen this as one of your personal research areas in…

STEWART: Well, it isn’t really, but I did edit one with Iain Duff on the Sparse Matrix conference. [Sparse Matrices and their Uses. Academic Press. I. S. Duff and G. W. Stewart (Editors) (1979)]. I followed the area with considerable interest, and sometimes wish that I had gotten involved myself. It’s a very exciting area, and some very good people have just done wonderful work. Only recently have I gotten down into the technical details of it with these people. But it’s a very exciting area. On the other hand, I’m not at all sure that I would have done very well if I had tried to do this because this isn’t quite my style of work. The kind of mathematics that it’s based on are not the kind that I do all that well. So for my own work, it was probably a good thing. But I have nothing but admiration, and certainly nothing to keep me from going to the meetings and serving on a program committee or editing a volume of it. I certainly do enough of an area to do that.

HAIGH: So is there anything specific that you want to say about the meetings?
STEWART: Not really. On some of them I served, I was actually active; then in others I came to meetings to review the papers that were coming and this sort of thing. I think this is a very typical sort of thing here, but there’s nothing particularly outstanding in it. It’s what you would expect of anyone who’s active in the field and not especially into doing a lot of administration of meetings and this sort of thing.

HAIGH: So returning now to your own personal research after the LINPACK project, I understand that you have a general statement that you would like to make before we talk about the paper?

STEWART: Well, it’s just that I think that in some sense, I’ve done a lot of work since that period. I mean we’re talking in decades. So I’ve done a lot of work. But I think the pattern of my work by then was pretty well settled. It was essentially problem-driven in the sense that I would see an interesting problem, usually a need for an algorithm to do this or to do that. I would pursue it usually with some energy, until I had solved it, which usually meant either getting an algorithm for it and often doing a good deal of mathematics, including perturbation theory or rounding-error analysis to understand it. I think that if you look at my vitae, you will see that this is very typical of the way that I’ve approached this.

HAIGH: In terms of seeing the need for an algorithm, is that connected with what you’d mentioned earlier about liking to hear papers in different areas and being in touch with what people are doing, more generally?

STEWART: Yes. It’s very easy to sort of become inbred in the sense of wanting to generalize a particular matrix decomposition or this sort of thing without having a particular application reason for doing it, and then this is, on occasion, done. I’m not going to point any fingers or anything, but for me, I only really like to think about an algorithm if I think there’s some real chance that it will be used by someone in a real application. Now this hasn’t always turned out to be the case, but at least that’s where I like to start from.

HAIGH: So you have identified these particular papers. Following up on what you said, if you have about those or any of the other papers, any particularly illustrative stories about where your perception of the need for the algorithm came from. That would also be interesting to hear.

STEWART: Okay. The paper on Simultaneous Iteration was basically a generalization of my thesis to non-hermitian matrixes, or non-symmetric matrixes. [“Simultaneous Iteration for Computing Invariant Subspaces of a Non-Hermitian Matrix,” Numerische Mathematik 25 (1976) 123-136].

This was a considerable difficulty. We’ve really talked about it before. For example, it was this paper that I spun off the QR perturbation theory from. We’ve mentioned it before, but again, the necessity of solving large eigenvalue problems that were non-Hermitian, is a no-brainer, as far as its need in various areas. I must say that this whole approach has largely been succeeded by the Arnoldi method, or variances of the Arnoldi method, which is an even better way of doing it, although this approach does have some very strong virtues over the Arnoldi in some particular cases. As a general-purpose tool, the Arnoldi method works better, especially in the form that Danny Sorenson developed, an ARPACK, which may have been released in their documentation as a SIAM publication.

So in any case, that, as I say, was a generalization of my thesis. As in my thesis, it involved new mathematics and new algorithmics in this. At the time I had written a good quality code but I had
not done the work to publish it. So ultimately, there was a real algorithm and code that existed, even at the time I had published this. All of the paper was basically the mathematical analysis of the algorithm.

Later I took it and cleaned it up and published in Transactions on Mathematical Software. [(with Z. Bai) “Algorithm 776: SRRIT: A Fortran Subroutine to Calculate the Dominant Invariant Subspace of a Nonsymmetric Matrix,” ACM Transactions on Mathematical Software 23 (1997) 494-513]. So this was quite a bit later. In fact, we had considerable difficulty translating my earlier code from the card text that it was originally written in. So I guess that’s all that we need to say about that.

Actually, in making this list of papers, I just chose some that appealed to me. There are many algorithms in my papers. So “Computing the CS Decomposition of Partitioned Orthogonal Matrix,” [Computing the CS Decomposition of a Partitioned Orthogonal Matrix,” Numerische Mathematik 40 (1982) 297-306].

I had introduced a thing called, “Cosine/Sine Decomposition,” or “The CS Decomposition,” and this decomposition goes all the way back to Jordan of the Jordan Canonical Form thing, of the last half of the 19th century, but not expressed in matrix form. It actually appeared in Davis and Kahan [C. Davis and W.M. Kahan Some new bounds on perturbations of subspaces. Bull. Amer. Math. Soc.,75: 863-868(1969)] in a somewhat disguised form. And the SIAM paper I made it explicit as a matrix form, as a decomposition of an orthogonal matrix, and the name stuck. [On the Perturbation of Pseudo-inverses, Projections, and Linear Least Squares Problems," SIAM Review 19 (1977) 634-662]. I named it the CS Decomposition and that name stuck and it’s known by that. And the difficulty is that it turns out to be a damnedly difficult thing to compute. The proof is constructive, but owing to rounding-error can fail completely to actually compute the decomposition you want. This was sort of an open problem, so on this particular paper I made a stab at it and actually got an algorithm based on what would be called a Jacobi-like iteration for the thing, and was successful. It turns out later that Charlie Van Loan and others got better algorithms. One of the things that I often find is that when I tackle a problem, if I’m first in it, I get a good algorithm, but there are better ones waiting in the wings. A person ended up talking about patent laws said, “One thing you have to realize is that patents are so generous, is because no one can get everything right the first time.” So you have to have a lot of leeway on that. That’s true also of algorithms. You may get the basic algorithm and the ideas there, but you’re bound to leave things out that can be improvements.

