An interview with
ALAN C. HINDMARSH

Conducted by Thomas Haigh
On
5 and 6 January, 2005
Livermore, California

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

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

ABSTRACT

Alan C. Hindmarsh discusses his career as a mathematical software specialist within Lawrence Livermore National Laboratory, with particular attention to his work on the GEAR, LSODE and ODEPACK families of high performance ordinary differential equation (ODE) solvers. Hindmarsh obtained a bachelor's degree in mathematics from the California Institute of Technology and a Ph.D. from Stanford where he worked with Charles Loewner. During this time he interned as a mathematical programmer at the Lockheed Missiles and Space Company, Stanford Linear Accelerator Center, and in the Meson Physics division of Los Alamos. After graduation in 1968 he took a permanent job at Lawrence Livermore, and remained there until his retirement in 2002. He discusses the growth of this group, its relationship to users and its use of mathematical software libraries. Hindmarsh also explains changes in technology and patterns of lab funding which led to a decline in its commitment to mathematical software, leading to a dismantling of the central mathematical software group in 1989. Hindmarsh wrote or co-wrote a large number of ODE packages, including LSODE, VODE, CVODE and ODEPACK. His first widely distributed package, GEAR, was based on the DIFSUB routine produced by Bill Gear. His collaborators included George Byrne, Linda Petzold and Peter Brown. Hindmarsh also documents the development of a community interested in writing ODE solver software. In particular, ODEPACK was originally intended as a collaboration between staff members at different laboratories, and Hindmarsh discusses both the original goals of the project and the tensions which prevented its full realization.

Keywords:

HAIGH: This is the beginning of session one of the interview, conducted on 5 January, 2004 in Dr. Hindmarsh's home in Livermore, California.

Thank you very much for agreeing to take part in the interview.

HINDMARSH: You're very welcome. I'm deeply flattered and honored to be a part of this.

HAIGH: Excellent. I wonder if you could begin by talking a little bit about your family background and upbringing.

HINDMARSH: I was born in 1942. My dad was in Naval Intelligence, so we lived in the Washington, D.C. area and in the San Francisco area alternately over the years. I was born in Washington, D.C. We lived there for six years and then California for another four, then back in Virginia again for another four. Then he retired after having a heart attack in 1955. He decided the family needed to go back to California, so we settled on the peninsula—Los Altos, to be specific, in 1956. So I've been in California ever since then and love it here; I certainly prefer living in California to East Coast. I went to high school in Los Altos. My dad was a very strong believer in education. Both of my parents have bachelor's degrees and my father has a Ph.D. from Harvard. His field there was international law. When it was time for me to go to college, he was very forceful about that and wanted me to go to the best school I possibly could. And I didn't know a whole lot about the colleges at the time; I was pretty much influenced by his thinking on that. Caltech was one of the top schools available. I was technically inclined, so that was a naturally choice.

HAIGH: Can you remember when your interest in science and mathematics was developed?

HINDMARSH: Yes, pretty early on, really. I remember probably 8th grade, I was more interested in mathematics than any other subject in school, and it continued that way and it never did change. I was a good student all through the other subjects too, but I never really got hooked on any other subject as much as I did in mathematics, all the way through.

HAIGH: Did you have any technically-oriented hobbies?

HINDMARSH: No, can't say I did. As a youngster I used to make models—model airplanes, model cars, stuff like that. I've always loved music, and that's still a big part of my life. I do choral singing. In high school, when I had free time, well, my dad gave me books to read—math books of various kinds. There's a book over there called, *What Is Mathematics?* like a popularized math book that is easy to pick up and read. So things like that I read in my spare time in high school. I taught myself a lot of mathematics just doing that, and loving it more and more as I went along.

HAIGH: At the time you applied to Caltech, did you know that you would want to major in mathematics?

HINDMARSH: Yes, I certainly did. Certainly did. I didn't know within the subject area what I would want to specialize in, but by the time I finished at Caltech, I realized that I liked complex analysis more than anything else. So that's where I specialized for my graduate work. The

subject of complex variables and what you could do with complex functions seemed to be a very beautiful subject, very elegant. I enjoy that a great deal.

HAIGH: Did that subject area occupy a prominent part in the Caltech mathematical undergraduate curriculum at that point?

HINDMARSH: Yes. Yes, quite a few courses I took there. Well, I took a lot of other courses. I covered the whole area of mathematics though. I mean I covered a lot of various mathematics as an undergraduate. I just seemed to home in on complex analysis later. I don't remember any specific turning point or professor that made a huge difference in that. It was just something that appealed to me.

HAIGH: Were there any classes or instructors in general that you found particularly memorable or influential?

HINDMARSH: As an undergraduate?

HAIGH: Yes.

HINDMARSH: Hmm. You know, not that I remember right now. I enjoyed the whole experience at Caltech. It's a very intense school. In a lot of ways, I felt out of my league. There were a lot of people there that were smarter than I was, and I studied very hard. But I did enjoy all of my math classes.

HAIGH: During your time there, were you exposed at all to digital computers?

HINDMARSH: Not at all. Well, I take that back. I had a friend that was doing a little bit of computing and he took me down to the computer center and showed me a little bit of what he was doing one day. And I thought to myself, "This isn't really something that turns me on," you know? [laughs] At that time, it seemed kind of messy, and I didn't go for it. He was showing me a lot of numerical output of his program, and it seemed a little too messy compared to the theory of conformal mapping or something in analysis. It didn't appeal to me.

HAIGH: According to your résumé, you did spend the summer of 1960 and 1961 at Lockheed working as a mathematical technician as a programmer.

HINDMARSH: Yes, that's true. In fact, really all of my training in numerical analysis and computer programming has been outside the formal schoolwork. The Lockheed experience was very important, actually, for me. My first job there was using a Frieden desktop calculator to do checking of the output of some other program that I knew very little about, but I knew enough to do these long calculations with a calculator and crank out some things, to spot-check some numbers that the computer was generating. There were quite a few of us students there that had these summer jobs at Lockheed, and we were given a lot of free time, of course. It's a good thing because there was no way we could've done this mechanical calculator work all day long. With that free time, we were encouraged to learn FORTRAN, so that's where I learned FORTRAN. I don't know that I ever got to really use it very much in that first summer, but the second summer I did. I actually got into developing a program for ODEs that I didn't do from scratch, but somebody had something that was a dusty deck, so to speak, and it was something somebody

else wanted to get cleaned up and use. So I had an Adams method program to my credit by the end of that second summer, which was very fun.

HAIGH: As you began to program in FORTRAN, did you feel that this was something that you were enjoying and satisfied you?

HINDMARSH: Yes, I did. I did, because I was learning the mathematics behind the methods, and that seemed to be quite powerful and interesting. And I was making a contribution. There was one early experience, I can't remember which of the summers it was, but I was assigned to look over a program that somebody else had written with a Runge-Kutta method in it for many ODEs. And it had to do with tracking missiles of some kind or another for the Air Force, and I took what I knew of that method and looked at the program, and I said, "I don't think this program is right. I found a bug in the program." And I took it to my boss and he looked at it, and he said, "You're right, there's a bug in the program." So he quickly sent a memo of some kind off to the Air Force, because this program had already been delivered to his funders at the Air Force. So I got credit for a small, but significant improvement of a program there.

HAIGH: Do you remember what kind of computer they were using?

HINDMARSH: In those days they were using IBM, probably, I don't know, 704s, something like that.

HAIGH: Or could be a 709 by then.

HINDMARSH: 709? Yes, that sounds right.

HAIGH: As a programmer, did you approach the machine physically and put your cards into it, or was a staff of trained operators who would do that for you?

HINDMARSH: Yes, it was a batch processing system of course, and I think we delivered our card decks to somebody who did the input for us. We got to run the machinery once or twice, but that was as close as we got to being hands-on with it.

HAIGH: Did you punch your own cards or would they punch them for you?

HINDMARSH: Yes, I think we had both available. We had punch card machines that were available to us, but we also had the punching service that was available to us. I remember writing out the bulk of the programs on programming sheets and turning those in, and then we would do our own corrections.

HAIGH: Do you have a sense of about how long it would be until you got your output back from the job?

HINDMARSH: [chuckles] It was at least a full day, I know that. Of course we weren't doing anything that was very time pressured in those days, as far as our programming. The importance of what we were doing as student employees was pretty far down on the priority list, so it didn't really matter too much to anybody that it was a full day or more before we got our answers back. But the experience was great.

HAIGH: A last question on the Lockheed work. During that point, do you know if there was a strong division enforced between programming and analysis work among the permanent staff?

HINDMARSH: As far as mathematical software is concerned, you mean?

HAIGH: Yes, I was wondering, did they have the idea that the mathematicians should just write some specifications and that a separate team should program? Or were the two tasks more intertwined?

HINDMARSH: Yes, they were fairly all intertwined. I was certainly encouraged to be expert in the mathematics behind the program as well as be responsible for the program. As a matter of fact, I pulled out a document that was generated back in that second summer, so it's dated 1961. This is a document on that program, that Adams integration program that I worked on then, and a big part of that is the description of the mathematical formulas for it. The software itself was pretty primitive by modern standards, of course, but it was certainly appropriate for the time. What we tended to do in those days with a program like that was to present it to the user in pieces, and the user then needed to put these pieces into his larger application program so that everything worked together. It wasn't so much a black box with lots of ways to alter inputs and options from an external position.

HAIGH: In this case, were you aware of what the application was that it was to be used for?

HINDMARSH: In this case, I don't think I was. I was actually asked to do this from a general setting, a general point of view, which I enjoyed. That was my first experience with the idea that a mathematical piece of software, software that was specified in purely mathematical terms, could exist in its own right, even though the user interface was a little difficult. But then that could be applied within an application program that could be from any discipline—physics, chemistry, whatever.

HAIGH: Do you recall whether at this point Lockheed had anything that could be called a library of mathematical functions?

HINDMARSH: I think they did. This was entered into a collection that was made available to the user community there. It's got an identifier that was used in their listing process.

HAIGH: It looks quite formal. Do you remember if they had strict rules for documentation?

HINDMARSH: Yes, somewhat. At least the names of the sections of the document and all are pretty formalized. We had to spell out what each block was for and how it was used. And we had to write flowcharts—there's a flowchart at the end of this, too. That was big.

HAIGH: Then for the summers of 1962, '63, '64, and 65, you were a mathematical analyst and programmer at the Stanford Linear Accelerator Center, SLAC.

HINDMARSH: Yes, yes. That was a completely different experience, for the most part. I did a lot of different things there. It was exciting to be in a physics lab, for one thing. I got into that when the name SLAC had not even been decided on. They called it "Project M" for "Monster." The very first summer I was there, I was doing something not related to ODEs at all but that had

to do with alignment target designs that had to do with the laser device they were going to use to align the accelerator tube. And then another summer I worked with a fellow named Karl Brown, and he was an expert on the transport of particle beams through electromagnetic fields, or what they call quadripole magnets. So I did a lot of work on analyzing these magnet's properties, more optics than anything else. Then in a later summer, I did get back to ODEs, working with Karl Brown. Again, it was a matter of designing or upgrading a general purpose program for general use, Adams method again, as a matter of fact. I don't know if I have the program that came out of that. I left it there and I left it in a library. There was a quite well organized formal library of mathematical software that was being developed at SLAC. So I left my program there. I remember I think the last day of my last summer there, having a long session with the gal who was sort of in charge of that library and getting it incorporated in there. That, again, reinforced my idea about the power of mathematical software.

HAIGH: How would you contrast the work environment at SLAC from the one that you had experienced at Lockheed?

HINDMARSH: That was a big difference. Things were very regimented at Lockheed, and the profit motive was always very clear. As a matter of fact, at my second summer at Lockheed, they had a budget crunch and they had to let go all of the summer hires in the middle of August, so I was only there half a summer that year. Stanford, on the other hand, of course is an academic environment, even at SLAC, which has its own mission. And especially in the early years, when there was a fairly small group of people finding their way into the process of building this two-mile accelerator, it was very interesting to see the informal interplay of the physicists and the computer people and the mathematicians and others. Again, as student employees, we were encouraged to take part in the lecture series that were offered and learn all about what was going on. So that was great fun.

There's a little sidelight to the connection with Karl Brown there. When I was at Livermore, probably about seven years ago, we had a new hire in our group who's name was David Brown. Sometime after he became a member of our organization, I got an e-mail from him that his father died, and he sent some information about his father. Well I realized only at that point that his father was Karl Brown, and I hadn't known this. In fact, David Brown was my group supervisor for a couple of years. So I had that kind of reconnection with that whole SLAC period through Karl Brown's son, which was very pleasant.

HAIGH: Were there any other experiences or relationships from this period that exerted influence later in your career?

HINDMARSH: You're talking about the SLAC years and the summer hires?

HAIGH: Yes, I think '60 through '64, '65 period. Around your undergraduate days.

HINDMARSH: I don't think there was any one experience or any few experiences that made a big difference. The whole business of being assigned to do general purpose software had a big influence on me. Karl Brown, in particular, was very encouraging about that, although his background was physics and he was famous for doing a lot of other things that had nothing to do with software.

HAIGH: Did any of those experiences start to influence the way that you thought about mathematics while you were doing your courses at Caltech, or did the two worlds stay quite distinct?

HINDMARSH: For the most part, the two worlds were pretty distinct for me. That was still true through graduate school. When I was spending summers at SLAC and going to graduate school at Stanford, I was learning a lot of mathematics and a lot of analysis in particular, and enjoying that. But I was enjoying the software work in the summers at least as much and not really being able to connect the two at all.

HAIGH: Can you remember at what point you first decided that you wanted to go into graduate school?

HINDMARSH: I think that really wasn't a decision. That was just something in my family environment that was just taken for granted: that I would because my father had, and I was in the scientific discipline, and you just didn't consider not going to graduate school.

HAIGH: So you never seriously considered any alternative path during your undergraduate base?

HINDMARSH: No, not really. Not really. My father was disappointed that I wouldn't go to Harvard. And he was a little disappointed that I didn't become a lawyer, too. That was his original plan for me, but he was okay with my going to Stanford and being a mathematician.

HAIGH: Do you think a lot of the mathematics students at Caltech would then have gone on straight to graduate school?

HINDMARSH: Yes, I think most did. Most did.

HAIGH: How did you choose Stanford?

HINDMARSH: I applied to several places, and I was really influenced more by geography. Among the places that I was accepted at, I was influenced because I love California, the Bay Area, in particular. Stanford accepted me and it was certainly a very good place to be as a mathematics graduate student. I knew that they had strong people in complex analysis, too, which was at that time where I wanted to be academically. I think even at that point, I was pretty sure that I didn't want to stay in academia after graduate school. I had made that decision fairly early on.

HAIGH: Why was that?

HINDMARSH: I could see enough of people that had been too isolated in an academic life, and I wanted to be more out in more of the real world and I wanted to make more of a difference. I had already seen a lot of the outer world in the summer jobs, and I could see that I was more suited to making a difference that way, in an environment like one of the national labs, than I would be in an academic position. I could see other people around me at Stanford, especially at Stanford, being a lot brighter than I was, frankly, in academic research. My own subject area there was relatively much more down to earth than a lot of the abstract mathematics that was being done. I really just wasn't able to get excited about the more abstract topics that people were doing.

HAIGH: Were there any courses or instructors there that you did find particularly stimulating?

HINDMARSH: At Stanford?

HAIGH: Yes.

HINDMARSH: Yes. Karl Loewner was my advisor. He was Bavarian originally. Still had a heavy German accent. And he was a delightful old man. He taught the freshman seminar, which was kind of a problem-solving seminar. It was not very well structured, but he was just delightful to listen to and work with. He was very soft-spoken, but very instructive and very encouraging, and I liked him right away. So my contact with him began right away in the first year as a first-year graduate student, and I ended up choosing him as a thesis advisor, and he was willing to take me on. He had a number of areas that needed some work, so the connection just seemed to fall into place in a natural way. He had actually written a groundbreaking paper in I think 1923, and it had to do with something called the Bieberbach Conjecture. His work related to that opened up a whole set of questions in analytic function theory that I found interesting. Finally there was something I could do there. I didn't do anything nearly as revolutionary as what he had done himself, but he encouraged me to get some results that were enough to get a thesis. I was happy with that.

HAIGH: Did you ever publish your thesis work?

HINDMARSH: I published one of the main results in it that came out in 1968. Something about Pick's Condition. [Pick's Condition and Analyticity, Pacific J. of Math., 27 (1968), 527-531].

HAIGH: I've seen that. So it was your first published work.

HINDMARSH: Yes, that was. Now Karl Loewner died the last year I was there; he died in I think January of 1968. I think he was 73 years old.[1] He had been in perfect health as far as anybody knew. Apparently, he had a weak heart, but it was very shocking to hear that he died.

HAIGH: Did that make defending your thesis somewhat hard?

HINDMARSH: It did. I was switched to Professor Halsey Royden. Royden's interests were a little different; his interests were more abstract than Loewner's. He agreed to see me through to the end of the process, and that went okay; it went pretty well.

HAIGH: Could you briefly describe the intellectual contribution made by your thesis?

HINDMARSH: You know, if I had to read my thesis again now, I'd probably have a tough time. It's long.

HAIGH: Were there any relationships with fellow students that proved important later in your career?

---

[1] Loewner was born 29 May 1893 in Lany, Bohemia (now Czech Republic) and died: 8 Jan 1968 in Stanford.

HINDMARSH: No, there weren't. Most of the fellow students of Loewner's, or students in the Math Department generally, went onto academic careers, as far as I know. I only had a little bit of contact with any of them afterwards, but nothing that really amounted to much.

HAIGH: In summer of 1966, you worked with Meson Physics Group at Los Alamos.

HINDMARSH: I had one summer there, and that had to do with the design of a meson accelerator at a meson physics facility there. I had a friend named Harold Butler that lived in Los Altos when I was going to high school there. He went to Los Alamos at some point and he had known me as a youngster, and he contacted me somehow or other and invited me to come to Los Alamos for a summer, and that's how I ended up doing that. I didn't work with him so much while I was there, but he was kind of my host there, in a sense. I worked with some other people on some mathematical software—not really mathematical so much at that time. But it was exciting in the sense that it was a lot of cutting edge particle physics at that time and I was overhearing a lot of that, and it was kind of exciting to be just a part. I wasn't up to the physics, but I was writing programs to do modeling of the particle beams and the accelerator that they were going to build. So that was exciting. I left behind some programs that got used for some time.