[Tape 5, side A].

HAIGH: So you had just finished talking about that one paper, and you were about to start on another.

STEWART: Yes. Well, this is an updating algorithm for sub-space tracking. [An Updating Algorithm for Subspace Tracking," IEEE Transactions on Signal Processing 40 (1992) 1535-1541]. I’m very pleased with this one. It’s very nice. The problem that it addressed was array signal processing, in which people have arrays of sensors that are receiving signals, and you try to use the differences between how the signals are received on different parts of the array to find information about the body that’s making the signal: it’s velocity, whether it’s approaching you. Obviously, it’s sort of a radar-like thing, except that you have more than one sensor receiving the signals.
HAIGH: Would that be useful in radio astronomy?

STEWART: No. It would be more useful in protecting a flight or a plane or tracking other airplanes and things like that.

So in any case, the problem was that this leads to certain matrix problems that have to be solved in real time, and they have to be updated each time you get a new signal. You receive signals from period to period, and these matrix decompositions have to be updated. Largely the algorithms they use to analyze it were based on what’s called the singular value decomposition. So I was at a meeting on the use of singular value decompositions, and I’ve always felt that as excellent as singular value decomposition is, very often it can be replaced by the much cheaper QR decomposition with pivoting to reveal rank deficiencies. So I was trying to work that into this context, and not with any success. It was not very successful, but I was very focused on trying to get these simpler algorithms. Then at the meeting, I was talking about my frustrations to Frank Luck, who was at that time at Cornell and later went to RPI. He made some offhand comment to the effect of, “Well, what about the complete orthogonal decomposition?” You don’t need to worry about what that is exactly, except that it was like a light coming on. I went back to my room—this was that evening—and I sat down and worked out the basic ideas of this whole algorithm in a couple of hours’ time. So I published it there in the IEEE Journal. So you can tell that it was definitely devoted to them. There were some generalizations of it that I made, but it was quite a hot topic for some time, and is still used in some applications. I was thinking that this was a real-life problem, but in point of fact the whole idea of array signal processing was itself sort of researchy and experimental and there weren’t any real arrays out there. But my intentions were good.

HAIGH: Did they ever get around to building any?

STEWART: I haven’t thought of it since then, or for some time. But by and large, they were using old, non-array techniques that they’d used. So this was fun. So this was, incidentally, an algorithm that required no mathematical analysis whatsoever. It was purely algorithmic.

HAIGH: So that didn't feed back into any more general mathematical insights?

STEWART: No, it did in some sort of interesting way, because it actually suggested some new ways of looking at the perturbation and of invariant sub-spaces, but not because of its applications to signal sub-space tracking or the algorithm itself. It’s just that the basic structure in the decompositions that you used allowed you to look at things in a somewhat new way. And yes, there were new theorems to get out of it, but they didn’t reflect back on the algorithm. They weren’t necessary to prove, except to say how good your results were. Those came somewhat later. But again, yes, there was stuff there.

Then another one that I very much liked (we’re skipping some years now here, because I don’t want to go over everything or take up much time). Four Algorithms for the Efficient Computation of Truncated Pivoted QR Approximations to a Sparse Matrix,” Numerische Mathematik, 83 (1999) 313-323]. An important problem in data mining is latent semantic indexing. You’ve got basically a matrix that stores the frequency of words occurring in documents. So you’d have to have words along the columns and documents along the rows (I may be getting it backwards as far as the standard presentation). The idea originally was to look at the singular value decomposition of this matrix. This is, again, a singular value, versus QR. From this, you can extract information that allows you to predict which references certain key
words will occur in. Obviously that’s a nice, important thing to be able to do. So I may have been refereeing a paper for it. But in any case, I was reading a paper on it, and the author said that the QR decomposition is not really suitable for this. My immediate reaction was, “BS” you know, “it’s got to be.” So I started examining it, and basically I got this tier of four algorithms of increasing sort of complexity, that you may say that provided an alternate solution to the problem, and I published this. Again, it was written rather quickly. The only thing that kept me from writing it even faster was that I discovered Algorithm One, and then I discovered Algorithm Two, then Algorithm Three, and Algorithm Four, and then I’d have to rewrite the paper.

HAIGH: If you didn't have tenure yet, you could just write four papers.

STEWART: I never played that game [chuckles]. I just don’t believe in it. You write the paper that has its own integrity and publish it as that. I mean, what you could say I was playing the game, except I had tenure when I spun off the QR perturbation theory from the other paper, but I just thought it was just such a neat little topic that it deserved its own place. But maximizing publications has never really been a desire. Besides, they all fit together so nicely, it was pretty. And so I published this in Numerische Mathematik. It went through in record time, and then I did nothing with it for a time.

So then I did nothing. I really hadn’t done any implementations, but then later on I decided it would be really nice to see how well it really worked, so I got in touch with Michael Berry, who has been doing applications in data mining and related things. He’s at the University of Tennessee at Knoxville. He and his graduate student, whom I’d never met, and who was working on a master’s degree, we three got together and did a MATLAB reference implementation of this code, and actually applied it to a problem where it worked with the massive data set, and it worked very well, as it was supposed. [(with M. W. Berry and S. A. Pulatova) Computing Sparse Reduced-Rank Approximations to Sparse Matrices," ACM Transactions on Mathematical Software (TOMS) 31 (2005) 252-269].

HAIGH: You had mentioned when the tape wasn’t running, this issue of using MATLAB implement things and some of the differences you’d find when you’re using that method, versus using the old days when you’d be doing everything in FORTRAN and compile some library code in with it.

STEWART: Well, MATLAB has been revolutionary. That’s the only way to describe it, and Cleve was very courageous in making it so. I had thought that some sort of package like that might be good, but I worried so much about when I was thinking it about optimizing memory usage and all kinds of things. I didn't think for long about it And Cleve just went ahead and did it without thinking things through completely. Of course that was its success. Probably he did it just simply because it was originally intended to be an educational tool. The lab means, the matrix laboratory where students are working. That’s where the name MATLAB came from.

HAIGH: Yes, and the original late-1970s version, I think, it was just for educational purposes.

STEWART: Yes. Yes, exactly.

HAIGH: And then they added the programming language.

STEWART: No. It had the programming before.
STEWART: Well, it still is interactive. It’s interactive and interpretive.

HAIGH: Okay, but as I recall, it was only interactive. You wouldn’t write functions, but you would write the series of instructions and...

STEWART: Oh, yes, at the very first. But by the time I saw it, they were in files in various ways. But yes, initially, it was even more primitive. But in any case, Cleve just charged forth and did it, and there we were. So what you had in that lab, the virtues are that it’s very easy to work with. It has almost all the various matrix functions that you could want, as far as the primitive ones, multiplication and this sort of thing. At the time, basically Cleve grafted LINPACK onto it, and EISPACK, so you had good computational algorithms underlying it. Later on, they went to LAPACK for the algorithms. This was done at MathWorks. So you had the idea of working in a workspace, which is really wonderful because you can just see what your variables are. You can run it. If it bombs out, all the variables are there. One of the most important operators in the MATLAB is the semicolon, which suppresses the printing of the statement, so that if you’re debugging something, getting a debug out is just a matter of removing a semicolon off it. Wonderful. I mean this is just great. It makes experimentation easy. I’ve saved myself no end of trouble in mathematics now, eliminating false lines of inquiry by writing up a little MATLAB code to see if I can find a counter-example or something. So it’s just wonderful.

But it also has some of the defects of its origins. It originated as an interpretive language, and interpretive languages are necessarily very slow, even though MATLAB does some pre-processing. Basically if you write things at the level of inner loops that you would in ordinary matrix computations, it’s slow; there’s no question about it. What that means for the user of MATLAB is that you can’t expand the package yourself efficiently. If you have a brand-new decomposition that you want, that requires use of things like plane rotations or even Householder transformations, well there are no Householder transformations or plane rotations in MATLAB as an explicit construct. So consequently if you write it out you would in a higher-level language where it could be optimized then it’s extremely slow. The only way you can do it is to actually write it in C or FORTRAN, and then connect it through MEX-files which is sort of an unsatisfactory thing. You’d really like to expand the language with it, with a new function that didn’t have to worry about the mechanics of interfacing between MATLAB and some other language that would differ from computer to computer. One of the nice things about MATLAB is that even though you’re working on different computers, it looks the same everywhere.

So this is a real disadvantage of MATLAB, and you can see it in this paper. Although actually MATLAB does a pretty good job on this—we’re using a lot of the MATLAB primitives and very little looping in this. We do devote a good bit of time as to how you would structure it and see if you wanted to convert it there. Even on a paper that I’m working on now, as a matter of fact, that I was working on when you walked in the room, is on a block version of the Graham-Schmidt algorithm. I’m implementing it in MATLAB because I need to see its numerical properties. But this wouldn’t be at all suitable for timing it or to see how it works. So MATLAB just has its very great virtues. To some extent, it revolutionized the way we approach matrix computations. But it also has its deficiencies and these deficiencies can be traced back to its origin. There’s not much you can do about it. That’s my opinion of MATLAB. I’m a great user, and a great fan of it, but you are up against this fact that it originated to do interpretive language, and if MATLAB primitives don’t fit what you’re doing, you’re bound to be slow.
HAIGH: In terms of the use that you’ve made of this kind of technology in your own research, would you say that the increased availability of powerful interactive computing has let you research in a more experimental way than you would have been able to do previously?

STEWART: Oh, certainly. As I say, very often you’ll come to a fork in the road. You can’t be like Yogi Berra and take it, you’ll have to go one way or another. The old fashioned way, you would just have to sit down there and do the mathematics and say, “Is this going to work, or is this going to work? If I go this way, am I going to end up further down the road at the dead end?” Usually, you would have to do a mathematical investigation because it would take a long time to write up a program to try to ferret out and counter examples or this sort thing. MATLAB makes it so easy that you can very often save yourself a lot of time when you come to one of these forks in the road by just simply writing a little program, running a thousand randomly generated cases (well-chosen; I don’t mean “random” meaning all over the map, but within the constraints of your problem) and see if something goes wrong. Very often it does, and usually the way it goes wrong often gives you an idea of why it was going wrong. So it’s a marvelous tool for research, especially algorithmic research. But useful for others too. In any area, this would apply. The ability to write things quickly and easily is an important aspect of MATLAB.

HAIGH: There are a couple of papers in your work on perturbation theory I’d like to discuss.

STEWART: Well, this is an important area for me, but most of my perturbation theory has come from either my algorithmic work, or consideration of actual practical problems. It’s not been a theoretical mathematical structure. One of the papers here, that was really sort of my second paper substantial paper. It was on the continuity of the generalized inverse. [“On the Continuity of the Generalized Inverse,” SIAM Journal on Applied Mathematics 17 (1969) 33-45].