HAIGH: Had that been your first contact with a national lab?

HINDMARSH: Yes it was, yeah.

HAIGH: Do you think the experience there influenced your later career?

HINDMARSH: Yes, it definitely did. I liked the atmosphere there at Los Alamos. It was a little bit academic and informal in a lot of ways, a little nonacademic in that sense, and I liked the mix of that. It was a big laboratory with a lot of things going on, and I liked that aspect. A lot of computing power, a lot of expertise in a different areas, so that if I got stuck and I needed help on something, I could go to somebody. So, yeah, I liked that aspect.

HAIGH: Were you required to get security clearance for that?

HINDMARSH: Yes. Oh yes. That wasn't a problem for me, but I did get introduced to the whole world of clearance there. That was interesting, just to know that some people could get it and some people couldn't, and that was an issue.

HAIGH: As you began to near graduation from the Ph.D. program, were there any specific career options that you were considering?

HINDMARSH: Well, I wanted to do mathematical physics or mathematical software or some mixture of those two things. I wasn't really too specific about what that would be. I knew that I didn't want to be in an academic position, and so that left the whole area of private industry and the national labs open. I did apply to some private industry slots when I was doing job applications, as well as Livermore and Los Alamos Laboratories. I was never really impressed with any of the visits that I made to private industries when I went for job interviews. Some of them were not impressed with me either, I have to say. I wasn't what they wanted, and in most

cases, they weren't what I wanted. One of them actually pursued me pretty aggressively, but I didn't see a fit for them.

HAIGH: At that point, would those have been companies like Bell Labs or General Motors, those companies with large research centers.

HINDMARSH: Yes. Those weren't among the places I applied to, but that kind of place was what seemed to be the right thing for me. So when I did apply to Los Alamos and Livermore, I got accepted to both, and I had a long debate with myself and involving my wife, about which to go to, and ended up again being influenced strongly by geography and wound up staying in California. Los Alamos is an exciting place, but it's very isolated. The summer visit there was enough to find out that that can have some really negative effects. It's not so much now, but at that time, 1968, there was not much else going on around Los Alamos and you had to spend quite a bit of time to get to somewhere else where there was something other than the laboratory, any kind of other cultural activity. Although Livermore might have seem isolated, too. It's certainly a lot more connected with other places that have plenty to do.

HAIGH: When you were hired at Livermore, what was your position and what were your responsibilities?

HINDMARSH: I was hired as a Mathematician, and that's the only title I've ever had there from 1968 till I retired in 2002. My initial assignment was to work with a group that was doing Monte Carlo neutronics, which is a matter of picking random numbers to direct the paths of particles in a simulation. I got to be fairly good at that, although I didn't have any background at all in it, but I did a lot of self-study and learned what that game was all about in the first few years. It didn't take me very long to realize that that wasn't the place for me in the long run. I didn't have much of a statistics background, and there wasn't any analysis going on in that work, and analysis was my forte. Monte Carlo business was fun for a while, but it wasn't going to last.

HAIGH: How were things organized there? As a mathematician, were you part of a central mathematics group?

HINDMARSH: Not right away. They didn't have any group of mathematicians when I first got there, so I was just kind of part of a computer programming and mathematics pool—a pool of people who did mathematical programming for the physicists. That changed pretty quickly, especially with the addition of me. There got to be a critical mass of people that least were interested in doing mathematical software work and numerical analysis. As a result of that, a group was formed in 1969, so it was only a year after I got there.

HAIGH: What was that group?

HINDMARSH: It was originally called the Applied Numerical Analysis Group—ANAG. The first head of that group was a man named John Killeen, who was a physicist by training. He worked on magnetic fusion modeling, which has always been a big subject at the lab; has always been and still is. But he had a very strong interest in mathematical modeling and computing, so he headed up the group. Now he was also still very busy, and so he didn't have a lot of time for the running of this group. He had a deputy whose name was Fred Fritsch. Fred really ran the group, and pretty soon, Fred was not just the deputy, he was the group leader. Fred was a

mathematician. He was a student employee for a number of years, but was a PhD mathematician. He was a great supervisor to have for this group and he was for many years. He's retired now, too, but still around. So that group formed in about 1969, and for the next 20 years we had a great thing going. We had a core group of people who were interested in the discipline of numerical analysis and mathematical software and educating users about that area, and being a resource, a central resource.

HAIGH: Did being part of this group mean that people stopped spending so much time helping users with their day-to-day programming needs, or was it something that you would do for a portion of your time?

HINDMARSH: We didn't stop doing that by any means. No, that was definitely a part of our mission and we just incorporated that in. The mission of the group evolved over time, but it got to be a mission with a whole spectrum: research in numerical analysis, development of software associated with that, in-house education in a formal sense, as well as informally talking to groups of users, and certainly being available as a walk-in service to anybody around the lab who had problems involving mathematics of any kind, whether it was software-related or not.

[Tape 1 of 3, Side B]

HINDMARSH: I don't think I answered part of the question about how it was connected with the organization. The math group was formed within the computation organization, and it's always been that way. It's always been a little odd for us to be compared to the non-mathematicians in the group, people who were doing strictly programming or strictly computer operations-type work or other computer-related activities—compiler writing, for example; that was a big activity there for a long time. So we always felt like we were sort of oddballs in the whole mix. Frankly, the higher-up administrators did not always appreciate what we were doing. As a group, we always felt we knew what we were doing and we had lots of contacts out among the programmatic divisions at the laboratory—among the physicists, the chemists, the biologists, the engineers, the whole range—and we felt pretty good about our justification for our existence in terms of serving that community of scientists. We always felt undermanned; we always were trying to hire people to fill in gaps to cover the range of mathematical subject areas, and bring in software to cover the gamut too.

HAIGH: How large was the group over time?

HINDMARSH: In those good years, it grew to be twenty-five to thirty people.

HAIGH: How large was the computing operation as a whole within Livermore.

HINDMARSH: Two hundred and fifty to three hundred and fifty people in the department. The names are a little confusing compared to academia. You have departments, which are a few hundred people; then you have divisions, which are fifty to one hundred people; and then within divisions, you have groups, which are on the order of ten people. So we started out as a group, and as we grew, actually they have the term "section" for intermediate sizes; we became a section and then we became a division. Within that division we had groups more specifically devoted to mathematical software, groups devoted to statistics and probability, groups devoted to applications. So we grew to be quite a comprehensive organization over those years.

The name kept changing. There's one place that I listed all the names that we had, or at least the initials. We were the Applied Numerical Analysis Group; then the Numerical Analysis Group; the Numerical Mathematics Group; the Numerical Mathematics Section; the Mathematics and Statistics Section (that was when we had a good statistics component); then the Mathematics and Statistics Division, moving up; then the Computing Research and Development Division (that didn't last long because CRDD became "crud" when you pronounced it); then the Computing and Mathematics Research Division; and then the CCSE -- Center for Computational Science and Engineering; then the final name that still exists now, is CASC; it stands for Center for Applied Scientific Computing. So a lot of names.

HAIGH: Were other parts of the lab changing their name with such frequency?

HINDMARSH: I don't think quite so much, but there were a lot of changes and a lot of reorganizations over the years.

HAIGH: How would you describe the atmosphere at Livermore as a whole, during those early years?

HINDMARSH: In those early years, it was very pleasant for me, for our group, in the sense that things were quite informal. We had a kind of an academic type of atmosphere in the sense of being allowed to do our own research and have some freedom to choose the topics of our research and how we went about doing that. We didn't have a lot of pressures from higher-up as to deadlines and things like that, although there was some formalism for doing annual appraisals and making sure that we were producing, of course, and that what we were doing was connected to the programs at the laboratories. But it was all quite loose, semi-academic, and that I enjoyed.

The other important thing about those early years is that we didn't have to worry about where our money was coming from. They had a mechanism for funding our group and other groups like it that were service-oriented. It was very simple, and it worked just beautifully. The central Computing Center would charge all of the programmatic efforts that used the computers a certain fee off the top for using the computers, and that was always set in such a way that the money that was collected would fund the service groups. We would get our share, and we wouldn't have to worry about whether we were going to be funded no matter who it was we served around the laboratory. It wasn't perfect in the sense that the people paying were not always exactly the ones who were being served by us; there were some imbalances. But there was no way, really, to be any more perfect in that sense, without a lot of bureaucracy and paperwork that would not have made sense. So in fact some of the organizations around the lab that were less affluent in terms of their funding from the Department of Energy got a lot more than their share of service from the Math organization than some of the more well-funded programs. In particular, the weapons program, which has always been of course the dominant part of the laboratory, paid the lion's share in what they call the computer recharge money. They probably got far less than their proportion of services from us because they tended to be self-sufficient. The Weapons Physics programming groups were probably the last ones to appreciate the value of our work.

HAIGH: Was that because the weapons work would take part behind closed doors in a more secret part of the lab?

HINDMARSH: It was in part because of that, and I think it was just a whole culture, that you still see around now, although it has changed a great deal as computing environments have changed. But the culture was that the physicists knew everything, and if they knew the mathematical method for something they could look it up in a book and program it themselves and they didn't need us. That didn't really change until I would say the late '80s probably, when the laboratory considered it too expensive to maintain its own compiler writers. They wanted more standardization; they wanted to use vendor-supplied compilers and more standard software. So they had to do code rewriting at that point, and they needed to be able to get portable software into their big physics codes; then they got more interested in us.

HAIGH: Did people in the mathematical group ever feel any kind of moral qualms about the weapons programs at the Lab, or was that really offstage as far as you were concerned?

HINDMARSH: Yes, some did more than others. I frankly wasn't excited about working with a weapons program so much. I was always one that was very excited when the laboratory got non-weapons-related energy projects in the Carter years. That was happening, to a large extent, with a lot of new exotic energy sources being looked at. There were liquid fuels; there was oil shale; there was coal gasification; and of course the magnetic fusion and laser fusion came in there too in a big way. I was always excited to see those. I'm glad to see that the lab was getting involved in those things, because the lab and other DOE labs were the natural place for that kind of work to happen. On the other hand, when Reagan came in and during the Reagan-Bush years, the philosophy at the federal level was that private industry can do that kind of effort. I think they were wrong, but that was their philosophy, and so that kind of thing declined at the lab. It still happens, but to a lesser extent.

HAIGH: Were there any areas within the lab that stood out as being particularly important in collaborating closely with the mathematical group?

HINDMARSH: In general, yes. Different people in our organization collaborated with different other groups. I collaborated for a long period of time with people in the Atmospheric Sciences Organization. There was a big effort, particular during in the early '70s, to do modeling of the atmosphere from the point of view of atmospheric pollution. A particular project that I got into was the climatic impact of supersonic transports, which this country was talking about building at the time. We knew that the British and French were doing it. The question of what would be the impact on the upper atmosphere of the exhaust from supersonic transports was something that the lab was in a position to be able to go into, and it was a very important question. So a lot of the physics and chemistry and mathematical modeling was done just centered on that one topic. That happened to be a large motivator for what I was doing in the mathematical software area—for ODEs in particular. That was early '70s, so that was probably the biggest thing that got me into doing ODE solvers in those early years.

HAIGH: How would you characterize the general state of software libraries within Livermore when you joined the lab in 1968?

HINDMARSH: It was actually pretty good. There were several people who ran something called the Computer Information Center, CIC, and they had little reports on each of their different items that were available, and those were collected in libraries, that you could get a hold of. There was a strong emphasis on source form for libraries as opposed to binary libraries because there was

this feeling that you probably would need to change the source as you incorporated these into your application. So people wanted source. One of the very first contributions that I made to this library was actually a rootfinding algorithm, just finding the root of a real-valued function of a real variable. Seems like a simple problem now, but that was something that came out of a research project that I did. So I wrote a rootfinder and put it in the library. I don't think it was used very much, but it was a good exercise for me.

HAIGH: Do you know if at that point, any of the functions in the library would have come from outside the lab or were they all introduced internally?

HINDMARSH: I think they were pretty much produced internally. Until our math organization got going, people just were not inclined to go out and look for what was available. Maybe there wasn't much available anyway, if they had looked. So people tended to write it, by identifying math available in textbooks, or in journals or academic papers, and writing the software from that.

HAIGH: What was the main machine at Livermore in those days?

HINDMARSH: You know, I didn't keep track of machines very well. Hardware was never my thing. But they had a lot of CDC machines in those early years before CDC went out of business. CDC STAR was an experimental machine. "STAR" stood for "String Array Processor." Well, I guess before CDC they had IBM machines. When I first came there, I think IBM was predominant, and then CDC came along after that. And then the Crays started coming in. Yes, there was quite a succession of machines, both large and small. Of course, all those early machines tended to be fairly large, at least the ones that Livermore bought, because the needs were large.

HAIGH: Did you begin to see a change in the '70s with smaller groups being able to install their own computers?

HINDMARSH: Yes. Decentralization of computing resources happened…I guess it started in the '70s; it really got going in the '80s. That really is what ended the good times for us, unfortunately. That meant that the computer recharge money that we were getting kept going down little by little every year, because the computer users were getting their own little machines and not paying that fee. So little by little, we were asked to then go to find sources of money outside the Computer Center, out in the programs. First it would be like 25% of our money would come that way, and pretty soon it was 50%. It wasn't long before it was about 100%.

The result of that was a decline in our organization, too, because we couldn't always get the money that easily. We worked at it, but that took a lot of time out of our other activities. We had to spend a lot more time justifying what we were doing and targeting narrower audiences. We couldn't say, "What we want to do for such-and-such group is develop a solver, and that solver is going to take us several man-years of effort." Well, they couldn't afford to pay that, even though the software that would result from that effort might be used by a whole wide variety of groups after it was done. We couldn't make the argument that the one group we were targeting should pay for all of that development when they were only one of many beneficiaries of it. It just wasn't practical to go around and collect from all the potential beneficiaries of a software effort.

So it made the whole idea of general-purpose mathematical software non-tenable after the late ‘80s.

So actually the result of that for me was that in 1989, I lost my job completely. I had no more funding to do general-purpose mathematical software development and user consulting, and I was on my own as far as finding other sources of funding. I did manage to connect up then with a group that had been one of my big users in the laser program; I suddenly had to go over begging, and I got half of my funding from the laser program at that point, and I got the other half from some other overhead source that was available. But from then on, I was doing more a collection of odd jobs and was not really free to do the kinds of things that we had been doing for the previous 20 years. So that was a big change.

HAIGH: In the ‘70s, as externally produced packages in libraries, such as EISPACK and the NAG and IMSL libraries entered wider usage, were those incorporated into the Livermore software repertoire?

HINDMARSH: Oh, absolutely. One of the first things that Fred did as our supervisor was to get us all in touch with what was going on in other labs and in academic places where software was being researched and developed. Yes, we had LINPACK and EISPACK and MINPACK and other software from various places incorporated as quickly as we could. Those were great models for the other collections that we were developing ourselves.

HAIGH: Would you say that the availability of these externally produced routines changed the kind of work or the way that the mathematics group would approach things?

HINDMARSH: Yes. It changed the approach from doing strictly internal programming to getting these externally available things, and it certainly raised the level of library building to a higher art. We had a very strong effort to build a source library that was available to the whole user community on all the machines that we could possibly put it on. We didn’t have an Internet then, of course, so that it could be put in one place only and distributed. So instead it had to put on each machine. But it changed the way we interacted with users in the sense of saying, “Even though this wasn’t written here, this is a good program. We’ve looked at it; we’ve looked at the evaluations that had been done on it and the test results on it, and it’s good.” So we had to educate users that there was good stuff that wasn’t written at Livermore.

HAIGH: Would you ever make any changes to the code from these externally produced systems?

HINDMARSH: Yes and no. In the case of things like LINPACK, no; we would certainly take that as is. That was not something we would change because it had gone through such a thorough development and testing phase. It was so highly polished, we wouldn’t touch it. The EISPACK experience was a little bit different in that we were actually involved with not only incorporating what had been written at Argonne but then writing some drivers on top of it to facilitate the user interface to that collection. But again, we didn’t modify the underlying basic routines in that collection, but we would write it on top. There was a lot of other software that we imported, though, individual programs for different things, that needed work. Sometimes, it needed a lot of work. So the ODE solver that I first got a hold of from Bill Gear was certainly in that category.

HAIGH: I just would have one more question about the way the software was incorporated. Would you try and integrate all these pieces into a single collection, so that as far as the users were concerned, they didn't have to worry about where they'd come from? Or did the routine stay distinct from the user point of view?

HINDMARSH: We would collect them in a way that the users didn't have to worry about what individual outside institution they came from. But they did have to be concerned about different levels of portability. We divided our software into classes. I remember we had Classes 1, 2, and 3, and partly this was based on software quality, partly based on the level of support that we were willing to give to the software. So Class 1 was software that we knew a lot about, we were willing to support, we knew this quality was good no matter where it came from—whether we wrote it or brought it in, we would stand behind it. Class 2 is kind of in between. Class 3 was stuff that we got, but we didn't want to spend a lot of time supporting it, and it was "Use at your own risk." But we'd make it available. So we divided things that way.

HAIGH: And those were official terms that would show up on the lists of available packages?

HINDMARSH: That's right. And we did develop an access utility routine that would let users get all of this stuff, and right away they would see which class it was in. We had the commercial libraries in there, too. That was a big part of the mix. So IMSL and NAG were in there, and they were put in Class 2 because we were not in a position to support them to the same level as the Class 1 stuff because we didn't write it and we weren't allowed to change it if we found bugs. But we felt the quality was up there, so we put it in Class 2. The commercial libraries were heavily used, but they disappeared at some point in the '90s when it was decided that they were just a luxury we couldn't afford any more. Couldn't afford to maintain them.

HAIGH: The externally produced commercial libraries?

HINDMARSH: Yes, like IMSL.

HAIGH: Oh, so the subscriptions were canceled.

HINDMARSH: Yep, yep.

HAIGH: That sounds like it might be a false economy.

HINDMARSH: Oh, it was. No question about it. That happened in combination with a number of other developments in '96, but that's getting a little ahead of the story.