Actually, it’s an interesting story. I wrote it because basically I wanted to analyze least squares problems and the perturbation theory for it. It turns out that this was known in some sense. Gene Golub and Jim Wilkinson had analyzed it, but they had only done it in terms of order terms. They had basically done what’s called a first-order perturbation theory, and it gives you expressions for the perturbations, but it doesn’t tell you rigorously when those expressions actually apply. So I decided I wanted to cross the t’s and dot the i’s and do it. Actually it turned out at the time that there were a couple of other people who were doing the same thing: Hanson and Lawson and Victor Pereyra who was at Stanford, working in the Gene’s group, they were all doing it. So I did this thing, and I was really pleased because it was a very elegant solution. I sent it in through Alston Householder to SIAM Journal on Applied Mathematics. I got back a review of it that’s by a person who is a generalized inverse person, to which this problem is related. He said he had done it all, and it turned out that all he had done is a special case. So I basically decided to write up the general case for generalized inverses, and then point out the application to least squares. I did it, and then it was published. You know, when you point out exactly what the referee had said he had done, or claimed he had done, and what was actually needed to be done, it was very clear that the referee had treated a rather special and trivial case.

HAIGH: I think you have talked about that one before, actually.

STEWART: Yes, I think we did, too.

HAIGH: Does that lead into any other ones?

STEWART: Well, yes. This whole work ended in this third paper that I’ve listed, the 1977 paper on “The Perturbation of Pseudo Inverses, Projections, and Linear least squares Problems.” [On

That was just, a lot of other people had worked in this area, and notably, Per-Ake Wedin, who’s sort of a poet of the mathematicians of this area. He doesn’t publish a lot, but what he publishes is beautiful. There was a lot, and so I just decided to collect it all together and write a nice, big survey paper of it. It was that paper in which I also introduced the CS Decomposition that we talked about earlier. As I say, I don’t want to claim credit for the decomposition itself, but I gave it what today is the standard package, you might say. It was a very much fun paper to write, and you could just see it flowing naturally. I’ve done beyond this, a lot of work in perturbation theory, a lot of it inspired by the algorithms that I’ve had to work with. This should be clear from some of my other comments earlier.

Finally, basically the summation, not that I’ve quit doing it, but in 1990 with the book that I wrote, with Sung. [(With J.-G. Sun), Matrix Perturbation Theory, Academic Press, 1990]. He was visiting at Maryland for a while, and we fell to talking. He was starting a book on perturbation theory, and he showed me the thing. Originally, I was going to help him with his English, but it became clear that really he needed some organizational work. So worked together on it. We were equal co-authors in this, I want to state, but the writing’s basically mine. The book is equally inspired by both of us, and it’s been rather well received, certainly by the more applied people. Margaret Wright has told me that it was influential in the optimization community. I had heard that maybe the theoreticians didn't like it so much, because I always chose the most useful result rather than the most elegant or the hardest result to prove. If I can easily prove something with a constant of 2 in it, I wouldn’t worry much about reducing it to the square root of 2.

HAIGH: What would the audience be for that book?

STEWART: Anyone who wanted to know about matrix perturbation theory. By then, of course, I’d been doing it a lot, and since a lot of it came from algorithms or practical problems and this sort of thing, I felt there was a need to summarize this stuff. And also because it was there.

HAIGH: Could it be, for example, for a graduate student who might want to write a thesis that was related? Or would it be suitable for a textbook for a specialized course?

STEWART: Well, certainly it could be used. It has exercises in it and everything, and it can be used for that. I would just say that it’s a place where a graduate student who needed results of this kind could come and find them collected, without having to pile through a lot of research appears, and maybe this was what was there. So as I say, perturbation theory has always formed a large part of my research. It’s a natural, in some sense, with rounding-error analysis. Rounding-error analysis tells you how your problem is perturbed, and then the perturbation theory says what the perturbation does to you. As a matter of fact, that marriage of the two is basically due to Jim Wilkinson, and revolutionized matrix computations.

HAIGH: Except for the LINPACK User’s Guide, that’s your only co-authored book. Was that a happy collaborative experience?

STEWART: Yes. Yes, it worked out well. He corrected a lot. I would write up the stuff, and he would agree on what was going on, but yes, it was.
HAIGH: I think you had mentioned yourself that you have an unusually high proportion of single authored papers. Is that for the same kind of reasons you discussed when explaining why you might not make a particularly good advisor?

STEWART: Yes, it’s exactly the same. I have a very hard time coming up with focused ideas until they’re almost at completion. When they’re there, they’re there. I can tell somehow that I’m making progress, but it’s like walking through a fog or something—I know there’s something up ahead, but I can’t express it very well. Then suddenly it breaks, and things often come just very fast to me.

HAIGH: Does that conclude with this group, then?

STEWART: Yes. I think so.

HAIGH: All right. So we do have two more recent books there. In fact, I think the other books there are the main thing to discuss. Let me cross-reference that with what you have prepared. Oh yes, the MATRAN Software Project. [Matran: A Fortran 95 Wrapper for Matrix Computations," Research Triangle Distinguished Lecture Series, Raleigh, Durham, Chapel Hill, 2005].

STEWART: I think it can be brief because we don’t really need to talk about it too long; it’s still a work in progress. It originated actually at NIST where it was proposed to try to do a package for matrix computations in Java. Cleve was involved, as was Ron Boisvert and some people from MathWorks and this sort of thing. We had a polite disagreement on how it should go. They wanted to do it in a classical, object-oriented approach with a matrix class. I felt it should have more of a feel of a package, sort of MATLAB like, in fact. So we went our separate ways. They produced one and I produced the other.

HAIGH: So by having more feel of a package, do you mean a high level of abstraction?

STEWART: Not so much that. One of the troubles with object-oriented programming as it’s traditionally conceived, is that it really wasn’t designed with binary operations in mind. If you have an object like a matrix, a class like for a matrix, the functions associated have or are associated with the matrix itself. So when you want to multiply two matrixes, you’ve got two instances of its class and you have to decide which is actually the responsible one to multiply. And things like this just make these binary operations not as easy to conceive of. So I conceived of it as there being a class of matrixes with the operations sitting outside of the classes—over them rather than built in.