HAIGH: An issue that would relate to that then: Other than the difficulty that you mentioned, with the weapons people and physicists wanting to do their own codes, would you say in general that it was hard to persuade users that they should use standard library components instead of writing their own?

HINDMARSH: Outside of the weapons physics people, it wasn't. It was fairly easy. And we really enjoyed that process of getting the various groups to know that we existed, that we had stuff they could use; to know that we would go out and help them write the calling sequences to this stuff and get them to use it. That was really great. There were a lot of great success stories.

One of the things that we did as an organization was to document the contacts we were having with all the groups around the laboratory, and I've got one example I brought. I don't have the whole thing, but at the end of each year, as a division, we would make a list of the consultations, whether they were software-related or not. These were consultations we had with all the other people outside computation. This is a sub-list of my contributions for fiscal year 1984. It is six pages long with about 20 items on each page, showing all of the individual contacts with groups all over the place: mechanical engineering, electrical engineering, chemistry, chemical engineering, all of it. So we prided ourselves on how widely our expertise and our software was being used out there.

HAIGH: Were there any cases where people went from user groups and then became more interested in joining your mathematical group within the lab, or did you hire purely externally, to recruit people with Ph.D.s?

HINDMARSH: There were a few instances of people coming into our group because they were really generalists at heart and they wanted to be doing the kind of thing we were doing. But the majority of our growth came from outside hires. We were always looking for people, as I say, to fill the gaps. We started out pretty strong in ODEs and special functions, but we'd say, "Well, we don't have many people in the PDE area or approximation or statistical software," so we'd go hire people to fill those gaps.

HAIGH: Were there any other people who joined the group who you feel made particularly distinguished contributions?

HINDMARSH: Oh, I can't single anybody out. No, I wouldn't single anybody out. We had a lot of good contributors all across the board. We also had good people who were not so much contributing in any one area, but were good at organizing the library aspect and the utility routines to make those libraries available to people.

HAIGH: To ask a slightly different question, were there any pieces of software produced within the lab in particular areas that went out into the world and were widely used elsewhere?

HINDMARSH: Well, I guess the ODE area, the ODE software, are the main examples of that. To go ahead a little bit in the story, when LSODE came out after its long and somewhat tortured history-- The "L" stands for Livermore; it's Livermore Solver for ODEs. So in a sense, that piece of software put Livermore on the map, I think, in a lot of ways.

HAIGH: So the ODE packages were the only ones that really made a name for themselves beyond the lab.

HINDMARSH: Let's see. I think that's probably true.

HAIGH: All right, well let's move on to discuss them, then. I believe that you wanted to talk about the experiences with some of Bill Gear's software and bring them back into the light.

HINDMARSH: Yes. One of the fellows in the Physics Department, or he was partly in the physics department, but then he was partly in the atmospheric sciences group --- was a fellow named Bob Gelinas. He was working on chemical kinetics problems, chemical kinetics models

of the upper atmosphere. This was all eventually connected with this atmospheric impact project I mentioned earlier. And he was trying to integrate these kinetics equations with what was available. He would come to me about it, and we were both realizing that things were just not working very well.

He had heard about this business of stiffness and about Bill Gear. This was back in 1969, 1970. I didn't know anything about that subject area, so we were both just kind of feeling our way into this. We went to a talk that Bill Gear gave at NASA Ames, Moffett Field. So that's where I first learned about what he was doing. At the time, he was working on a sabbatical, I believe, at SLAC. He had just briefly spent some time at Argonne when he was based at the University of Illinois. But during those two periods away from Illinois, he had developed this piece of software for stiff differential equations. The more we heard about it, the more we decided that's probably the thing we need for these chemical kinetics equations. So we got in touch with him; he sent us a card deck. I remember getting a whole box of cards with his program, and it was called "STIFF." That was the name of the subroutine. When he published it himself, it was called DIFSUB, and I don't know where the name change occurred, but we always had it by the name of "STIFF." So we got this thing and took a look at it, and right away, noticed that it was going to need some work. It wasn't a very portable piece of software. I happen to have here a listing of the thing as we first got it, and already it's got a bunch of markings on it about things that we could see we were going to have to change. Also, one of the things I spent a long time on was generating a flow chart. Here's my flow chart; it's stamped 1972. I have a copy you can have. So I spent quite a while in 1969, '70, '71, '72 figuring out what this business of stiff equations was all about, and about the Backward Differentiation Formula methods that Bill Gear had developed into this software. So I became the local expert on these methods and on the software.

HAIGH: Now, this idea of stiffness: I got the impression from reading your 1986 survey paper that this is rather a subtle and potentially elusive quality that only comes to light when you attempt to solve them. [Current Methods for Large Stiff ODE Systems, in Numerical Mathematics and Applications, R. Vichnevetsky and J. Vignes, eds., North-Holland, Amsterdam, 1986, pp. 135-144].

HINDMARSH: Yes, that's right.

HAIGH: You can't tell just by looking at something; that it might even depend on the parameters that you plug into something.

HINDMARSH: Right, right. You can't see it in the solution if you look at plots of the solution. The solution doesn't look any different than it would for a non-stiff problem—for most of the time, anyway. The thing about stiff problems is that they are characterized by a very strongly damped mode within the problem. The term "stiffness" actually came from a paper by Curtiss and Hirschfelder back in 1952, which is frequently cited. And they were thinking about, I think, servo mechanisms. [Curtiss, C. E., and Hirschfelder, J. O., 1952. Integration of stiff equations. Proc. Nat. Acad. Sci. U.S. 38, 235-243].

HAIGH: Would you think that this would've been a mechanical differential analyzer?

HINDMARSH: It could be, could be.

HAIGH: You trace a curve using something like a pen on a graph paper, so I imagine it would allow you to feel how the machine was responding. And then the second arm on another table draws the results.

HINDMARSH: Yes, yes. I think that may be it. That may be it, although I never knew exactly the details of it. But stiffness comes about in a lot of other areas, although the term "stiff" certainly has mechanical connotations in the word itself. The typical example I give now to people is chemical kinetics, because we know that some chemical reactions go very rapidly—you know, you put two things together and they explode or the reaction goes very rapidly; and other things react slowly. And when you have a mixture of things going on, the time scale associated with the different reactions in the system varies greatly from slow to fast. In such a system, that would make it stiff, that variation in time scale.

There's been some controversy about exactly how you define "stiff" in mathematical terms. There are some people who want to call a problem stiff if it has a range of time scales, no matter what kind of time scales they are, and I think that's wrong because it isn't very helpful. This definition that's really helpful and useful is that there's a mix of time scales, and the short time scale is associated with a damping process. And you do see that in the solution. If you take a stiff system and plot its solution, then if you make a small perturbation at any one point and look at the solution to the initial value problem from that point on, you have a rapid response back into a smooth solution. It's that rapid damping that we define as stiffness. If the stiffness comes from a small time constant associated with an oscillation, which is something very different, then you've got to deal with that with a very different kind of method. So it's not useful to call that stiff also.

HAIGH: I imagine that this concept of stiffness would only really become important when computers became more widespread and it was possible to attempt to solve these problems automatically.

HINDMARSH: Right.

HAIGH: So before that, that wouldn't have been an idea that existed at all in mathematics. Is that true?

HINDMARSH: That's right. You could deal with these problems in analytic approaches sometimes, if you can deal with them at all. But as soon as people started doing traditional numerical methods on them, they found that the biggest discrete step size in time that they could use was the one associated with the smallest damping time constant in the problem. Even though the solution was very smooth on that time scale.

HAIGH: That's interesting as a mathematical property that was really discovered experimentally.

HINDMARSH: Right.

HAIGH: Have people since then been able to go back and reason more formally about this property?

HINDMARSH: I'm not sure what that would mean.

HAIGH: I'm not sure what it would mean in this specific kind of case, but having discovered it literally as a stiffness in a servomechanism, I got the impression from your comments that they have since been able to produce some more theoretical body of knowledge that would describe exactly what is "stiff" or help to predict whether it will or won't occur.

HINDMARSH: Yes, that's true. In mathematical terms, the characterization is done, nowadays, in terms of the eigenvalues of the Jacobian matrix of the problem, that kind of thing. So it's characterized in mathematical terms that way now, yes.

[Tape 2 of 3, Side A]

HAIGH: So we've talked a little bit about the concept of stiffness and how you became more familiar with it. One of the things you said was that when you first received this software that was called DIFSUB from Bill Gear, that you looked at the listing and you immediately became aware that things would have to be done to it.

HINDMARSH: Yes.

HAIGH: I know this was around about the time that people were first beginning to talk about structured programming as an idea and to become more aware of style as something that applied to computer code. Can you reconstruct how you would've looked at the program then and what the clues would've been that something might be better expressed in another way, or that some of the properties of the code should be reformulated.

HINDMARSH: Well, we weren't computer scientists so much. We were mathematicians looking at this, and we were working with a given computer environment at the lab. So it's not so much that we're thinking about structured programming, but thinking about a number of things: firstly efficiency on the given machines, which called for a certain amount of reorganizing within the code; and secondly, the area of the user interface, which does involve structure and maintainability of the code. So we were I think on our own, evolving our own philosophy about what good software should look like, and Fred Fritsch is the one responsible for that more than any other person. He taught an in-house course, as a matter of fact, on that, called "The Construction of Mathematical Software." It was a course in which he had each of us, a half of dozen or so, talk about a particular project and how it was done and how it could be done better in terms of the organization of software. My GEAR package was one of the things that was talked about in that course.

HAIGH: Was portability a direct concern for you?

HINDMARSH: It was in the sense that we did have a number of different machines and a number of different compilers, some vendor-supplied and some home-grown, and those were different enough that even on the same machine, we wanted to make sure that the code was portable across the compilers. We were working with long-word-length machines in those days. The CDC machines and the Cray machines were 60 and 64-bit machines in single-precision. So our software was single-precision, whereas most of the outside world was using 32-bit machines, it seems. Everybody else was using double-precision arithmetic for scientific computing. So that was one thing that actually got in the way for us. We would import software that was written in double precision and convert it to single. If we wanted to export what we had, we would arrange

it so that it was as easy as possible to generate both double and single-precision versions. But we learned how to do that, and we didn't use anything very fancy for that. But we just learned over the course of the years how to design software so that was as easy a thing to do as possible.

HAIGH: You mentioned issues with the user interface of the code as you received it. What in those days would you have considered to be a good user interface and what were the signs of a bad one?

HINDMARSH: In the early codes that people wrote—and STIFF is an example of this—you had a single routine that took one time step to solve the initial value problem. And you as a user would have to write a loop around the call to that routine, and you would have to be responsible for getting the integration to go from the initial time to the final time in your simulation. There would be a lot of other individual inputs. [Points to a paper] You can see the call list to this routine covers two lines, and there's a lot of things in it.

HAIGH: So the standard code would really just cover the inner loop.

HINDMARSH: Just cover the inner loop, that's right. So one of the first things that we did, and that the GEAR package had, if not initially then soon after, was a driver routine where the user would simply tell us the initial time and then the next time, "Where do you want the answer next?" which could be the final time or could be an intermediate time, whatever. That would contain this loop that would loop over calls to the internal single-step routine. The other thing that we knew from users that they wanted, was a variety of options on how to control errors and how to do other things. There's always tolerance information that the solvers have to get from the users, and that tolerance information is applied within the solver, but we don't know until the user tells us whether he wants to control relative error or absolute error or some kind of combination of the two. So there's that kind of issue, and we want to make that as easy as possible for the user. A lot of times, there are other constraints that the user wants to impose, and so we need options available for things like that.

HAIGH: So if the software sensed that the error was rising above the acceptable levels, would it just give up and issue a warning?

HINDMARSH: Well, in the early versions, that would be what would happen. In the early codes, if the software couldn't meet the tolerance, maybe it would make some attempt to meet it, but it would tend to give up early on without doing too much. Now, the STIFF program Bill Gear wrote actually did have quite a bit of logic within it to try and avoid giving up. Most of that was retained, though we made a lot of changes to the details of that. That was quite a step forward compared to other ODE solvers that people had written. But people wanted things like the ability to solve for the root of a function while you were integrating the ODE initial value problem. So for example, you might have a model that tracked a particle going through a geometrical configuration, and you wanted to know when the particle hit a wall. Well, that would be defined by the root of a function. You needed to have the option of providing a function whose root you wanted while you were integrating the ODEs, so that kind of thing got added to some of the solvers. There are a lot of things like that changed the way we wrote our user interfaces.

HAIGH: Were there any important generally used pieces of software during this period for other kinds of ODEs?

HINDMARSH: We had non-stiff solvers; we had Adams solvers around. We had Runge-Kutta solvers around, and those were used. For example, Bob Gelinas with his kinetics problems, was trying an Adams solver that I had pulled out from somewhere. I don't remember now if it was one I wrote or one I got from somewhere else. That was a nice solver, but it wasn't any good for stiff problems, and we could see that right away.

HAIGH: So were stiff problems substantially harder?

HINDMARSH: Yes. The reason that they were harder is that you had to go with methods that were implicit, and implicitness means that each time you take a time step, you have to solve a nonlinear system of equations whose solution is the new value of the vector of dependent variables. So talking about a system that we denote y'= f (t,y), y is the vector of dependent variables. Each new value of y, as you take a time step, is a solution of a nonlinear system. There were various standard methods for doing that, but it means that you have to deal with the Jacobian matrix, which is the matrix of partial derivatives of f with respect to y, and you have to then solve a sequence of linear systems, to use a Newton iteration, in order to get the solution to this nonlinear problem. What it really comes down to is you have to be able to solve linear systems in which the matrix is very closely related to the Jacobian matrix of the ODE system that you're given. That's what makes the problem expensive to solve, and it makes the software difficult to write because there are so many possibilities for what that matrix can look like, and you have to deal with it. With non-stiff solvers, you don't care what that matrix is like. That really is the single biggest reason why there are so many different solvers for stiff systems, so many possibilities for that matrix.

HAIGH: So you finished up dealing with all kinds of special cases.

HINDMARSH: Yes. Originally, when we got the GEAR package, the treatment of the matrix was done with a purely dense matrix solver. A big n by n matrix comes out, and you give the linear system problem to a standard linear system solver that solves n by n linear systems, Ax = b. The matrix in that system is treated as if every element in the matrix is non-zero. There's no exploitation of structure in a non-zero structure within the matrix. So that was the GEAR package, which grew out of Bill Gear's STIFF routine.

HAIGH: Was GEAR the name that you gave in-house to the package that you produced, based on DIFSUB?

HINDMARSH: Right. One of the things I've got here is the GEAR package manual, actually in Revision 3; so this is the fourth version, dated 1974. One of the things that we put in here for the user's benefit is a list of the revisions that we made to the original, to the DIFSUB routine, just to point out that we weren't just repackaging the original. This list goes on for several pages here; it's got 28 items. But the first item in it is: the driver has been added as an interface between the user and the rest of the package. It oversees the integration of the ODE over the interval between two of the user's output points. And then the list goes on and on, and the things had to do with both changes in the user interface and changes internally to how things are done. The GEAR package was made available internally from about 1971 I think, that is the original one. That made a big hit. But pretty soon as the problems people were solving with it got bigger and bigger, we all realized that we would have to do a lot more to get solutions, and the thing we had to do was to exploit the sparsity structure of the Jacobian.

HAIGH: Was the GEAR package ever disseminated outside the laboratory?

HINDMARSH: Oh yes, it was. We went through a code release process for that, and pretty soon I was getting requests for it, and then we had to wonder how the heck I was going to transmit it to people who were asking for it. The release wasn't too bad. That's a whole subject area in itself. But as long as we could convince a number of people that this was not going to help anybody build weapons or something, they would let it go as a piece of general purpose software.

HAIGH: There was an internal process you had to go through?

HINDMARSH: Yes. We had to fill out some forms, and eventually those were about four pages. It asked questions about whether it was copyrighted or protected by anybody's commercial restrictions of any kind.

HAIGH: Was that similar to what you would have had to go through to present a paper at an open conference, or was it more elaborate?

HINDMARSH: It was more elaborate in the sense that the lab seemed to be more concerned about the usability of the code, because it was code, and also about the copyright restrictions.

HAIGH: Was the thought in the lab at that point that its investment in the intellectual property should be protected in some way or was the concern just about externally produced code that somebody else has copyrighted?

HINDMARSH: There was interest for protecting the lab's intellectual property in a sense. As the years went by, there got to be more of a formalization of that in terms of a technology transfer office that tried to commercialize and license software that was still up to the lab. They did successfully do that with a lot of lab software. I was never in favor of restricting it that way. For any mathematical software, I wanted to see it go out unrestricted, unlicensed, get in the public domain, because I didn't want to hassle with the business of licensing, for one thing, and I wanted to promote the free exchange in both directions of this technology. Our organization in the lab benefited tremendously from the feedback that we got from the number of users that we gave this stuff to, and from other software similar to this that people wrote or sometimes that they gave us in exchange. So I was all for freedom of exchange.

HAIGH: Did you include any kind of copyright statement or license in terms with the software?

HINDMARSH: Not in those early years. At some point later we were required to, but it was a nonrestrictive statement.

HAIGH: With this original GEAR package, were you the only author within Livermore, or were there other collaborators?

HINDMARSH: In the very first edition of this, Bob Gelinas was a co-author. In the revised versions, I was the only one involved so my name is the only one on that. And I was the mathematical point of contact for it, and I was the only one that was making card decks to

answer the requests we were getting, which was no fun. I was sure glad when floppy disks came along because that simplified the dissemination problem.

HAIGH: At this point it would be distributed primarily on punch cards.

HINDMARSH: Punch cards and then tapes. I did send a lot of tapes out too. But we never had a very efficient clean system, from my point of view, of going in and writing a tape that would be readable at any other site, because the lab wasn't used to doing that.

HAIGH: Was the FORTRAN code itself able to run efficiently on a variety of platforms?

HINDMARSH: We made sure of that, and of course that improved over the years. We got a lot of feedback about ways that it wasn't that portable in the beginning, and we improved it every chance that we could. We made a lot of improvements on that.

HAIGH: What was the process for testing and debugging and getting user feedback?

HINDMARSH: This is something that Fred Fritsch was responsible for. We developed a procedure that was quite thorough and quite careful about quality control for the software. We would first of all test individual components of it whenever we could to make sure that each one was doing exactly the job that it was supposed to do. And we would always work up a set of test problems that we thought were representative in some way of what the software was going to see in the real world, at least in basic characteristics if not in size, because we don't want to have huge problems. But we would always make sure the software would run on these test sets. And sometimes there were very large test sets. We had a couple of a dozen problems that we ran with the GEAR package before it went out after all. And we'd always make sure that we would run each test problem with a variety of inputs, and a variety of tolerances, different method options, and make sure everything worked. We tried to make sure that every part of the code was being exercised as you went through the test set.

HAIGH: And that testing was done within the lab?

HINDMARSH: Right, right. Fred would require a test file to be generated that documented all the problems and the test outputs to make sure everything looked good. Those were in the days where we really had time to do that, and we took the time to do that quality control. That unfortunately just degraded over the years as the funding situation changed.

HAIGH: Now I'm looking at a list of publications. It appears that around 1975 or '76, that after the first couple of publications, you began to shift your work so that these software packages would be the main topic of your published research.

HINDMARSH: Yes. The atmospheric chemistry had a lot to do with that. The very first thing that happened on that, and was really quite typical of the others too, was that we put out a variant called GEARB. B is not just the second letter of the alphabet; it stood for banded. If you have an ODE system in which the Jacobian matrix has all of its non-zero elements lumped along the main diagonal, you've got a band matrix. Band matrices come up typically when you solve large systems that come from partial differential equations, or systems of partial differential equations, which of course is really what the lab has more of than ODE systems. GEARB was put together

by taking a banded linear solver, and there were plenty of choices around for banded linear solvers, so that was nothing new, although I remember having to do some rewriting to make things sufficiently flexible to combine it with the GEAR package. So we put the banded linear solver with the GEAR package, and that was GEARB, and it was able to solve systems with banded structure and therefore solve a huge class of problems that you could never solve with the original GEAR package, because it exploited that structure. And that sold like hotcakes. The GEAR package was a big seller, but GEARB was a much bigger seller because all of a sudden people who were not only interested solving PDEs but writing general purpose software for PDEs could incorporate it within their software.

HAIGH: What year was that released?

HINDMARSH: That was first released in 1973. That was the beginning of a sequence of software that had GEAR as its core solver and then variants on the structure of the problem being treated.

HAIGH: As a researcher in a lab rather than an academic department, did you feel pressure to publish things?

HINDMARSH: Not really. We had some. Fred would tell us that we should generate a paper for a journal if we really had some good stuff that we did, and he encouraged us to do that; but it wasn't a pressure situation in terms of not being up for tenure if we didn't get a certain number of things out.

HAIGH: During those good years then, do you feel that the lab was an environment that was more supportive of this kind of work on software than an academic department was?

HINDMARSH: Absolutely. When we looked at what was coming out of our academic places, on the whole they were good at the mathematics of methods maybe, but generally they were very poor at implementing the methods in an efficient way. And they never worked on large problems either. We always had large problems that we needed to solve. The methods produced in academia were inappropriate for large problems, and they didn't even realize that. They would test it on small problems and it looked good there. When it came to large problems, there would be something that would make it terribly inefficient.

HAIGH: Was there anything characteristic in the nature of problems found in specific areas? You have mentioned the stratospheric research, so would particular groups within the lab, or particular kinds of science, be more likely to produce these stiff equations.

HINDMARSH: Very definitely. So chemical kinetics is one huge area that generates stiff problems. Then in the atmospheric modeling, we went from doing models that had one spatial variable involved and a set of chemical reactants at each point in space, to going to two dimensional models and much later 3D. Some 2D problems could be handled with GEARB because the problem was still banded. As soon as we went to two space dimensions, the problems got really big, and the structure of those matrices that came up were such that the banded treatment was not efficient. At that point, we went to a block-iterative treatment. I think we were probably the first to have a solver that used iterative methods within an ODE solver. We had a lot of firsts. That was one of them.

HAIGH: I wonder as the ‘70s developed, I see, for example, that one of the things on your resume is a 1976 panel discussion on quality software for ODEs. [Panel Discussion on Quality Software for ODE's (with G. D. Byrne, C. W. Gear, T. E. Hull, F. T. Krogh, and L. F. Shampine), in Numerical Methods for Differential Systems, L. Lapidus and W. E. Schiesser, eds., Academic Press, New York, 1976, pp. 267-285]. So at what point did you begin to feel yourself as part of a community of mathematical software people in general, and part of a specific group of people focusing on ODEs?

HINDMARSH: When our packages started getting exported, other software writers were also looking at them so they gave us feedback and they were comparing with their own stuff. It was a very healthy competition going on amongst various people that were doing the same kinds of things—we certainly weren’t the only ones writing stiff solvers. The first time that this actually led to a meeting of any significance that I was involved with was in 1974, and people at Argonne decided to call a workshop on ODE solvers. I’ve got a folder on several of those workshops. That was the first of a sequence of them. At that workshop we had Bill Gear and George Byrne, and Fred Fritch was there; a student of Gear’s was there; Marilyn Gordon and Larry Shampine from Sandia; Tom Hull from Ontario, Toronto; Fred Krogh from JPL; Joan Walsh from Manchester; David Sayers from Oxford; Brian Ford from NAG was there for part of the time. Those are all big names in the business, hosted by Argonne at this meeting. It was a very informal thing, but we sat around and we talked about the state of our own work in the field and what seemed to be needed in the future, and we exchanged a lot of good ideas. It was great. That’s the first point where I really felt that now I was part of an international community involved in ODE software in particular.

HAIGH: You also mentioned that there was a kind of a competition between the different groups producing software in this area. So how would you define the balance between collaboration and competition?

HINDMARSH: It was tricky. That’s another reason that you would see a large number of names of solvers out there. There was really a sense of provincialism, I guess, in the sense of people saying, “Well, I can write a solver that’s better than the other guy’s,” even though the method is not new. So there was a lot of that. And it was tricky to balance especially within the Department of Energy labs. There was particularly strong competition between the Sandia people and the Livermore people. That was very rocky at times. Shampine and Watts and others at Sandia had a lot of experience writing Runge-Kutta solvers and then Adams solvers, though they had not gotten into stiff solvers. But they had a lot of things to say critically of the stiff solvers that were coming out of Livermore. It was a good thing for the most part. It was good to have that feedback. But it was also negative sometimes too, so we didn’t appreciate that.

There was a lot of discussion in this sequence of workshops about trying to standardize too. Even if we were all doing different things as far as what kind of solvers we were doing, there was a definite sense of a need for standardization, that a non-stiff solver from Sandia and a stiff solver from Livermore really ought to look similar to each other if we’re going to present the whole collection to a user community. It would be nice to make them look similar. So we talked about standardizing calling sequences. Should we call the independent variable T for time, or should we call it X? Should the dependent variable vector should be called Y or U? That sounds kind of silly, but that’s the kind of thing that we talked about, as well as deeper things like what methods to include and how methods should be made available, and the user interface.

The workshops that happened over the next several years got into all these things in great depth. The one at Argonne was just the beginning. We agreed at that meeting to have another one in a couple years, and we had one in San Antonio in 1976.

HAIGH: Was there a formal name given to this group?

HINDMARSH: The one in San Antonio was called the ODE Circus, and this circus name was kind of appropriate in some ways because we were such a diverse bunch. We were all touting our own thing for a lot of it, but it was good because we were all benefiting from what other people were doing. I've got some notes from that meeting, too. We had to formalize it. But the group broadened, too. Instead of a dozen or so people that we had at Argonne, I don't know what the number is, but the scope certainly broadened. We had Roland Bulirsch from Germany; we had Alan Curtis from Harwell; we had Wayne Enright and Tom Hull from Toronto; we had Ralph Klopfenstein from RCA, New Jersey; Werner Liniger from IBM, Watson; and Ted Bickart from Syracuse. So the range of people participating was getting broader and broader every time. I remember that we had some kind of a house out in the country near San Antonio that we had all to ourselves. It was beautiful. We had all the food and drinks we could ever want. We just had a great time. But we had a lot of late-night discussions that got pretty heated over ODE issues where people felt strongly and differently with each other in a lot of ways.

HAIGH: How much of the disagreement was about these relatively trivial things that you've mentioned like what to call the variables and how much do you think reflected deeper mathematical issues?

HINDMARSH: There were both kinds of disagreements. People seemed to, for some reason, have a big vested interest in keeping their call sequences and their variable names as they were, which I never understood too much. And people differed also on matters like how to represent the error control, how to offer different options in error control. I remember one really long but still friendly argument that Alan Curtis had with the rest of us. He was on the short end of the stick because he believed that error control should always purely relative. The rest of us kept saying, "What do you do when a variable goes through zero? You can't control relative error on zero." He'd say, "Well, I do it all the time, because zero is a tiny number and you control relative error on that," or something like that. I don't remember exactly how he argued it [chuckles]. There were things like that.

HAIGH: Did these meetings lead to tangible results in terms of standardization?

HINDMARSH: Somewhat, yes. That's a rocky story too. This gets back now to the laboratory situation and the way we were funded. I had part of my funding from what we called an external source. It was the DOE Office of Scientific Computing. It was the Applied Mathematical Sciences Program. That was funding that we received directly, so it was not part of the big pot of money that the laboratory got from DOE to run the laboratory; it was separate money that we wrote proposals for. Part of the money that I got from the Applied Math Sciences Program was earmarked to develop an ODEPACK, without saying what that was to be; it was the idea to develop the concept. So I had some meetings that grew out of these other workshops with people to start trying to standardize and discuss what a systematized collection of ODE solvers would look like in the model of LINPACK and EISPACK. At that point, we needed to keep the scope much smaller than the workshop, so we were just talking about DOE laboratory people then. We

had meetings with people from Los Alamos and Sandia, both Sandia-Albuquerque and Sandia-Livermore. And we went through these issues all over again with that smaller group, and we found that even with that smaller group, there was a great deal of jealousy and not wanting to change what was already there, and we had trouble agreeing.

The discussions went on for several years, and we thought we had an agreement at one point. There was a report that George Byrne and I put out in '76 called "A Proposed ODEPACK Calling Sequence," and this was supposed to be something that was the starting point for a standard user interface for an ODEPACK. We thought we had agreement across this group of people about that, so we started doing modifications of our existing software to implement that. It turned out that the Sandia people were mounting a rebellion and they decided to abandon that and pull out of the agreement. They just went their own way, and there was nothing we could do about them. They developed something called DEPAC. That is a good program, there's nothing wrong with it; it's just that they had a different philosophy of user interface style and documentation that was different from what we were developing, and it was very similar to what they already had and they didn't want to change. So they went their way.

HAIGH: Interesting, because I was looking at the Shampine-Watts chapter in Wayne Cowell's book. [W. Cowell, "Sources and Development of Mathematical Software." New York: Prentice-Hall, 1984]. I did notice that there was less overlap than I expected between the things discussed in that chapter and the things discussed in your own survey article. That is to some extent reflecting a division between the two labs.

HINDMARSH: Anything you see by Shampine will have an absolutely minimum mention of anything from Livermore, if anything. He will go out of his way not to mention anything from Livermore.

HAIGH: I don't think he did in here.

HINDMARSH: Although, in fact, in the DEPAC collection, the stiff solver is a modified version of the GEAR package. They probably don't mention that.

HAIGH: You can check the references.

HINDMARSH: I am mentioned. LSODE and I are in the reference list. Okay, it does say that DEBDF is an adapted version of LSODE, but that's it. What came out of that for me was that I didn't particularly care for the philosophy that Sandia had adopted on its own. They weren't being flexible about it. We had something that the rest of us tended to agree on, so the advice I was getting from people, like the Argonne people, was to just go ahead and put out ODEPACK based on the Livermore solvers that we had, and I did that. So the thing that is out there that is called ODEPACK is a collection that begins with LSODE, and LSODE is rewrite of the GEAR and GEARB packages combined, and then variants of that for other structures for implicit systems and systems with different Jacobian structures. That's what happened there. There was an intermediate phase with something called ELSODE, pronounced the same, but it didn't last very long, and that was Energy Laboratory Solver for ODEs, so that was the acronym. That was meant to be something that was a version that we would agree upon among the DOE laboratories. But as I say, the Sandia people pulled out of that.

HAIGH: I'll talk to you in some more detail about those other packages tomorrow. I just have a couple more questions on the cultural context and the professional community.

[Tape 2 of 3, Side B]

HAIGH: This relative lack of success in standardizing the software and achieving collaboration between the different labs, compared with LINPACK or the later accomplishments with LAPACK, to what extent do you think that's just because the mathematics is harder and the answers are less obvious in the ODE area and to what extent do you think it's just the result of the personalities of the people involved?

HINDMARSH: That's hard to say. I think both contribute. I think a lot of it is the momentum that the different groups of people had going with the software that already existed. In the case of LSODE, which itself became ODEPACK, LSODE was fairly well along by the time the Sandia people pulled out, and it didn't make any sense to rewrite it in the model of the Sandia DEPAC or vice-versa. At the same time, I think that the fact that the Sandia people had very little experience with writing stiff solvers meant that they didn't appreciate some of the issues in designing the other members of the ODEPACK family that we had. There were mathematical reasons, too.

HAIGH: I saw that one of your papers was published in the SIGNUM Newsletter. I was wondering what your involvement was with that group and what your feelings were about its function.

HINDMARSH: SIGNUM was a good informal group, and I went to their meetings a number of times. It was a good outlet for material that didn't have to meet the high standards of journal publications. You could submit things that were quick and dirty and get it out there, so it was nice that way. The meetings that we had --- I don't remember a lot of them, but they were good exchanges, and people concentrated on numerical analysis and software issues at those meetings that were pretty relevant. There were other meetings like that that actually were more significant for me. The mathematical software meeting that John Rice hosted at Purdue in 1975 was a good example. That had a focus on mathematical software that was great. That was at the time that George Byrne and I were coming out with a new solver, which was presented at that meeting. Unfortunately I came down with pneumonia during that and I came home early. I missed out on almost the whole meeting.

HAIGH: Rice's series of meetings in the 70s is something that many of the interviews have spoken about at this point, an important moment in the community coming together.

HINDMARSH: We were. We had some involvement with those. A couple of the other meetings that happened: one was at Park City, Utah in 1982. That was particularly devoted to stiff computation. That was a very good meeting, and it was good also in the aspect of an outside person there hosting it and being willing to generate a document. I don't know if you've seen this, but this is the book that is essentially the proceedings of that conference, and it's a thick book.

HAIGH: I should say for the tape that this is Stiff Computation, and it is by Richard C. Aiken, published by Oxford University Press 1985.

HINDMARSH: That's got a wealth of material in it, both formal and informal, because there was a panel session that we had. The transcript to that panel session is in there too.

HAIGH: I see it; it's called 'The Future.' Do you think that was a particularly important conference?

HINDMARSH: It was. Another one that comes to mind is one that happened in Söderköping, Sweden in '83, a year after Park City. That was an IFIP meeting. [Working Conference "PDE Software: Modules, Interfaces and Systems", the proceedings, edited by Björn Engquist and Tom Smedsaas, were published by North-Holland in 1984].

HAIGH: Was that with the working group 2.5?

HINDMARSH: I believe so.

HAIGH: I know that group played an important part in several areas of mathematical software. Would you say they've been active in the stiff ODE area?

HINDMARSH: I'm really not sure about the workings of that working group. I just know that they had that meeting. I know that one of Gear's early contributions came out of an early IFIP working group meeting. I don't think I ever went to any other meetings of that particular organization, so I didn't connect that much with them as an organization. I'm sure the people involved were at the other meetings that I went to though. IMACS is another organization that was heavily into numerical methods and software, but that's a very big organization. They have these world congresses that cover a huge variety of subjects including applications. But I got involved with both organizing sessions and contributing papers at IMACS meetings over the years. Then IMACS also had a sequence of meetings devoted to solving PDEs exclusively. Bill Schiesser at Lehigh University was the one in charge of that. He's not doing it any more. But he is a good friend of mine, and he ran these PDE meetings, which included a lot of people and had wide variety of talks on both methods and applications, and I would typically go and talk about how the ODE solvers that I worked with could be used to solve PDEs. They were being used, not just by me.

HAIGH: That takes us well into the 1980s on the professional and community front then, so I think this would be a sensible place to stop for the first session. Tomorrow we'll return to that topic, then talk in more detail about the evolution of these various packages through the '70s and '80s.

[End of Session 1].

HAIGH: This is the start of session two, on the morning of January 6, 2005 at Dr. Hindmarsh's house in Livermore, California.

Before we pick up where we left off last time, I understand that you have a few additional points you want to make over the material already covered.

HINDMARSH: I'm afraid my mind went blank on a couple of spots yesterday I want to catch up on it. You asked me what other software out of Livermore was popular other than ODE

solvers, and there are a couple that I should have mentioned. There were some PDE software packages that were written by Niel Madsen and Rich Sincovec, and one of those was called PDECOL and that appeared in the TOMS journal. It was quite popular and still is, I think.

The other thing I should've mentioned is some data interpolation software that was written by Fred Fritsch and Ralph Carlson. They had some things for both 1D and 2D data interpolation, especially with monotonic interpolant constraints. Those were a one-of-a-kind things that were popular.

There's one other thing that I wanted to relate that was kind of a small story related to the GEAR package, and it's an illustration of how effective that kind of package was compared to the image that people had of methodology for solving stiff systems at the time. A colleague of mine at the lab named Mike (and I'll just use first names for this) was hired in about 1975 or so. I'm not sure exactly. He was hired by a computational physicist named Julius. Julius had some kinetics equations he wanted Mike to work on. Mike also had a pet project of his own that he wanted to work on. Julius came in and gave Mike this set of equations and said, "See what you can do with this and I'll get back to you in about a month." Julius was very busy, so he didn't supervise Mike very closely. Mike looked at the problem, and he'd been there a few days. I don't know how long, but he had heard about the GEAR package and he went and got the GEAR package out and plugged this problem into it and got his answers by the next day. So for the other 29 days of the month until Julius came back he got to work on his pet project. He was very pleased about that. I heard this story from Mike much later. I didn't know about it at the time. It was just an interesting indication of how Julius, at least, and others also felt about what was available, what the technology could do, and the GEAR package made quite a leap forward in that sense.