HAIGH: So that would be more like the traditional approach where you have a function and feed it two matrices.

STEWART: Yes, exactly. I felt this was the right way. Now the difficulty is that Java doesn’t have any overloading, so these functions had to be written out explicitly as C=+AB. This leads almost to an assembly-like language coding of matrix operations; it’s not MATLAB-like at all. So my approach didn't really succeed, in that sense.

But then I happened to look at FORTRAN 95. Actually I was teaching a course on computer science for scientific computing in our class, and I looked at this and realized that FORTRAN 95, which I hadn’t looked at carefully before, could define new binary operators. The syntax is two dots, with the name of the thing. So you could say, “A dot dot times,” and then “B” and you could build up expressions like this. I suddenly realized that what looked rather ugly in Java would look rather pretty in FORTRAN 95. It wasn’t a matter of translation, because there were
many other issues to decide. Java is not FORTRAN 95, and I was also working to understand it. I got a lot of help from John Reid, who with Metcalf has written a lovely book on FORTRAN 95 and now FORTRAN 2003. He actually personally helped me over some hurdles. In the end, I had coded this package MATRAN but only for real arithmetic. I’ve released it and some people have actually used it some, but basically I’ve got a student now who is working on his degree and is also helping me translate this so that we have the full complex arithmetic package with all the complex decompositions and this sort of thing. Then we’ll see how it works. My only guess is that it will thrive if FORTRAN 95 ever catches on in the U.S. If it doesn’t, it probably won’t.

HAIGH: You mentioned this was related to the JAMPACK system.

STEWART: Yes.

HAIGH: So was JAMPAK what the other people wrote?

STEWART: No, they developed JAMA. They wrote one called JAMA, and I wrote one called JAMPAK. And then they wrote one called JAMPACK.

HAIGH: So who wrote JAMPACK?

STEWART: I did.

HAIGH: Oh, so at first you wrote JAMPAK.

STEWART: And then I wrote MATRAN.

HAIGH: All right. So JAMPACK is what you did produce for Java, but you weren’t happy with the aesthetics of it; it would have been very clunky to use.

STEWART: Exactly, so it never really caught on. JAMA has caught on to a certain extent, but owing to some of the limitations I mentioned before, it doesn’t have potentially as wide a range as the approach that I took. The nice thing about MATRAN, and let’s stick with that now, is that it is very open. People can add to it very easily because it’s all built up with FORTRAN 95 modules, and you can just add modules to it.

HAIGH: What are your general feelings on these Java, for numerical work?

STEWART: It’s not very good. I mean you have to do the numerical work in it so people do the numerical work in it, but the people at Sun made decisions that made it very difficult to do any real numerical work, and they show no interest, as far as I can tell, in rectifying those. One of them was no complex type. You basically can’t call a language serious programming unless it has a primitive complex type. The large problem is to some extent, theoretical, but JAMPAK, for example, will not guarantee the storage arrangement of two-dimensional arrays, or for that matter, I imagine, even one-dimensional arrays. Java memory is just sort of an addressless space that is handled by the Java virtual machine and they will not relate it to physical memory. In point of fact, they do the right things in most implementations, or in all implementations I’ve seen. But it is a problem that you’ll be coding an algorithm in a language that just can fail completely, or at least get very poor performance, if they decided not to honor….

HAIGH: So when you say “wrapper” in the title of the paper, that doesn’t mean that it’s a wrapper for the Java; you’re re-implementing the Java in FORTRAN 95.
STEWART: No, I completely agree. It’s a FORTRAN 95 wrapper for the matrix computations. It wraps around matrices, and there’s no Java in it whatsoever. I wouldn’t have done that. That would just be adding another level of inefficiency.

HAIGH: Okay. You identified this miscellaneous gene of publications, which seems to break down into some translations and histories, and then a more recent surge of textbook activity.

STEWART: Well, as far as the translations, I’ve always enjoyed translating. As I mentioned earlier, I had translated some of Bauer’s papers, to my great profit, because it would probably be for part of my thesis. I’m not a great linguist, but actually translating mathematical papers is sort of a special case because you’ve got the mathematics to lean on. In a sense, it can’t mean anything that’s contrary to the mathematics, and you’ve got it. So I translated some papers.

One of them here is Schroeder’s paper on “Infinitely Many Algorithms for Solving Equations.” This was published in 1870, and was a seminal paper, actually. Alston [Householder] used to say, “If someone claims a result for non-linear equations in a singular variable, and he doesn’t reference Schroeder, he’s probably rediscovered something that’s in Schroeder’s paper.” So I translated that while I was writing my thesis, just sort of to keep my sanity. [Translation of “On Infinitely Many Algorithms for Solving Equations,” by E. Schroder. CS-TR-2990, 1992.]


STEWART: Yes. I can’t remember my motivation. I think someone may have asked me about it. They were interested in selling some stuff that Schroeder had done, and I gave it to them. So I decided to bring it out more formally, and so I polished it up. I didn’t change the translation much. That was done. But I reformatted it. Before it was just typeset, so I had to type it into the machine. I then wrote an introduction, placing the work and its context into the later history of this, and then put it out as a technical report and didn’t do anything with it.

HAIGH: So that was the 1992 Maryland Technical Report version.

STEWART: Yes. Then another one.

HAIGH: And that was translated from German, wasn’t it?

STEWART: Yes, from German. Another that was fun, was more history, but involved translations since most of the papers were in a foreign language, was just a survey of the early history of the singular value decomposition, just sorting out who did what and when and how. It was a lot of fun to write and to track down all of this stuff. I wrote it up. It came out fairly quickly. I sent published it at SIAM Review and dedicated it to Gene Golub for his 15th birthday—because Gene was born on February 29th—he had just had his 15th birthday. In fact, they came back to me, “Don’t you mean his 50th birthday?” I said, “No. It was his 15th.” And so he published it. [On the Early History of the Singular Value Decomposition,” SIAM Review, 35 (1993) 551-566].