HAIGH: As you developed further packages during the '70s, to what extent was the motivation to do that coming from internal users who were pleased with GEAR but had problems that it didn't work with or found some aspects of its functioning confusing?

HINDMARSH: Yes, in those early years the motivation was almost entirely from internal users and the problems they needed to solve and sometimes couldn't solve with the GEAR package, and there's a very good case and point there in the atmospheric modeling business. We had problems in which there was diurnal chemistry. What that means is that a chemical kinetic system included photochemical reactions, such that in the model when the sun comes out a reaction turns on and at night it turns off. So that turning on and off is quite rapid. It's not discontinuous, but it almost looks discontinuous when you stand back and look at it. And those problems often made the GEAR package crash. Not because they were big; just because of that demand on changing the step size in order to track that steep rise and steep fall. Some of the solution components follow the same kind of deep rise and fall as the reaction rate. So we looked at that problem for a long time. We tried to do tuning on the GEAR package to make it work well. We were only partially successful with that and we realized that we needed a different kind of method. I get a little technical here, but basically the GEAR package and its variants use a method called fixed-step and interpolation as the underlying method, so those packages use fixed-step formulas. They assume to begin with that you are taking fixed time steps, and then they achieve variations in the time step by an interpolatory process. We realized that what we needed was a variable-step method, a method in which the basic formula itself provided for arbitrarily different step sizes from step to step.

HAIGH: And have methods of that kind been implemented previously or is this a new idea?

HINDMARSH: There had been some work on that. In particular people at IBM had done this in their own particular way, they had put out some papers on it, but their software was not available. And in addition to that, even if we could get their software we would have wanted to do it in a different way, so as to have a package that looked like the GEAR package, because we already had developed a certain philosophy of design here. This was something we had on our agenda and that was about 1973, and at that time we had an application for summer hire from a fellow named George Byrne. He was a long time faculty member at the University of Pittsburgh, and I didn't know him before this, but the application we got looked good, and Fred liked it, so we said, "Let's hire him." He had some numerical ODE experience, so we said, "Let's do this variable-step BDF code together," George and I.

So he came for the summer of 1973. We worked through the theory of the basic methods so far as we had it from the literature. And then we had to do a considerable amount of further theoretical development in order to formulate this method in terms of what we call the storage of history information that we were going to use. Without getting too technical, it's the same history method that the GEAR package uses. How you implement a method depends very heavily on how you are going to store the history information, so that was a big issue. Also how you do the control of errors, how you estimate the error that's committed every step and then decide how to change the step size in order to satisfy tolerance information, and how to choose a new order of a method based on the error control. All of that needed to be worked out, and hadn't been done, so this was definitely our original contribution to the field.

By the end of the summer, we pretty much had it. It took another summer or two to get some of the kinks worked out and get the software to a point where we were happy with it, but we ended up with a code called EPISODE. It was a name we worked on for quite awhile. It stood for Experimental Package for the Integration of Systems of ODEs. By 1974, I think March of that year, we were ready to present this at the Purdue software conference. Fortunately George was the one that was going to do the presentation because I got sick at the conference and I came home and found out I had pneumonia. Didn't get to even see the presentation.

So we had something new that we were quite happy with. It solved the diurnal problems quite well, and so we put this code out for public consumption and explained to people, which was a little bit tricky, that this looked very much like the GEAR package, indeed the user interface was almost identical, yet it was appropriate for a class of problems that the GEAR package was not appropriate for. It needed to be a separate package because of the fact that too much of the basic algorithm in the code was very different from the GEAR package.

HAIGH: I saw a reference, I think in your 1986 survey article that the EPISODE package had this ability to change step size at each integration step, but on the other hand, had "a worse user interface and less modern linear algebra."

HINDMARSH: Well, no, that's only by comparison with LSODE. That's because we're out of order here. The GEAR package and the EPISODE package are of the same generation of software in terms of the internal design and the user interface, so they are very much alike.

HAIGH: So it had the same user interface and linear algebra as GEAR?

HINDMARSH: As the old GEAR package. At the same time EPISODE was being worked on, we were also working on the redesign of user interface and the redesign, internally with the ODEPACK concept, and LSODE was the result of that.

HAIGH: Let's stick with EPISODE for a second. Was there anything different in the development or testing methodology from the earlier GEAR package?

HINDMARSH: Not in the basic methodology. The only thing that changes is that we wanted to be sure to get in this harder class of problems, including diurnal chemistry problems. But the methodology for development and testing was pretty much the same.

HAIGH: And was it again true that the package was developed with the labs internal funds, that you didn't need special funds?

HINDMARSH: Pretty much, although by this time, by 1974 we had an external audience to think about too, because the GEAR package had been out and was somewhat known outside the lab. We would have these meetings at which we talked about our software and other people were interested in getting it, beginning just with ODE developers. So we were thinking about outside users too.

HAIGH: Did that lead you to use any external test sites, or was the testing still all done internally?

HINDMARSH: We let other people test it on their own; there was no problem with that. We didn't have any full collaboration on that, though. For example, the Toronto group, Tom Hull, did their series of tests and they included EPISODE in their tests, although we were never very happy with the thoroughness of that testing, the validity of a lot of it. They had what they thought were stiff problems in their stiff test set that were not stiff at all, so there were some issues with that. But we didn't connect so much with that. We didn't collaborate with the Toronto people in that.

HAIGH: Did the dissemination system remain the same as for the earlier package, which as I understand it, was basically you running off tapes and punch cards personally?

HINDMARSH: Yes, I'm afraid it was. By this time I was beginning to get phone calls and letters in the mail --- we didn't have email yet --- saying initially, "Send us your literature about these codes." Pretty soon they'd say, "Well, we'd like to have the code if you can send it to us." I was generating a bunch of card decks and then I was writing magnetic tapes and sending them to people. And it got to be a bit of a heavy burden, but I enjoyed it—the fact that these codes were popular was kind of fun to see.

HAIGH: Do you have a sense of what the total number of copies you sent out would have amounted to over the years?

HINDMARSH: I do now, but the numbers that I can give you now reflect the later generations of software too. I brought some things that give some idea of that. First of all, I kept the letters and notes of phone calls that I got every time I got any of these requests, and so I kept them in folders like this and I built up a sequence of folders that filled a three-foot-wide file drawer at one time.

Almost all of that I've had to throw away in my last office move, but just for kicks I kept a couple of the old ones. Here are requests from 1971 to '74, and it's about half an inch thick and it's got all kinds of letters and postcards and stuff. Here's one from Czechoslovakia, one from Cornell. They're from all over the world. Oak Ridge, Lockheed, General Electric. There's Czechoslovakia again. There was one that was kind of cute that I got from the Greek Atomic Energy Commission and I wasn't quite sure how I was supposed to handle that, but I did send them something. I think we weren't allowed to send code to some parts of the world, of course. I didn't count numbers in these earlier years in terms of how many people I had sent to, but there were lots. I was filling a folder like this about once every quarter by the time things got really going in later years.

HAIGH: So you think the number would have been in the hundreds?

HINDMARSH: Easily, probably a 100 per year, yes, easily.

HAIGH: One interesting question with respect to the parallels between this and more formal and recent open software projects would be the extent to which any of the users might become collaborators or might submit their own suggestions or improvements of the code, and my impression is that that didn't happen as much in this era.

HINDMARSH: It didn't happen very often. It did happen a couple of times. Back with the LSODE serious of codes, we had a user at Argonne who was using LSODE but he had a system of equations that was not in explicit form, y' = f; it was in an implicit form where there was a matrix multiplying the y' vector. He struggled for awhile to use the GEAR package or one of the variants on it we had at the time and realized that we really needed something new for that. So we said, "Let's do a version of the GEAR package that solves the linearly implicit form," and I generated a version of that for him. That was quite successful, and resulted in a variant called GEARIB and the LSODE variant corresponding to that, LSODI.

In another instance, there was an actual collaboration with somebody that I never met, a fellow down at China Lake Naval Weapons Center, Charles Kenney; he's mentioned in the ODEPACK summary description. He decided to construct a variant in which the package included a block-tridagonal solver instead of a banded solver or dense solver. He put that in the LSODI package. He generated that and he sent it to me, and I did a little more cleanup work on it to make it conform to the rest of the family, and that was added to the collection. That was a nice collaboration.

HAIGH: That's very interesting. Were there any cases where users discovered any bugs or limitations in the code after it had been released?

HINDMARSH: Oh, yes. There were cases of users having the code get stuck on their problem and if they investigated deeply enough, they would find out that there was actually a bug once in a while. In a lot of cases it was just their misuse of the code. Sometimes a bug would get found. So that kind of feedback was very important; got a lot of that. We often worked on cases where we never really could get to the bottom of what was going wrong with user code because the user's code was too big for us to get into, but we weren't invested in getting the other fellow's code to run. So sometimes we just had to leave things alone. One can carry things too far trying to get all users happy with a given code.

This reminds me of a little meta-theorem that Bill Gear gave one time that I like to quote. The meta-theorum is the following: For any given ODE solver that you like, there's always a problem for which it gives the wrong answer and the answer can be as wrong as you like; the error in it can be as large as you like. You can never make a code that's perfect for all problems. And the proof of the theorem is very simple. You take a problem that the solver works for very well, and you watch what it does. When it solves that problem it samples the right-hand side function at only a finite number of points, a discrete set of points between the beginning time and the final time. It can't know the function completely because it's a continuous function. So if you take that function and then you alter it to a different function that has a large spike in it, in between two of the points that it sampled in the first pass, you've got a new function, and the code, when it solves the new problem, will get the same answer because it's going to sample that function at the same set of points, but the answer will be completely wrong because you could make that spike as large as you like and the answer will be completely different. You can always fool a code with a problem that's pathological enough.

With that point of view, we try to draw a line in working on robustness of codes. We try to test the codes with a variety of problems and parameters so we are pretty sure things are working right.

HAIGH: So that would be not so much a bug as a limitation in the mathematical method itself?

HINDMARSH: Sure. Yes.

HAIGH: Did any of the Livermore code find its way into standard packages and libraries such as the NAG or IMSL?

HINDMARSH: Yes, very much. The IMSL people took up the GEAR package and they realized that they needed to have a stiff solver in their library. They took the GEAR package and they did a modest rewriting of the front end of it to fit their scheme. They had something called DGEAR in the IMSL library. The NAG folks did something very similar. They had the GEAR package and they actually generated four or five different solvers within the library because they split it out into different user requirements. But their stiff solvers, at least initially, were based on the GEAR package from Livermore.

HAIGH: Would you say that that gave adequate coverage?

HINDMARSH: Within those libraries, oh yes. Those were for people who were using those libraries exclusively, yes, those were great additions. I was very pleased to see that happen. Of course, within Livermore where we had those commercial libraries and we had our own stuff that we were supporting directly, I discouraged users from using the IMSL or NAG stiff solvers. I said, "I'd really rather provide support for the original code that I wrote, and users will always get the latest version," whereas what was in those libraries quickly became obsolete as we rewrote GEAR into LSODE and so on.

HAIGH: I see in the book *Stiff* Computation, edited by R. C. Aiken (Oxford University Press, 1985), that at the beginning of chapter four, "Stiff Packages", Aiken contributes a small section where he says, on page 149 that the vast majority of stiff users at the beginning of the 1980s used

early versions of GEAR, commonly available through the user's computer centers. Do you think that would be true at the beginning of the 80s?

HINDMARSH: Yes, yes. By then a lot of sites around the country and around the world, in fact, had gotten a hold of a copy of the GEAR package one way or another. It had been released with unlimited distribution from the lab, and so people not only got it from me but then they gave it to their friends and that was fine with me too. I did ask people to give me names and addresses of people that got it second-hand, but people didn't always do that, and that didn't bother me too much. I knew it was getting pretty widespread. Another thing that happened is that it was made available through the Argonne Code Center; that was an outlet set up by the Department of Energy for public dissemination. It wasn't really a very practical outlet because, at least after a certain point, they decided to charge fairly high fees for code requests. But there were quite a few requests for the GEAR package and later packages from there and every so often I would get a log from them of who had gotten it from them.

Jumping ahead just a little bit, when we got into LSODE and its variants -- ODEPACK, I started keeping my own log. It went from 1979, which was the original release of LSODE, to 1992, so this covers quite a few years, a thirteen year period. It has 419 people in the US and 145 people in foreign countries with names and addresses and what I sent them. In most cases they were getting the whole ODEPACK as far as it went, and I made a note of which precision they were interested, and so on. So that's one thing I kept just for historical purposes.

In about 1985 some people associated with Netlib contacted me and said they didn't have much in the way of ODE solvers and they'd like to have something. Netlib is a very widely used collection of software that's very popular. In fact Jack Dongarra at Argonne asked me to submit the Livermore ODE solvers to Netlib, so I did that, and they are still there. ODEPACK is there among other things of the Livermore collection. And they have a hit count. They have ODEPACK stored in its own subdirectory within Netlib, so the hit count on that tells me something about the popularity of it. Between the single and double precision versions of ODEPACK since 1985, as of Monday of this week, there were 448,374 hits—so that's another indication, close to half a million. That's the biggest number I've got as far as user count. Of course, not all of those are real users. Some people are getting things just to take a look and see what's there, but still, that's a large number.

HAIGH: Now, by the 1980s, had there been any kind of serious push within the lab to make it harder to give intellectual property away?

HINDMARSH: I would say that it did not get any harder for me. I knew the system, I knew how to write the code release forms to get this stuff out, although the procedure evolved and changed through the years. I didn't have any trouble with that aspect until the '90s. So that's a different story. Do you want me to go into that now?

HAIGH: We'll lump that in with the rest of the '90s. So through the '80s that wasn't a serious change then.

HINDMARSH: No.

HAIGH: Did you begin to include any kind of copyright statement in the code itself?

HINDMARSH: No I didn't. I never did that. These were just considered to be in the public domain as far as I was concerned. I actually was told that by one of the lab's legal people, that, because of the fact that it was out there, had been already distributed, there was no way that the lab or the University of California, which operates the lab, could claim any rights to it anymore. It was essentially considered Public Domain.

HAIGH: Would you have had any objection if somebody had attempted to make money by distributing or to roll it into something they charged a premium for?

HINDMARSH: I did have qualms about that when it happened, but not to any great extent. I was glad to see the package get used that way and get used on a possibly wider basis as the result of something like that. And presumably if somebody was adding something around the GEAR package or LSODE that made it worth charging money for, then that's fine. If they were charging a lot of money for just a little bit of a wrapping, then I figured that people faced with the choice could just get the package from me directly and maybe write their own wrapping and not have to pay anything. So I figured that problem would kind of take care of itself.

HAIGH: Did your collaboration with Byrne continue beyond the EPISODE package itself?

HINDMARSH: Yes, in fact the EPISODE package was the beginning of another family like the GEAR family. We did an EPISODEB with a band solver in it, just like GEARB. There was an EPISODEIB that had the implicit equation type treated, and then in the late '80s, starting 1988, he and I collaborated on the rewrite of the EPISODE family in the same way that the old GEAR package was rewritten to make LSODE. So now we are going from the first generation to the second generation, or maybe second to third, depending on whether you count the early stuff like DIFSUB. So George and I would definitely coauthor that next code, which was called VODE, and Peter Brown was actually coauthor on that because he actually helped us with the development of the basic methods there as we made a slight switch from fully variable coefficient methods to variable coefficient methods based on fixed-leading-coefficient formulas, and that's a technical thing which I don't think I'll get into. But it was a variation on the original EPISODE formulas.

HAIGH: And what time period would those variants have appeared during?

HINDMARSH: The EPISODE variants appeared during the late '70s and early '80s, I believe. We only got to three. It was just EPISODE, EPISODEB, and EPISODEIB; and then the VODE package that we wrote which was intended to supersede EPISODE, that was written in 1988. Both LSODE variants and VODE variants got developed over the '80s, a little bit into the '90s.

HAIGH: Now, when you call these packages, does that imply that there was more than one important routine in each of them, or are you using "package" and "code" interchangeably?

HINDMARSH: Well, we do use them interchangeably somewhat, but the name package applies because each solver consists of a fairly large number of routines, and some of those routines are shared among the different solvers. So for example the dense and banded linear system solvers that are in LSODE are exactly the same ones that are in VODE. And across a given family, such as LSODE and it's variants, there are some routines for certain common tasks that are shared

across the family. For example, taking the norm of a vector, that's the same thing in all of those packages within the ODEPACK family.

HAIGH: And would something like EPISODE itself have gone through multiple releases with different version numbers, or did it stay basically the same after it was released?

HINDMARSH: EPISODE didn't evolve that much further. There were some modest revisions and some bug fixes and things like that, but it stayed fairly stable. By the time EPISODE was written, we'd had a fair amount of experience with the GEAR package, so we didn't commit the same errors over again and it was fairly stable.

HAIGH: Did you have a formal version numbering system?

HINDMARSH: No, no we didn't.

[Tape 3 of 3, Side A]

HINDMARSH: We used dates to identify the different versions and every time we made a revision of any kind we'd update that date. When I'd get a call or an email or something from a user that said he was having trouble with one of our solvers, the first question I'd ask usually would be, "What's the date line on your code," so I knew whether he was using an obsolete version or not.

HAIGH: I think we should maybe backtrack slightly now and talk in some detail about the LSODE package which you've referred to a number of times in passing.

HINDMARSH: LSODE, as I think I said, grew out of these discussions that we were having, trying to standardize the arguments and also restructure things to enable a more modern development within the code by comparison to the old GEAR package. LSODE benefited from all those discussions and all the feedback that I got from users, and there was very widespread input to that. By 1979, we had LSODE pretty much ready to go and we released it and it became very popular. Of course, we knew right away that we needed to do more and do variants of it to cover other classes of problems. We did reduce the proliferation a little bit by combining both the dense and the band matrix treatments of the Jacobian into a single code, whereas we had GEAR and GEARB before to cover both of those cases. But the solution of linearly implicit ODE systems needed to have a different code, and that was LSODI. Actually, I need to mention Jeff Painter in that respect. He was a collaborator with me on that in our group.

So LSODE and LSODI was the pair of routines that we had to offer as the latest and greatest in the 1980-81 period of time. Pretty soon we were hearing from people who wanted something a little more sophisticated, first of all, in terms of switching between stiff and non-stiff methods. Linda Petzold, who had been hired at Sandia, was interested in that idea, and she pursued it herself there. She actually took LSODE and we wrote certain parts of it to produce a code that we called LSODA that did the automatic switching between the stiff and the non-stiff methods.

I guess I forgot to mention that LSODE was not just a stiff solver, because we knew that people would want to have a non-stiff solver available much of the time as well as the stiff solver. So we actually put the Adams methods, which are non-stiff, into the same solver with the BDF

methods. There's very little additional code involved in doing that combination. So switching between a non-stiff and a stiff method for LSODE is a simple matter of an input argument. But what Linda did with LSODA was to make that switch automatic and dynamic within the code.

HAIGH: Does the "A" stand for "automatic" then?

HINDMARSH: Yes. And that works fairly well. It's a little more expensive because you have to do monitoring of certain things in order to get enough data to make the decision that the non-stiff method currently being used needs to changed to the stiff method or vice versa. That results in an educational problem, too. And you tell people if they know the problem is non-stiff or they know it's stiff, use LSODE; but if they're not sure, or if the problem varies—this can happen within a problem, it varies between the two—then use LSODA; it hopefully will do the switching automatically.

And then another issue came along: People wanted to have root-finding in conjunction with solving ODEs. That means that while you're solving for y as a function of t, as a solution of the ODE, you have some function g(y) whose root you want to find as you go along. This might be from any number of things like just finding when a certain variable reaches a certain point or a certain value. That's an easy thing to add in. Some people at Sandia had already done something very similar with their codes, their non-stiff codes. So Linda Petzold and I together put that idea into the LSODA package. The result of that was LSODAR. So it was LSODE with automatic method switching and root-finding.

So now we're up to LSODI. Okay, LSODI I've talked about. LSOIBT was the variant that I mentioned that had the block-tridiagonal solver put in in place of the banded system solver. Charles Kenney is the fellow who did that with me, although I never met the man.

HAIGH: So the core difference between the basic LSODE and the earlier GEAR packages would be these ideas that came out of the ODEPACK discussions about improvements in calling sequences in the interfaces, plus the ability to switch between stiff and non-stiff methods within the same package.

HINDMARSH: Yes, for those two variants particularly, yes. There are two other variants I forgot to mention in that sequence. One was the sparse Jacobian variant, which actually goes back to a GEAR variant that existed earlier. I had a very nice collaboration with Andy Sherman, who was at Yale at the time. He was the co-author of the Yale Sparse Matrix Package. That was a very nice package for solving sparse linear systems, so we decided it would really be nice to have an ODE solver variant that would treat the case where the structure of the Jacobian matrix was sparse but not necessarily banded or block-tridiagonal so it didn't necessarily fit into the scheme of the other solvers. So there's this general sparse treatment of the Jacobian in, first of all, a variant of the GEAR package called GEARS. And then after LSODE was done, we did the same thing with LSODE and we have LSODES. There's also a version for the treatment of implicit ODEs with sparse matrix treatment, LSODIS. That wasn't done at the time of that paper there, but it's something I came back to later, to fill out the collection.

HAIGH: So when would that have appeared?

HINDMARSH: I didn't get that done until the early '90s. I'm not sure of the date.

HAIGH: Given that the broader collaboration originally envisioned for the ODEPACK project didn't occur, if someone used ODEPACK to refer to an actual existing collection of software, would it just refer to this collection of LSODE and its variants?

HINDMARSH: Yes. Yes, we didn't feel that we had the manpower or the mandate, really, to build a larger systematized collection which included rewrites of everything that we really had envisioned earlier. We would have continued if we'd had the support, but in the late 1980s the support was declining, too, so it just didn't happen.

Now, there's yet another development both within the LSODE series and the VODE series that's very significant, and it involves a new theoretical idea, and that's the Krylov method combination. So I should talk about that a little bit. That name refers to a class of methods for solving linear systems—Ax = b, if you like—in which you don't need the matrix A explicitly at all. All you need is the matrix as an operator, which means that given any vector v, you can give me back Av. So there are linear systems methods of the Krylov type and the one that we eventually latched on to was called the Generalized Minimum Residual Method, GMRES. Those were very attractive in combination with stiff solvers for the following reasons; I need to explain this a little bit.

In the stiff solvers, when we need to solve a linear system, the matrix A that occurs there is essentially the Jacobian matrix of the problem -- partial derivatives of the right-hand side function f. It's not exactly that; it's that Jacobian with a scalar multiplier and the identity matrix added. So in order to realize that matrix as an operator, which you would need to use a Krylov method, you simply need to know how to realize the Jacobian matrix as an operator. The Jacobian matrix as an operator, because it is the Jacobian of a given function, can be done just with a difference quotient idea. So if the function is f(y), our difference quotient for that looks like [f(y + epsilon v) – f(y)]/epsilon, as an approximation to the Jacobian of f times v. It's equal to that within higher-order terms, and if the choice of epsilon is done carefully, we can neglect the higher-order terms. It was a natural idea to include these Krylov linear solver methods into a stiff solver, and the first effort to do that was actually done by Bill Gear and Yousef Saad at Illinois. They were working together on this and Saad was really the one doing most of that, I think.

So they came out with something, which I think was a variant of the DIFSUB code, as a matter of fact. Anyway, they wrote a paper about it, and put that out. The rest of the world was very skeptical about it at first, including us at Livermore. But Peter Brown and I looked at it and we decided to try it ourselves (this was in the mid-'80s -- '84, '85). So we generated a variant of LSODE that did this Krylov method and it seemed to work. We were able to solve a number of problems with that rather successfully. But we also knew it was going to be limited in its applicability to certain kinds of problems. But when it works, it's really, really effective because unlike all the other solvers that we had, it never involved storing a huge matrix. So you might be able to solve huge problems with it you could never think of solving before. The thing we did soon after we did that first variant was to add something that made the idea much more robust, much more widely applicable, and that was to add preconditioning. Preconditioning in the linear system solver means that you multiply both sides of the equation by the inverse of some matrix that is an approximation to the original matrix. So you applied the method to the preconditioned system.

HAIGH: All right. So what appears from this would be “LSODPK”. No “E” in this one.

HINDMARSH: No, we wanted to stick to six-letter names. So LSODPK is a solver that’s still part of the collection, very widely used, that uses these preconditioned Krylov methods. It is a package for the more sophisticated user with very large problems. They have to know what they’re doing to use that; they have to be able to provide a preconditioner matrix which approximates the Jacobian in some sense, and yet at the same time is much easier to solve, because now we’re replacing the linear solve that we did directly, with direct methods in the other codes, with a solve with a user’s preconditioner matrix. That needs to be reasonably cheap in order for the whole thing to work right.

HAIGH: It seems that this profusion of different names might potentially confuse people. Did you ever consider calling the whole LSODE and just say something like, “LSODE Version 1.8, includes a new module that does Krylov solutions”?

HINDMARSH: Yes, that crossed our minds. People would suggest maybe just having one single subroutine name that had an option in it to select from all the other choices, things like that. We didn’t pursue that idea just because it would be such a huge single package, if we were trying to lump it all together. And somebody who wants only LSODE would get a whole lot of other things that they really had no interest in.

HAIGH: How large in terms of kilobytes would the code for these packages have been?

HINDMARSH: Each of those solvers alone is on the order of 2 to 4000 lines of code. Of course, they share a lot of routines, so when we put it out as a collection it’s not 9 times 3000 or whatever, but it’s… I don’t know the numbers offhand, but it’s in the tens of thousands of lines of code altogether.

So I should mention that when we had the success with the preconditioned Krylov variant for LSODE, we did the same thing with VODE, and George Byrne was involved again on that. VODPK was a very successful code. So the VODE family really consists now of VODE and VODPK, just those two. They are at the point where, as far as FORTRAN users are concerned, for ODEs, we generally recommend VODE as the first choice. And then if they get to large problems where the direct methods of VODE are prohibitively expensive, then we say try VODPK.

HAIGH: Has the massive increase in the computing power available to users changed anything about the way that you would think about the packages or recognize that they be used?

HINDMARSH: Yes, it has a changed a little bit. The fact that memory has become cheaper has changed our thinking a little bit, and that actually resulted in a feature that VODE has that LSODE does not. VODE, as a default, stores two copies of the Jacobian. What it stores is a copy of the Jacobian, and then a copy of the matrix that is a linear combination of the Jacobian and the identity matrix that is actually used in linear solvers. So when the matrix becomes out-of-date, because of the coefficients of that linear combination being out-of-date, the code uses the saved copy of the Jacobian itself to update the matrix. So the Jacobian doesn’t have to be recalculated again. It makes a decision based on certain data that says that this is probably a good idea, though there’s no guarantee, but that it probably will save time to use a saved value instead of

going back to the user and regenerating a new one. So if the Jacobian is an expensive thing to calculate, this can save a lot of time at the cost of memory. So this was something where we made a trade-off that we think is the right thing to do now that memory is so cheap.

HAIGH: Did anything significant change for LSODE and its variants in terms of the development, methodology, testing, relationship with users, or dissemination mechanism other than the Netlib issue you mentioned earlier?

HINDMARSH: Well, things changed in our methodology as the years went by, just for the reason that our level of support was declining all through the ‘80s. So where in the early years, we were being slow and careful, testing everything very carefully as we wrote code and put it out there. In the later years, we were just more quick about things. We were less cautious, less careful because we wanted to get our codes out but we didn’t have the time and manpower to do as much as we would’ve liked to do with testing.

HAIGH: Did that mean you became more reliant on users outside the lab to discover issues and do beta testing?

HINDMARSH: It wasn’t that bad because we were beyond what we would think of as beta testing, I think, when we put codes out. But we did get more feedback, I think, as time went on, from users who found bugs that were fairly subtle, fairly minor for the most part, and users who said they would like to see more features added.

HAIGH: Did anything change by the 1990s in terms of what you had to do to achieve portability?

HINDMARSH: Yes. Even earlier than that, we certainly learned a lot about portability in a number of ways in going from the GEAR and EPISODE codes into LSODE and VODE. We had used COMMON blocks in the early codes and we no longer used COMMON blocks. Well, that’s not entirely true. The LSODE family does use some COMMON blocks, but they’re a fixed size where as the earlier codes used variable-sized COMMON blocks. I believe the VODE series also uses some COMMON blocks. Portability with respect to double and single-precision was a big issue that we dealt with in a fairly simple way. We evolved into using generic intrinsic functions for the FORTRAN names, for example. When FORTRAN 77 became widely available, which was actually well after LSODE was out there, we didn’t do a lot to go back and rewrite the logic in FORTRAN 77 versus the earlier FORTRAN. We could have; in a few cases we did, but we didn’t do that in a wholesale way.

HAIGH: Was there anything in the hardware of new platforms such as minicomputers and workstations that required any modifications to the code?

HINDMARSH: Not really, except that at the very beginning of the PC development, the FORTRANs that the PCs had were somewhat restrictive. We didn’t consider it a problem with our codes, but the fact that the FORTRANs couldn’t handle our code was really the problem of the compiler. For a while, I was generating a special PC version of LSODE for people, that worked around the limitations of the FORTRAN compilers that were available at the time. Fortunately, that didn’t last very long. Compilers were developed for PCs that were just as good as the mainframe compilers.

HAIGH: By the early ‘80s, do you think a substantial proportion of the users would have switched to minicomputers and other smaller systems, or were most of them dealing with very big problems that would still need a large computer?

HINDMARSH: There were both. It was certainly both, the whole spectrum. In terms of actual numbers of users, the vast majority were using the smaller machines. When we look at the worldwide user community that we were dealing with, PCs and other larger desktops and minicomputers were widespread and those were handling small to medium-sized problems quite well. But at the same time, there were the huge problems that needed to go on the large mainframes. Our codes were portable across that whole spectrum, pretty much. We made sure of that as much as we could.

HAIGH: I have another question and that’s about the ODEPACK project. You discussed in some detail yesterday the personalities involved and the fate of the project. You mentioned that there was some funding coming from DOE specifically for this effort.

HINDMARSH: Yes. Off and on, during the 1980s, little bit in the ‘90s, we had some funding from the DOE. “Applied Mathematical Sciences Research Program” was the name.

HAIGH: Do you know who would have been responsible for providing that?

HINDMARSH: Jim Pool, formerly from Argonne, was the one that was there at the time that the ODEPACK effort started. He was the one that put that name on our list of funded programs when we got money from him.

HAIGH: So you think to some extent, the impetus was coming from him, that he thought there should be something more like LINPACK in this area.

HINDMARSH: Absolutely, yes.

HAIGH: That was just the one issue, I thought, to revive with ODEPACK.

HINDMARSH: Now there’s more to that story, because in later years we were discouraged from doing Argonne-type collections. Before and after Jim Pool was there, when we had meetings with the Applied Math Sciences people they would want to steer us towards the research and development, but away from doing the PACKs. Apparently they felt that that was Argonne’s role and that we shouldn’t be doing that. We disagreed, so we kind of tried to slant our descriptions appropriately, but in fact, we were building ODEPACK.

HAIGH: Was that the result of Jim Pool moving on to something else and someone else taking over?

HINDMARSH: Yes, right, right.

HAIGH: I also know that there was the SLATEC project. Did that impact in any way on Livermore?

HINDMARSH: Yes, it did. I should explain where that acronym comes from. That stood for "Sandia, Los Alamos, and Air force weapons lab Technical Exchange Committee." Sometime back in the '70s, I don't know when, those three laboratories agreed to get together and just share technology in the computing area. One of the things that they agreed on was to try to form a common mathematical library. That's actually the only part of the effort that survived any decent length of time. Livermore was not involved at the time, but Sandia Livermore and Lawrence Lab were added to that collaboration later. We were kind of considered minor members and we didn't have much clout in the organization, and it was dominated pretty much by the Albuquerque people.

HAIGH: Where do you think the impetus for the project was coming from?

HINDMARSH: For SLATEC? Well, it was a good idea, really, in retrospect, to try and develop a common set of mathematical software that all of the labs involved could use. That was actually quite successful in the sense that a SLATEC library was developed, which is quite large. It covers the range of mathematical topic areas very thoroughly. The five different labs I named have that and it's been made available publicly too. So it was a good thing.

HAIGH: Did the labs have special needs that would make it more appropriate to try and develop a shared library between the labs rather than standardize on something like NAG?

HINDMARSH: Yes, I think so. To a large extent, a lot of the software in our libraries, both SLATEC and our own individual laboratory libraries, was driven by local needs, local problem sets that the commercial libraries didn't address. The large stiff system area was certainly one of those areas.

HAIGH: Do you think it would be appropriate at this point to talk some more about Linda Petzold's work and your collaboration?

HINDMARSH: Yes. Linda Petzold was hired by Sandia Livermore in early to mid-1980s, I'm not sure of the date. She worked on a number of computational areas there. She had been a student of Bill Gear, so she knew the ODE area very well. She was given one particular problem there (at Sandia) that was not an ODE, but it was a combination of ODEs and algebraic equations that needed to be solved together as a coupled system. This was not the first time such a thing appeared, but it got a lot more attention from Linda than anybody else had really given it. In order to solve this problem, which she called a DAE, a Differential Algebraic Equation system--

HAIGH: So was that a term that she had coined to describe it?

HINDMARSH: I don't think so. I think that existed before because Gear and at least one student of his at Illinois had done codes for DAE systems before that. DASSL was the resulting code that she wrote in order to solve this problem, and it was an important new development in that some of the methodology that was in the ODE solvers was simply not appropriate for differential-algebraic equation systems, so she had to reinvent the appropriate methodology there. So DASSL was very popular as a DAE solver when she got it done and released. She also believed in public exchange of software and releasing things unconditionally.

There was some controversy about when to use it and when not to use it in the sense that DAE systems include ODE systems as a special case, so people thought, "Well, we don't need ODE solvers anymore; we'll just use DASSL." I had to point out to people that if you have an ODE system, you are really better off using an ODE solver and not a DAE solver because it costs more—there is extra cost to treating an ODE system as a DAE system, so there is an efficiency question. But it's certainly important to have a DAE solver in your bag of tricks along with everything else.

Linda, while she was at Sandia, was collaborating with me on these variants of LSODE as well, as I mentioned. Then at some point, and I don't remember the date, she came across the street from Sandia Livermore to Lawrence Livermore and was group leader at Livermore for a number of years. It was during those years, late '80s, in which she and I and Peter Brown again, collaborated on a variant of DASSL called DASPK, in which we again took those preconditioned Krylov methods that we had working very well in ODE solvers, and put them into the DAE solver DASSL. We did one thing a little different there than what we did with the ODE solvers. She felt that it made more sense, in this case, to have everything in one package. So she wanted DASPK to retain the direct methods, the dense and band direct solvers for linear systems, as well as the Krylov methods, and so all of those options are available in DASPK. You have to specify which of those you want. So DASPK actually supersedes DASSL. It's a much bigger code, of course, but it's a great package and it's out there. It's available through Netlib, also.

HAIGH: Did she have an unusually rapid rise through the lab structure, since she became your supervisor?

HINDMARSH: Yes, she was very effective, very good group leader. She did her research at the same time she did the group leader responsibilities, and generated her book. Her book is very popular; the co-authors were Brenan and Campbell. Toward the end of her stay there, she got dissatisfied, I think, with the way things were going—the same way that I did—and she decided that the academic environment was a better place for her. She left to go to Minnesota first and now is at Santa Barbara.

HAIGH: Has she continued to focus primarily on this area?

HINDMARSH: Absolutely. She has been an international leader in the field of differential-algebraic system treatment, both in terms of software and methodology. She's gotten into a very important aspect of that called sensitivity analysis that Livermore is also working in.

HAIGH: While we're dealing with this general area of the '80s, I know it's really during the late '80s that the Matlab began to become popular on PCs. Does Matlab incorporate capabilities for ODEs?

HINDMARSH: It does now. Larry Shampine put something into Matlab because for ODEs it only had a non-stiff solver for a long time. But he wrote something for stiff systems that's kind of a scaled-down version of a sort of a BDF type method. I really don't know much about it. It's suitable for small and moderate-sized problems, only it's not the kind of thing that we would ever recommend to somebody with a big difficult problem.