The third one, I have some real reservations about. This is translation of Gauss’s later work on least squares. [Translation of Gauss, Theoria Combinationis Observationum, SIAM. 1995].

HAIGH: Yes, I came across the review of this project, and I think it’s what you have in mind with your comment from Frederick Pukelsheim.
G.W. “Pete” Stewart, Oral History with Thomas Haigh - 88

STEWART: Yes. Maybe I’ll talk with you about the review later. Basically Gauss had published a theory of least squares in connection with his groundbreaking book on the motion of heavenly bodies, how to fit and do astronomical observations. Presumably, he used least squares in his calculation of the orbit of Sirius which, it was discovered, fell into the sun and he predicted where it would come out, and where to look for it.

But later on, he gave a completely different justification of least squares, and Laplace had also given a justification; another one, and they were somewhat at odds. He wrote this up in one paper and then Supplement One and Supplement Two. A lot of my interest was in the statistics and in the numerical analysis that was in it. So basically, I taught myself Latin to translate it. You know, it’s just fun doing it anyway, to translate it.

HAIGH: Did that take you a long time?

STEWART: Oh, yeah. Yes, this took a few years. I never got up to the point where I could read something like Cicero. I could read Caesar easily enough, but I never really got to the point where I could read the classics. But again, Gauss is relatively easy to understand, and I translated it. Someone suggested that I send it off to “Archive for Rational Mathematics and Analysis.” Do you know the historian’s name?

HAIGH: So “Archive for Rational Mathematics and Analysis” is a journal. I’m seeing a Springer web page and editorial board. The editors are J.M. Bull and R.D. James. Do you know them?

STEWART: No, these are not.

[Tape 5, Side B]

STEWART: So in any case, I sent it off and this was the first translation. He wrote back, “Nice try, but no cigar.” So I went back and studied more Latin, and came up, and he said it was a pretty good translation. By then I knew enough of the thing. I’d also worked through it and everything. So I mentioned it to SIAM, and they thought that it was a good idea to publish. This was pretty much a mistake to publish in SIAM, because I should have gotten in touch with someone who had connections with historical matters. One of my problems in writing any papers or anything is that I’m the world’s worst proofreader—my papers are full of misprints, and my bibliographies are, and everything else, something that historians can’t stand, and with good reason, and especially there. It doesn’t make as much difference in a mathematical context, because the mathematics provides the correction to a misprint, or something like that. But I’m very bad about that. I just see what I want to see there, when I’m trying to proofread. So the thing was laden with errors of this kind.

But I very much stand behind the translation of it, and some of the observations that I made on it, especially on their observations that actually appeared in a later paper, which I don’t want to get into, basically on how Gauss had used Gaussian elimination. Initially, although he had been using this computational tool, in its initial appearance was originally a statistical tool, a tool for estimating variances of his observations. So as I say, I regret in a way that I wasn’t able to get more help. I did finally find someone to do that and to put it into state. As I say, I believe the translation itself is an accurate reflection of what Gauss wrote.

HAIGH: Then your interest in history and translation in general, and that is a very unusual kind of interest. Do you know of any mathematical colleagues have tried similar activities?
STEWART: Well, there are some. But yes, it’s basically a rather unusual thing with that. But history has fascinated me from the time of high school when I first picked up Gibbon. We would have an edition of it actually in my mother’s room, and I picked it up. Of course I didn't read the whole thing. As a matter of fact, I never have. I’ve got it up there in my shelf, and I managed to get down to his long catalogue of heresies.

HAIGH: You preferred the Asimov version.

STEWART: Isaac Asimov?

HAIGH: Galactic empire.

STEWART: Well, Gibbon’s *The Decline and Fall of the Roman Empire*.

HAIGH: Yes. Well, I think you previously mentioned your admiration of the Foundation Trilogy, which is in many ways a science fiction retelling of the history of the decline and the fall of the Roman Empire.

STEWART: Yes, and as a matter of fact, it was to some extent, deliberate. He wrote a light verse that appeared in the *Magazine of Fantasy & Science Fiction*, in which he basically says how you write these things is, you take the Roman Empire and do it. But the parallels there are very weak. I think that Asimov may have just been using this to get yet another publication. He very much liked to optimize his numbers of publications and refer to them.

HAIGH: So that’s the history. Then you’ve got this series of textbooks.

STEWART: Yes, A*Afternotes*. I can dispose of them briefly. [Afternotes on Numerical Analysis, SIAM, 1996] I was just having fun. I decided that I think that most people don’t know what they’ve taught after they come out of a class. So I said, “I’m going to learn what I taught,” when I came out of this. Basically it was an upper division numerical analysis class, R466. So I resolved that I would sit down and after every class, I would write up what I had said in the class, and not before. I’d prepare notes for the class, you know, not for the students but for my lessons, and then I’d come in and get the lectures, and then I’d do it, although I taught one hour and 15 minutes Tuesdays and Thursdays. Then I had until the next class to get all of this written up. A typical lecture took about eight pages of LaTeX output. And not eight pages that were typed, more than that, of course. So it was a grueling job. It was the kind of thing that if you slipped once, you’re just never going to catch up. But I managed to get through it. I sent it to SIAM and they liked it very much, so I published it. I didn't put any exercises in it so it can’t really be used as a textbook, but I do get complimentary references to it, saying that, “This is a good thing.” And then the second.

HAIGH: Was there a tradition for that kind of a publication?

STEWART: No, I never heard of it being done before. Then I tried it on a graduate course that followed up this undergraduate course, and it was called “Afternotes.” There was a graduate school and did it. [Afternotes Goes to Graduate School, SIAM, 1998]. And I don’t think I’ll ever try this again, but it was fun. So it’s a great thing. And then I guess the last thing on this is the matrix algorithm series.