HAIGH: So that was not a core area for Matlab?

HINDMARSH: No, not really. Along the same lines, Mathematica, I'm told, has some version of our ODE solvers and I think some version of DASSL is incorporated too. I don't know the details of that; I just was told that.

HAIGH: It seems one of the things that we still have to discuss would be the PVODE parallel package, which I heard you worked on in the 1990s.

HINDMARSH: Right.

HAIGH: The other thing would be to talk some more about the change within the lab and your role in it around the same period.

HINDMARSH: Let me start with that, because that was a change that was underway before we got into the parallel computing business, although they're related. During the mid- to late-1980s, the trend at Livermore was decentralization of computing, and also getting seriously into the parallel computing business. Together, those developments meant that the support we got as a division declined for general numerical analysis and general purpose mathematical software development and math software library development and maintenance -- all of those activities. It was a gradual thing. It wasn't precipitous, but the extent to which we had to find our own sources of money to do what we wanted to do gradually grew and the money just disappeared. There was no source for some of the things we wanted to do. By 1989, the support that I got for numerical ODEs disappeared completely, and the support that I got for doing the library software work as another part of my job disappeared completely, too. So I lost my entire job in 1989 in that sense. I was given some support to work with a group of people that were doing Boltzmann transport, which I won't say any more about, but I worked on that area for a while for the next few years.

HAIGH: Was that something that you found personally satisfying?

HINDMARSH: No, not at all. My role there was pretty much to look at what other people had been doing and kind of provide some quality control with it, help with the documentation. So I got out of that at the first opportunity I had. With the other half of my time, I was told to go find support on my own, and I fortunately had a number of user groups around the lab that I could go to and say, "I'm out of a half a job. Can you help me out?" And the Laser Program did. They were big users of ODE solvers and they supported me half time for a few years there. They were using LSODA and LSODAR and later, LSODKR.

By the way, that's another variant that I didn't mention, and it happened as a result of my collaboration with the Laser Program. They needed the ability to solve large problems with the preconditioned Krylov methods, but they also wanted to do root-finding at the same time, so there was another combination we didn't have. So we put in that combination; LSODKR is the resulting code. That wouldn't have happened without the Laser Program collaboration.

[Tape 3 of 3, Side B]

HINDMARSH: At the same time that support for numerical analysis and software development was declining, we were all very unhappy to see that support for the software library work was

declining, too. It was just a result of a combination of things. We had loss of this tax-type funding; we had the emphasis on parallel work that meant that the serial programs in the library weren't so useful anymore (or were viewed as being not useful anymore—they weren't useless, in fact) and there was pressure to divert the manpower from what we were doing into algorithms for parallel computing. I think it was overdone, but that's, nevertheless, what happened. In particular, in the math software library work, that got down to a point where Fred Fritsch and I were doing it with a little bit of help from a couple of other people part-time. Bob Dickinson was one of those; he was another mathematician in our group. The math software library group was kicked out of the rest of the mathematics organization at one point and put into another part of the computation organization. And then it got down to be Fred and me part-time on this effort.

HAIGH: So what happened to the other members of the mathematics group? Were they scattered near around the lab?

HINDMARSH: Some of them were scattered. For example, the whole statistics and probability group was disbanded as a group and they scattered around the lab or to other places. Some of them went elsewhere. Others who had been doing the numerical analysis work, like me, just found jobs wherever we could and shifted our focus where we could. In the end, for me that meant getting a little bit into the parallel software business. So by 1993, I was convinced, and I managed to convince the people who were supporting me, that we should work on a rewrite of at least one of the ODE solvers that would be suitable for parallel environments. That concept resulted in the code called CVODE, and CVODE is a rewrite in C of the VODE package. It itself was not a parallel solver, but the rewrite in C, with particular attention to how the code was structured, was specifically done in order to pave the way to get into the parallel environment. In particular, the way we did that is to isolate the vector operations into a separate module, so kernels for doing things like taking a linear combination of vectors or doing dot products or taking the norms of vectors are isolated and everything else in the code is independent of whether you're in a serial or parallel environment. But the vector operation module then was going to be an architecture-dependent piece of the package. There would be a serial version and then later there was a parallel version. So that's how we made a parallel code.

HAIGH: Did you like C as a programming language?

HINDMARSH: Well, I never got very good at it and I never got very comfortable with it. I got good enough to do the job and I had a lot of other people around me who knew the language a lot better than I did, so I could go to them for help. It's a language where it's much easier to make errors that are difficult to find, I think. So I was never very comfortable with that. Debugging is always difficult in C for me.

HAIGH: When you talk about pressure to work with parallel systems, would these be systems with just four or eight processors, or the massively parallel machines with hundreds of processors?

HINDMARSH: Yes, they were going massively parallel. They were certainly intending to head that way from the very beginning, although the early machines had four and eight processors. There was an internal laboratory committee that had studied this whole idea and came out with the conclusion that massively parallel was the way to go for many of the problems that the lab was solving. Unfortunately, I think this conclusion was interpreted much more broadly than it

should have been, and so everybody was pressured to put their codes on parallel machines, whether it was appropriate or not. That led to a lot wasted effort, actually. Unfortunately. It would've been better for a lot of the programmatic efforts to simply have more effort on a larger serial machine. But the money to do that wasn't available. The money that was being given out, both internal money and external money from DOE sources, was only available, really, if you were going to end up on a parallel machine. So our support declined there. And software work in general became very application-specific, too; that was a simultaneous development with all the emphasis on parallelism.

CVODE was written in 1993, originally. We got that out. That was funded by some overhead account that we had. But by 1995, we were encouraged to actually submit formal proposals to a special internal funding program called the Laboratory Directed Research and Development Program—LDRD. That was a program for which you had to prepare a thick proposal, present it to a committee, and present slides showing how you were going to develop something that would benefit laboratory programs in a big way, and the programs would benefit so much that there would be a big return on the investment. "Return on investment" was a big term at the point. That was an important aspect of getting this LDRD funding. That also meant that the software was going to be restricted. No longer was it going to be easy to release the software when it was done because the potential return on investment was greater if it could be kept protected and possibly licensed. There were commercial collaborations possibly.

HAIGH: So return on investment was seen from the point of view of the lab itself rather than globally.

HINDMARSH: Right, right. I always kept making the argument that once I finished the code, that there was a big return on investment by having the international feedback from releasing it unlimited. But that argument didn't fly for that period of time, when I was getting this LDRD support. The LDRD support did permit us to do this parallel version, and the name that we gave it was PVODE. So PVODE, was nothing more than the parallelization of the CVODE code that we already had, which, as I say, really amounted to rewriting the vector operations.

HAIGH: How well did the code work?

HINDMARSH: The code worked well. For people who are comfortable with C as an application environment, I think it was easy to use, except for the fact that for large systems they were eventually going to be on a parallel machine. As a user, you need to provide a preconditioner for these preconditioned Krylov methods that were in there. I forgot to mention that CVODE is a rewrite not just of VODE but of VODE and VODPK together. So it has both the direct and the preconditioned Krylov methods. That's an important feature, because the direct methods don't go parallel; it's only the preconditioned Krylov methods that go over to parallel machines. So in that one sense, it's difficult to use. For a given user problem which is so large that it needs to be on a parallel machine, if it's at all nontrivial, you need to do a certain amount of problem-specific development of a preconditioner that would go in as input to the PVODE solver in order to make it work efficiently. We had already seen that the preconditioned Krylov methods without preconditioning have a very limited scope—they don't work very widely.

HAIGH: Was that lack of a preconditioner dictated by virtue of this being parallel or was a result of a lack of resources on the development team?

HINDMARSH: That aspect of the user requirement to provide a preconditioner, that had nothing to do with parallelism. That was an issue in the serial case, too. Parallelism just made things more complicated. For example, if you had a band matrix that was a good preconditioner, then it would be easy to put that into one of the serial solvers like VODPK. But to do that in parallel with PVODE, you would have to then have a parallel band matrix solver in there, and you would have to be able to generate that band matrix in the parallel environment. We never got the support to develop the ability to do that kind of thing. That is, we never got an effective, user-friendly preconditioner generator that could be put into this package and make that preconditioner input job easier for the user. We have a little bit of something there in the way of packaged preconditioners, but there is only one preconditioner for the parallel setting that's there. It's very limited and it doesn't do any communication among the processors if you have multiprocessor environment. So that's an area where the plan that we had laid out before the LDRD committee simply didn't get followed through. There wasn't support given to it. The funding that we got was not sufficient to do any more with that and the manpower wasn't available for it. I was told that certain people were going to be available to help out in that, and in the end they were not available to help out. So that was a mixed result for me. I had that funding for a period of three years, and work on that continued beyond those three years too by me and some others, but it never did fill that big gap of having a user-friendly preconditioner input.

HAIGH: Was it successfully used within Livermore?

HINDMARSH: It was, in fact, in a number of problems, and it continues to be, but only within efforts where there is somebody available to sit down and figure out how to use it and provide the proper input. That's not an easy job.

HAIGH: Was it, in the end, released outside the lab?

HINDMARSH: It was, and that was a struggle because as I say, the return on investment philosophy prevented us from going out with it in the early years. But there was a change of thinking about that after a while and our division leader finally said, "Well, what we would like to do is not send this to Netlib or the Argonne Code Center, where we have no more control how it's disseminated anymore. What we will do is set up a software download site that we run." There is something that the CASC organization has for that. We control that, we require that people fill out a form, and there is an open source license there attached to all the software, and we control how it's done. It took a year at least, as I recall, to convince the lab lawyers that this is okay to do because we had never been explicitly in the software dissemination business in this way before. I mean I was privately sending out diskettes and tapes and stuff, but that was very much unofficial. I wasn't violating any rules because I wasn't really sending out classified information or anything like that. But to have a formal download site like this was really a new thing. We got that through in about 1998 or '99, finally.

HAIGH: Do you think that if you had been able to release code earlier, that there might have been more external collaboration to fill in the gaps in the package?

HINDMARSH: There might have been a lot more input to us for what could be done, but I don't think that much would have happened because we didn't have the manpower to do anything with it. I remember asking some people at Argonne and elsewhere if they had anything available to fill that gap, and they were doing something sort of in that direction. I'm actually referring now

to something called PETSC, which is another big package that has come out of Argonne which is focused on the parallel setting. Its scope includes linear and nonlinear systems and now it includes ODEs because they have an interface to PVODE in PETSC. I asked them specifically, for example, "Do you have anything that would do an efficient generation of a sparse Jacobian on a parallel machine"? That's the kind of thing that I would have loved to have, in the same way that we were doing it on serial machines, very efficiently. And the answer was no, they just didn't have that.

HAIGH: Do you know if during the '90s, there were any other projects explicitly focused on parallelizing ODE solvers?

HINDMARSH: Well, there may have been, but after 1989 I no longer had the mandate to keep up with activities in the outside world in the ODE area, so I didn't know about it if it happened. I don't think there was very much, because actually the push for parallel computing is almost entirely restricted to the DOE laboratories, it seems to me. Although, there are a lot of other university sites that are doing parallel computing, too, but it's not predominant at those other sites.

HAIGH: Would I be right to think that over the first 20 years or so of your career, before the funding situation change, that you were doing essentially the same job?

HINDMARSH: Right. It was good 20 years.

HAIGH: During that time, you never felt that you wanted to move up into a different position or try something different?

HINDMARSH: No, I was very happy doing what I was doing. I was very happy being at what you might think of as the bottom level of the ladder. I consistently told people I didn't want to be a group leader, I didn't want supervisory responsibility because I wasn't cut out for that. Although I never failed to get supervisory tasks dumped on me anyway because it was a natural thing for supervisors to do. But I didn't want the job in any formal way.

HAIGH: Looking just at some other topics with respect to your participation in the broader mathematical software community, were you involved in any way with the ACM Transactions on Mathematical Software?

HINDMARSH: Not in a very direct way. I did some refereeing for them. I did a lot of refereeing; that's another activity that I mentioned. I did refereeing and reviewing for all kinds of journals, and other tasks similar to that involving numerical analysis and software. That was always one of those tech transfer kinds of activities that was important to the field, but hard to fit into the general work scheme and hard to justify taking time out to do. But I tried to, in all my refereeing and my work as an editor, too—I was editor for the SIAM *Journal of Numerical Analysis* for a period of six years, from '75 to '81, and an editor of the *Applied Numerical Mathematics* journal from '85 till I retired in 2002.

HAIGH: How did you become involved with SIAM?

HINDMARSH: Well, I think as soon as I had gotten some publications out on the ODE work and the software development, I had something of a reputation in the early '70s, and I got called by Cleve Moler to be an editor on the SIAM *Journal of Numerical Analysis*. So he in fact gave me that job, and I enjoyed doing it, although I felt that it was a hard job. I tried to be a pretty thorough and careful as an editor and it wasn't an easy job. I felt that six years was a good term and it was time to turn it over to other people after six years.

HAIGH: In those editorial roles, do you think there was anything specific that you did to put your stamp on the journal or maybe cause any topics to be thought up in a different way from how they might have been otherwise treated?

HINDMARSH: Oh, I think only to a very minor extent. I did try to loosen up the philosophy that SIAM *Journal of Numerical Analysis* had in terms of algorithms and software. They had originally been very much opposed to putting in material and papers on algorithms and software, and I kind of liked seeing some of that, although it never became a journal like the *Transactions on Mathematical Software* where they actually explicitly put software in the journal. But there was a little bit of movement in that direction with SIAM. I certainly wasn't the only one pushing for that.

HAIGH: Anything else you think was important in your general participation in the mathematical software community?

HINDMARSH: There were some educational activities that I got into. This is not exactly an answer to that question, but let me mention that because that was something that I did spend quite a bit of time on, as it turns out. First of all, in terms of in-house education, I was asked to do an in-house course on numerical ODEs back in 1973. I worked up a course and gave that in-house. I was very pleased with that. I generated a set of course notes that's over 200 pages long, and then I gave that course several more times over the next five years. Probably as a result of that and the result of the work that George Byrne and I were doing together, we got asked to do a short course for AIChE, the American Institute of Chemical Engineers. That was a good vehicle for presenting some of this new technology that we were now putting out to the world. It happened to be focused on the chemical engineering community, but I thought it was a good start. We did that course twice, starting in 1977, and in 1978 a second time. I did a repeat of that course alone, without George, at the General Electric Research Center in Schenectady another year. I've forgotten which year, but in the late 1970s. So that was kind of fun. That was another way of getting the information out about our methods and our software.

There were two more extended stays that I've had at outside institutions that I haven't mentioned explicitly. One, in 1975 when I spent a month at Argonne, and that happened to be a time when George Byrne was there too, so we continued our collaboration then. It was only a month, but it was a good exposure to another community of users in a computing environment. Later, in 1986, I was asked to spend a month at the University of Trondheim, Norway, by Syvert Norsett. He invited me to come and work with people there and also work on my own, doing development of another piece of software, which I haven't mentioned yet. That was really another rewarding experience because it was an exposure to another way of thinking about a lot of these things. I gave some talks there, and at the same time that I was there, there was a short meeting with Bill Gear and Linda Petzold visiting and also some other Europeans. We just kind of had a small ODE conference there, or really a workshop in which we all gave talks. But my

software work there was to develop a variant of one of their implicit Runge-Kutta codes in which we put in the preconditioned Krylov methods again. So we found out that it was just as easy to put these Krylov methods into a very different kind of stiff solver, mainly one using implicit Runge-Kutta methods, and so there's a code called KRYSI that has this combination in it. It's also publicly available. [KRYSI, An ODE Solver Combining a Semi-Implicit Runge-Kutta Method and a Preconditioned Krylov Method (with Syvert P. Norsett), LLNL Report UCID-21422, May 1988.]

I think I've kind of alluded to one other set of interactions, but maybe I should elaborate on it a little bit. I interacted with the partial differential equation community more and more through the years, because after all, there are more uses of ODE solvers to solve problems that are really from PDEs than there are uses to solve ODEs as such. And so as I did mention, I got hooked on going to the IMACS PDE conferences and giving talks to them, and I collaborated on PDE mathematics issues with a fellow at the lab named Phil Gresho, which was another direction for me completely, because that was not so much working with software, but on analytical issues related to PDE discretizations. We did some fun collaborations that resulted in some publications.

HAIGH: On that topic, you have a long list of publications, most of which appear to be quite tightly tied to these specific packages. I'm wondering, would there be any publications on that list that you would draw attention to as expressing more general ideas that go in scope beyond a specific piece of software?

HINDMARSH: Yes. There's one that I am particularly fond of that I spent a lot of time on for an IMACS congress, and it was a paper called "Current Methods for Large Stiff ODE Systems." And that was a paper where I spent a lot of time looking at other work in the area, not just a little more effort, and tried to summarize all the different approaches that there were for solving large stiff systems, and put it into a single paper. I enjoyed that. [Current Methods for Large Stiff ODE Systems, in Numerical Mathematics and Applications, R. Vichnevetsky and J. Vignes, eds., North-Holland, Amsterdam, 1986, pp. 135-144].

I also got asked to do a survey paper on methods for second-order differential equations in the setting of a workshop on orbital dynamics. This involved people who are designing accelerators and things. And I gave a paper to a conference at Berkeley on that. And second-order equations is not something I had specialized in at all, but I did a survey on that, gave a short paper. [Numerical Techniques for Ordinary Differential Equations, *Particle Accelerators*, Vol. 19, Nos. 1-4 (1986), pp. 67-71.]

HAIGH: So would you say in general you've enjoyed that kind of contact with people, working in diverse application areas?

HINDMARSH: Yes, I have. I enjoy both involvement with different kinds of mathematical problems and the applications that are being seen. I had one collaboration with some people, I remember, that had something called a bubble cavitation problem, which is interesting. It was a small problem, but the solution of it had a curve that looked like a bouncing ball, and it had very sharp cusps in it periodically, and those people's solver didn't know how to handle that. That was back in the period where we had Episode, and Episode had no problem with it. But it had the feature that when you had to go through one of these cusps, in the numerical solver, the value of

the step size, which we called h, and the value of the time t, was such that t + h = t on the computer. In other words, h was below the round-off level in time. A lot of people thought that was a very odd thing and you should never allow that to happen, and I pointed out that there were perfectly legitimate cases where it should be allowed to happen and there was nothing wrong with it. It's just that at the time, you're going along with a time that might be about 1, but the value of the step size might be 10-15, so on a 14-digit computer, t + h = t, and t doesn't look like it's advancing, but yet the solution advances and the solution is correct. So that was another interesting mathematical issue that came out purely as a result of a real problem somebody gave me.