HAIGH: I think this in a sense was considered an update of your 1974 book.

STEWART: The story was a little more complicated. I had the idea of writing a sequel to the 1974 book, “Introduction to Matrix Computations.” It would cover iterative methods and
methods for large eigenvalue problems, and at the same time, revise the other book to bring it up
to date and this sort of thing. Well, the original book, along with my book with Stewart and Sun,
had been published by Academic Press, about which I have nothing but good to say. They really
were wonderful and helpful and I enjoyed working with them very much.

By the time I got around to looking at this project, Academic Press had been sold to someone
who didn't care about academics. They took a lot of books out of print. Academic Press was in
the old-fashioned thing; “If we make enough profit to keep going on, then that’s good enough,
and we will keep as much as we can in print,” and this sort of thing. Then they got into one of
these profit-maximizing organizations and it tore it apart. Since my book was still selling, they
were very interested in this. But the kind of person they assigned to it, I could see no way of
working with him.

HAIGH: This would have been in the mid-1990s?

STEWART: Yes. So then I decided it would be nice to write this material anyway, so I
conceived of writing a survey of numerical linear algebra. And so this is going to be a four and
possibly a five-volume series. In retrospect, this seems to be hubris of Knuthian proportions.

HAIGH: So you thought that it should be broader in scope, not just in volume?

STEWART: Yes. Yes, it was to cover not only the basic decompositions in eigenvalues, but
large eigenvalue problems, sparse matrix technology, iterative methods. Basically I got the first
two volumes done, the basic decompositions and the matrix algorithms, and the Volume II,
which was on eigen systems. Matrix Algorithms: Volume I. Basic Decompositions. SIAM,
1998]. That was very useful in working, because I actually discovered some interesting new stuff
in the course of writing that. [Matrix Algorithms: Volume II. Eigensystems, SIAM, 2001]. That
appeared in 2001, but by then it was moving into areas that I really wasn’t really expert in. I
knew a lot about the two areas.

By the time we got to sparse matrices, I realized that by the time I taught myself all that was
necessary, that they would probably be beyond my retirement age. Furthermore, what fortunately
has happened is that two excellent books covering exactly the areas that I wanted to came out or
are coming out. One is Yousef Saad’s book on iterative methods. It’s a revision of an original
book that he did, which the material was good, but it was not very well done, in my opinion. But
the revision is just gorgeous. It’s really nice. As a matter of fact, it’s right up here It’s a SIAM
publication and he did a really fine job.


STEWART: Yes. So it would be very hard to compete with this book. And then on his sparse
matrix technology, Tim Davis, who is at the University of Florida Gainesville, is writing a book.
He’s a student of Iain Duff, and he’s writing another excellent survey. [Direct Methods for
Sparse Linear Systems, T. A. Davis, SIAM, Philadelphia, Sept. 2006]. I’ve glanced at this and
there’s really not much point in my going on with it. So I can hand in that project.

The last thing I’m actually working on now, we’ll have to see, I’ve got about half of it written
and it’s out to review. It’s a book called Computer Science: The Science of Computing, which
surveys computer science for people coming into a scientific computer program, with let’s say,
engineering or mathematical or physical sciences backgrounds, who may be very good at
computing, but don’t have a good overview of computer science itself. We actually have a
graduate course that teaches this, and I’m teaching it in the fall, and I’ve taught it three times
before. So this book is a result of that. As I mentioned earlier, I like to teach around in the courses, and this book is the fruit of that.

HAIGH: So would that be a general round up of architecture and complexity theory and things like this?

STEWART: Well, not complexity theory. Architecture, networking, a long section or part on following code from inception in higher level languages all the down to compilation, to making and loading, and things like run time shared libraries, and things like that. So these are all thing that you really need to know about in the course. I’m trying to keep it well pruned to the things that a person doing scientific computing would need to know. I’ve got two more things to write on; networking, and then on parallel computing. So I’m hoping to do that.

HAIGH: Actually, looking through the two volumes of Matrix Algorithms you did complete, I noticed that the style was unusually conversational.

STEWART: Yes. That was deliberate.

HAIGH: I think you have some of architectural issues here too, but let’s me just look for that. You discuss BLAS and you’re got the diagram of a memory hierarchy and the cache and so on. So was that something that was distinctive in your conception of what was urgent to cover?

STEWART: No. It wasn’t particularly distinctive, because these issues had been done before. This was done before in the BLAS 2 and BLAS 3.

HAIGH: Well, that’s true. I can also imagine someone writing a book and assuming that if someone didn’t already know how computers worked they could go and look that up somewhere else. You must have made a decision to try and integrate those concerns with the mathematics and the methods themselves all into one kind of pre-digested package.

STEWART: Yea. No, I felt that that chapter is the updating of the chapter of how the matrix algorithms in my original textbook. Real matrix computations is a hands-on thing, and the thing you’re putting your hands on ultimately is a computer.

HAIGH: Do you have anything to say about to say about the reception of those books?

STEWART: They perceived it to be good. The first got good reviews, and the second did, too. People call me and ask me, or I can refer people to questions for this. It’s a very technical book. It turned out to be much more technical that I really had intended originally. It’s just the nature of the subject. You know, when you write a book, it does take a hold of you, at some point and goes its own way. A book is not something that you create. After a while, it becomes something that you’re talking with.

HAIGH: Then have they acquired the same kind of audiences as the original volume, in terms of specialized courses?

STEWART: No. They’re not really intended for textbooks. They are surveys. There are no exercises in them.

HAIGH: Okay. So that’s a thing that a graduate student would read it to fill gaps in their knowledge?
STEWART: Yes. They would, if they wanted to find out what the efficient use of the QR algorithm was, they would find it here, and other things. So if they were reading a technical paper, for example, they might want to come back to these books and look for background.

HAIGH: I see one big difference between the era of the earlier volume and the more recent ones: the availability of software packages and libraries. Was that something that you addressed or were planning to address?

STEWART: No. This is not a book really on software, as such. Of course, I reference where this software for the things, but it’s not about writing software. It’s really about the understanding of the matrix algorithms themselves.

HAIGH: But you wouldn’t, for example, say, “This method is used in LAPACK”?

STEWART: Oh, yes, I do. There are references to that in the notes and references there.

HAIGH: So you readers where to find an implementation for the method?

STEWART: Yes. Of course for some of them, especially in the second volume, there are no really good implementations. People have just sort of done implementations to prove that they’re on the numerical examples which appear in their paper. Especially in eigenvalue problems, there is a dearth of real quality implementations of things.

HAIGH: Why do you think that is?

STEWART: I’m not sure. I suspect it’s the habits that people got into. The people who were doing the EISPACK type codes for dense linear algebra were implementers, starting with Wilkinson. I think the people that got into the eigenvalue problems weren’t. I suspect that it would have gone quite differently if one or two them had been the kind of people to go out and implement them in quality software. Then we would have seen a lot more implementations. I really think that small things can cause a group to go in completely different ways, and a lot of the progress of a thing are accidental. You’ve got a small group of people starting a field, and it’s almost certain that their proclivities and that their views of things are going to have an enormous influence later on. So I suspect that it could have easily gone a different way, if they had just had different people there. I suspect one of the reasons they didn’t is that the theoretical problems were so much more difficult for the large eigenvalue problems, that people were concentrating deeply on those, and necessarily, that they didn't have time to focus on software. The people that could get into the thing were the people who could do the theory. But this is just a guess, of course, and you can’t go back and change the past.

HAIGH: So I think that concludes those specific areas that were identified for the discussion. I have the standard wrap-up questions at the end. We could also talk about your plans for the future. But is there anything else you’d like to say that you think we’ve neglected?

STEWART: No. I think we can go on. My plans for the future is that I’m in my mid-sixties, and I will be generally running down. I don’t necessarily mean physically or anything, but it’s that time of life when I expect to be doing less and less, and this sort of thing. But even when I retire, even at this school, I think I’ll keep my office and everything. I expect to continue working in this, doing much the same, but there are other things that would be fun to do, too. So actually, part of the thing is that I’ve seen an awful lot of people who don’t retire, and try to work beyond their abilities or dream up great schemes of things that they are going to do, and I never want to
be in that category. I’m very happy to do what I can do well, as long as I do it, and when I don’t, I’ll try something else.

HAIGH: They don’t have any grand projects penciled in, or plans that are in a new direction?

STEWART: No, not really. But I never can tell. I can never tell what’s going to happen. Something just might be right around the corner that I don’t know about. That, as a matter of fact, is what’s been so exciting about this deal; that every day you can have some hope that you’re going to get a new surprise, something new, a new insight. This has really been a wonderful thing about this field. Aside from its breadth, ranging all the way from very difficult mathematics to fun programming. Between those two, it’s been a life in research and then academia that I’ve really enjoyed.

HAIGH: And as you may recall, I like to finish up by asking people to choose one regret and one accomplishment. So it could be one regret that would be in terms of your career, either something that you may have not done personally, in something maybe in the way that mathematics as a field as the whole has progressed. What do you think would be your biggest single regret?

STEWART: I saw that, and I don’t think read right to the end of the interviews you gave me, so I didn’t realize that this was a standard thing. Of course I could point to lots of regrets. From a personal point of view, I could point out that I relatively say I’d had been more political when I was young, and realized the leverage that I had. I could have maneuvered that much better, and played with it much better.

HAIGH: In the academic sense of politics?

STEWART: Well not politics so much, as just touting what I had done better and making better use of it, initially, I think. But every time I think of it as a regret, I think of what I would sacrifice to change it. To change it would have been changing the way that I operate in very different ways. I seldom look back. I’m always looking at the next problem ahead, and that doesn’t make you a very good politician. Again, I don’t want to use this in a sense of a dirty way of perceiving that. There are lots of people who can very well look after their interests, and I admire them—they do great things and this sort of thing. But it’s not me. So, yes, of course I regret not having made full use of what I had done as far as my own advancement. But would I say, I really regretted it? No. It would have changed things in a way that wasn’t me.

HAIGH: So are you going to choose another regret instead, or are you going to stick with that regret, but not really regret it?

STEWART: Well, they all fall into the same category. I can regret that I didn't get involved in sparse matrixes. That was a burgeoning field, and in some ways, I might have done much better if I had jumped in, and not done what I have done, which was mostly in dense matrixes. I considered it. But I’ve done perfectly well with what I’ve done. So the answers are, “Yes, I have regrets, but I’m not sure I would want to be in the place of eliminating them.”

HAIGH: How about the single accomplishment? Is there any special thing in your career that you have been most proud of?

STEWART: Again, I’ve got a lot of “children” with the problems that I’ve solved, and it’s very hard to say which child you love the most. I could name several, but in a way, I have a favorite that I’m extremely proud of the work that Cleve and I did on the QZ algorithm. I think that this
was a real breakthrough, and it just wraps up so nicely. So we can leave it at that. There are many other things that I think I can be equally proud of.

HAIGH: Well, if you want to mention a couple of others.

STEWART: I’m very proud of what came out of LINPACK. That I think we did a bang-up job, even if we had to raise our voices occasionally. The ultimate result was very nice, and I think it helped set standards for the future efforts, and that was very nice in that respect. I’m proud of the body of perturbation theory that I produced, especially because I see that it’s useful. So let’s just leave it at that.

HAIGH: All right. Unless you have anything else to say, that will be the end of the interview.

STEWART: Then that can be the end of the interview. Thank you very much. You’ve been a very good guide.

HAIGH: Thank you for taking part.