HAIGH: I noticed that one of your papers, a presentation to a conference, it's called "Getting the Power to the People," and it seems, looking at your comments and those of the other people in the stiff computation book, that people generally had a sense that maybe users weren't getting the latest and best methods as effectively as they could have been. [On Numerical Methods for Stiff Differential Equations--Getting the Power to the People, in the *Proceedings of COMPCON 80*, San Francisco, February 1980, pp. 491-495.]

HINDMARSH: Yes. There was a lot of momentum all over, everywhere you looked, it seemed to me, to just pick a method out of a textbook and use it when you needed to, to solve an ODE problem, or to pick up what was available in the office next door or down the hall or in the local library, which may or may not be the best thing available. So I was trying to make the point there that we had stuff that was likely to be better than any of those other sources.

HAIGH: So in the 20 or 25 years since then, do you think that better methods have percolated through to the user community, or do you feel that this is still an issue?

HINDMARSH: It's still an issue to some extent, but by and large, I think the user community has very definitely learned a lesson overall that these things are there and they're available easily. And especially now, with the Internet and software download sites that we have, and Netlib before that, people know that these things are there and available, and as I say, we've seen half a million people get stuff out of Netlib just from ODEPACK.

HAIGH: On page 367 of the Aiken book "Stiff Computation" you make some predictions for the future, and one of those, you say, "Far-term automation will allow a user to get solutions on specifying nothing but the problem itself, i.e., nothing about solution method, tolerances, etc. This will impact the casual users by greatly reducing their setup time." Do you think substantial progress has been made in that direction?

HINDMARSH: Yes, I think a lot of progress has been made. It's not gotten as far as what we would like to have dreamed at that point, and one of the things that has always been a bottleneck there is the idea of tolerances. Users have always had trouble choosing the tolerances that they input to these solvers, and they would frequently say, "Can't you write a solver that doesn't need to ask for that information, and just chooses its own tolerance?" People have tried to do that. But in the end, to cover the full variety of problems that people solve, you have to ask people for tolerance information. You have to say, "Do you want this variable accurate to three figures or do you want it to ten figures?" and it makes a big difference. "And do you want relative error control on that when its value is 10–10, or do you only want absolute error on that to a level that's larger than that number, maybe?" And so you have to get that information from the user.

The solver cannot second-guess that. So there are just hard limits to what you can do in terms of being fully automated, in a sense.

HAIGH: You retired in 2002, is that correct?

HINDMARSH: October 1st, 2002. It was my 60th birthday, and I was pleased to have lasted that long, actually --13 years after having lost my real job. Although I have to say that the last three years of that period were much better than the previous ones because I had a project leader, Carol Woodward, who really gave me a lot of moral support for what I was doing, appreciated what I was doing, appreciated the past history of what I had done. Even though she wasn't in a position to support the kind of thing that I would have liked to do in the organization, the moral support was there, and I enjoyed that. And I've stayed on for that reason, because there is still work to do that I enjoy doing, on a strictly part-time basis of course, but I get the good feedback of helping with the software quality control and documentation and dissemination and seeing the feedback from users on that; that's very satisfying.

There's one other name I should mention for this later period of time. The CVODE and PVODE solver pair, which eventually got renamed simply CVODE with parallel/serial versions, is part of a larger collection that we call SUNDIALS, and it includes a rewrite of other FORTRAN packages, older packages, one for nonlinear algebraic systems and one for differential-algebraic systems. So one of the key parts of that is a rewrite of DASPK in C in the same way that we rewrote VODE and VODPK to get CVODE. So that SUNDIALS collection is now the state-of-the-art for this stuff, at least in terms of C software, including the parallel settings.

HAIGH: Why is it called Sundials?

HINDMARSH: SUNDIALS is an acronym for SUite of Nonlinear and DIfferential ALgebraic Solvers. I'm actually still involved with coauthoring documentation for that and watching the logs of the downloads that happen on the download site.

I've got one other thing here just for fun that answers the question of what people are using these solvers for. It's a question that I am often asked. I looked at the download logs a little bit that we get, for the ODEPACK entries in the software download site that CASC runs. From the period over the last two years and two months, there have been 2,876 downloads there, and there are places on the form that they fill out that says, "What is your intended use?", and "Do you have any other comments?" and so I decided to print the Intended Use line when it was non-blank. Most of them are blank, but you can scan through this and see the wide variety of people's applications: gravitational three-body problem, systems describing river flow, investigation into computer geometry, combustion, kinetics, internal dose calculations, CFD, astrochemistry, astrophysics shockwaves, circuit simulation, combustion simulation, astrophysics again, and it goes on and on. It just delights me to no end to see how people all over the world are using these packages, for things that are really exciting, I think.

HAIGH: Do they still keep in touch and send emails sometimes?

HINDMARSH: Oh yes, they do. My name is still there, although when I finally quit for good, I guess I'll have to take it off, but my name is there as a contact on these packages.

HAIGH: I think moving to conclude, then, I wonder if there's anything that you particularly regret about either the way that your crew developed or decisions that you wish you could go back and have done something differently at some point?

HINDMARSH: Not very much. I really am happy about all those 20 good years, and the 13 bad ones that followed I couldn't do much about. I made my case for the continuation of the things that I thought the organization should be doing, and the people in charge chose not to listen to those arguments.

I think there's one thing I would do differently, back in the period of the ODEPACK discussions and development. I would have pushed the development of that software a lot sooner. Instead of waiting around to see if the consensus agreement on the design would hold up, I would have just pushed ahead. It could have been anticipated that the Sandia people would pull out, given their history, and so I should not have waited for them to agree on that. That's the only thing I can think of, really, that I would have done differently.

HAIGH: And to reverse that, what single accomplishment during your career do you think you're proudest of?

HINDMARSH: That one is tough. The whole collection of ODE software, if I can treat that as a unit, that would be my legacy.

HAIGH: So the routines that were eventually grouped together under ODEPACK?

HINDMARSH: Yes – ODEPACK, and the VODE family, and SUNDIALS.

HAIGH: That concludes the topics that I prepared. If you have anything else to add, then now would be the point.

HINDMARSH: I don't think I do.

Appendix: Resume

Resume: Alan Carleton Hindmarsh

1. Current business address:

Center for Applied Scientific Computing, L-551
Lawrence Livermore National Laboratory
P.O. Box 808
Livermore, CA 94550

Present status: Participating Guest

2. Education:

(a) California Institute of Technology, 1960-1964; B.S. in Mathematics with honors.

(b) Stanford University, 1964-1968; Ph.D. in Mathematics.
Thesis: "Positivity Conditions in Analytic Function Theory."
Advisor: (the late) Charles Loewner.
Held N.S.F. Fellowship, 1964-1968.

3. Employment:

(a) Lockheed Missiles and Space Co., summers of 1960 and 1961; mathematical technician and programmer.

(b) Stanford Linear Accelerator Center, summers of 1962, 1963, 1964, and 1965; mathematical analyst and programmer.

(c) Los Alamos Scientific Laboratory, Meson Physics Division, summer of 1966; mathematical analyst and programmer.

(d) Stanford University, Mathematics Department, October to December of 1966 and 1967; part time assistant instructor.

(e) Lawrence Livermore National Laboratory, August 1968 to October 2002; Mathematician; last position held: Numerical Methods Group, Center for Applied Scientific Computing (Carol Woodward, supervisor).

4. Professional activities and interests:

(a) Primary professional interests: Numerical methods and mathematical software, especially for solution of initial value problems for systems of ordinary differential equations and for systems of linear and nonlinear algebraic equations. Current areas of major interest: algorithms and software for stiff systems of differential equations, and applications to the method of lines solution of partial differential equations. Former areas of interest: complex analysis, analysis of rootfinding algorithms, and Monte Carlo methods.

(b) Job activities: Analysis of numerical algorithms; design, development, and testing of computational algorithms and mathematical software; development of adaptations of general algorithms for specific scientific applications; consulting on mathematical and software matters with LLNL personnel and others; in-house education courses and lectures (including construction of a course on Numerical Solution of Ordinary Differential Equations, 1973); travel to conferences and business meetings (including 4 weeks at Argonne National Laboratory, January 1975, and 3 weeks at the University of Trondheim, Norway, September 1986); editorial board of SIAM Journal on Numerical Analysis, 1976-1982; editorial board of IMACS journal Applied Numerical Math., 1984 - 2002. Have refereed for SIAM J. Num. Anal., BIT, J. Comp. Phys., ACM Trans. Math. Software, SIAM Review, SIAM J. Sci. Stat. Comp., J. Comp. Appl. Math., J.A.C.M., Appl. Numer. Math., Int. J. Num. Meth. Fluids, N.S.F., A.F.O.S.R., and others.

(c) With G. D. Byrne, prepared a short course, Numerical Solution of Stiff Ordinary Differential Equations, for the Amer. Inst. of Chem. Eng. (including lecture notes, 114 pp). Gave course (with Byrne) March 21-22, 1977, and November 14-15, 1977.

5. Professional societies:
   (a) Tau Beta Pi (honorary engineering fraternity)
   (b) Mathematical Association of America (MAA)
   (c) Society for Industrial and Applied Mathematics (SIAM)
   (d) Association for Computing Machinery - Special Interest Group on Numerical Mathematics (SIGNUM)

6. Publications:

(a) Open literature:

Pick's Condition and Analyticity, Pacific J. of Math., 27 (1968), 527-531.

Optimality in a Class of Rootfinding Algorithms, SIAM J. on Num. Anal., 9 (1972), 205-214.

Number-Dependence of Small Crystal Thermodynamic Properties (with W. G. Hoover and B. L. Holian), J. of Chemical Physics, 57 (1972), 1980-1985.

Simulation of Chemical Kinetics Transport in the Stratosphere (with J. S. Chang and N. K. Madsen), in Stiff Differential Systems, R. A. Willoughby, ed., Plenum Press, New York, 1973, pp. 51-65.

A Polyalgorithm for the Numerical Solution of Ordinary Differential Equations (with G. D. Byrne), ACM Trans. on Math. Software, 1 (1975), 71-96.

Applications of EPISODE: An Experimental Package for the Integration of Systems of Ordinary Differential Equations (with G. D. Byrne), in Numerical Methods for Differential Systems, L. Lapidus and W. E. Schiesser, eds., Academic Press, New York, 1976, pp. 147-166.

On the Use of Rank-One Updates in the Solution of Stiff Systems of Ordinary Differential Equations (with G. D. Byrne), ACM SIGNUM Newsletter, vol. 11, no. 3 (October 1976), pp. 23-27.

Panel Discussion on Quality Software for ODE's (with G. D. Byrne, C. W. Gear, T. E. Hull, F. T. Krogh, and L. F. Shampine), in Numerical Methods for Differential Systems, L. Lapidus and W. E. Schiesser, eds., Academic Press, New York, 1976, pp. 267-285.

Note on the Startup of ODE Solvers, ACM SIGNUM Newsletter, vol. 11, no. 4 (December 1976), pp. 12-13.

A Comparison of Two ODE Codes: GEAR and EPISODE (with G. D. Byrne, K. R. Jackson, and H. G. Brown), Computers & Chem. Eng., 1 (1977), 133-147.

Numerical Solution of Stiff Ordinary Differential Equations (with G. D. Byrne), AIChE Today Series, American Institute of Chemical Engineers, New York (1977).

Numerical Solution of a Bubble Cavitation Problem (with G. J.

Lastman and R. A. Wentzell), J. Comp. Phys., 1 (1978), 56-64.

A User Interface Standard for ODE Solvers, in the Proceedings of the 1979 SIGNUM Meeting on Numerical Ordinary Differential Equations, April 1979, Univ. of Illinois (Dept. of Computer Science) Report 79-1710, R. D. Skeel, ed., 1979; also in the ACM SIGNUM Newsletter, vol. 14, no. 2 (June 1979), p. 11.

A Collection of Software for Ordinary Differential Equations, in the Proceedings of the ANS Topical Meeting on Computational Methods in Nuclear Engineering, Williamsburg, Va., April 1979, pp. 8-1 to 8-15.

On Numerical Methods for Stiff Differential Equations--Getting the Power to the People, in the Proceedings of COMPCON 80, San Francisco, February 1980, pp. 491-495.

LSODE and LSODI, Two New Initial Value Ordinary Differential Equation Solvers, in the ACM SIGNUM Newsletter, vol. 15, no. 4 (December 1980), pp. 10-11.

GEARS: A Package for the Solution of Sparse Stiff Ordinary Differential Equations (with A. H. Sherman), in Electrical Power Problems: The Mathematical Challenge, SIAM, Philadelphia, 1980, pp. 190-200.

ODE Solvers for Use with the Method of Lines, in Advances in Computer Methods for Partial Differential Equations - IV, R. Vichnevetsky and R. S. Stepleman, eds., IMACS, New Brunswick, NJ, 1981, pp. 312-316.

New Ordinary Differential Equation Solvers, in LLNL Magazine Tentacle, vol. 1, no. 4 (November 1981), pp. 6-8.

Toward a Systematized Collection of ODE Solvers, in Proceedings of the IMACS 10th World Congress on System Simulation and Scientific Computation, IMACS, 1982, vol. 1, pp. 427-429.

Large Ordinary Differential Equation Systems and Software, IEEE Control Systems Magazine, 2 (1982), 24-30.

ODEPACK, A Systematized Collection of ODE Solvers, in Scientific Computing, R. S. Stepleman et al., eds., North-Holland, Amsterdam, 1983, pp. 55-64.

Implicit Systems of Ordinary Differential Equations, in LLNL Magazine Tentacle, vol. 3, no. 9 (September 1983), pp. 1-7.

Numerical Dynamic Simulation of Solid-Fluid Reactions in Isothermal Porous Spheres (with S. H. Johnson), J. Comp. Phys., 52 (1983),503-523.

The Stability of Explicit Euler Time-Integration for Certain Finite Difference Approximations of the Multi-Dimensional Advection-Diffusion Equation (with P. M. Gresho and D. F. Griffiths), Int. J.Num. Methods in Fluids, 4 (1984), 853-877.

ODE Solvers for Time-Dependent PDE Software, in PDE Software: Modules, Interfaces, and Systems, B. Engquist and T. Smedsaas, eds., North-Holland, Amsterdam, 1984, pp. 325-337.

Reverse Communication in General Purpose Fortran Math Software, in LLNL Magazine Tentacle, vol. 4, no. 11 (November 1984), pp. 2-7.

Stiff System Problems and Solutions at LLNL, in Stiff Computation, R. C. Aiken, ed., Oxford U. Press, 1985, pp. 24-29.

The ODEPACK Solvers, in Stiff Computation, R. C. Aiken, ed., Oxford U. Press, 1985, pp. 167-174.

Experiments with Quasi-Newton Methods in Solving Stiff ODE Systems (with P. N. Brown and H. F. Walker), SIAM J. Sci. Stat. Comp., 6 (1985), 297-313.

Experiments in Numerical Methods for a Problem in Combustion Modeling (with G. D. Byrne), Appl. Numer. Math., 1 (1985), 29-57.

Application of the Method of Lines to the Analysis of Single Fluid-Solid Reactions in Porous Spheres (with H. Y. Sohn and S. H. Johnson), Chem. Eng. Sci., 40 (1985), 2185-2190.

Matrix-Free Methods for Stiff Systems of ODE's (with P. N. Brown), SIAM J. Num. Anal., 23 (1986), 610-638.

Numerical Techniques for Ordinary Differential Equations, Particle Accelerators, 19 (1986), 67-71.

Current Methods for Large Stiff ODE Systems, in Numerical Mathematics and Applications, R. Vichnevetsky and J. Vignes, eds., North-Holland, Amsterdam, 1986, pp. 135-144.

Solving Ordinary Differential Equations on an IBM PC using LSODE, in LLNL Magazine Tentacle, vol. 6, no. 4 (April 1986), pp. 10-18.

Stiff ODE Solvers: A Review of Current and Coming Attractions (with G. D. Byrne), J. Comp. Phys. 70 (1987), pp. 1-62.

The ODE Solver LSODE on the Cray-X/MP, in LLNL Magazine Tentacle, vol. 8, no. 3 (March/April 1988), pp. 10-15.

Numerical Methods for Solving Ordinary Differential Equations and Differential/Algebraic Equations (with Linda R. Petzold), in LLNL Magazine Energy and Technology Review, September 1988, pp. 23-36.

Dynamic Simulation of Reversible Solid-Fluid Reactions in Non-isothermal Porous Spheres with Stefan-Maxwell Diffusion (with S. H. Johnson), Chem. Eng. Sci., 43 (1988), 3235-3258.

Reduced Storage Matrix Methods in Stiff ODE Systems (with P. N. Brown), J. Appl. Math. & Comp., 31 (1989), 40-91.

VODE: A Variable-Coefficient ODE Solver (with P. N. Brown and G. D. Byrne), SIAM J. Sci. Stat. Comp., 10 (1989), 1038-1051.

An Examination of Converting Software to Multiprocessors (with R. E. Strout II and J. R. McGraw), J. Par. Distr. Comp., 13 (1991), pp. 1-16.

Detecting Stability Barriers in BDF Solvers, in Computational Ordinary Differential Equations, J. R. Cash and I. Gladwell, eds., Oxford Univ. Press, Oxford, 1992, pp. 87-96.

"Dundee Hosts 13th Numerical Analysis Conference" (with G. D. Byrne), SIAM News, vol. 23, no. 2 (March 1990), p. 14.

"RK Methods Prove Popular at IMA Conference on Numerical ODEs" (with G. D. Byrne), SIAM News, vol. 23, no. 2 (March 1990), pp. 14-15.

Dynamic Simulation of Multispecies Reaction/Diffusion in Nonisothermal Porous Spheres (with S. H. Johnson), Chem. Eng. Sci. vol. 46 (1991), pp. 1445-1463.

A Dynamic Model for Helium Core Heat Exchangers (with W. E. Schiesser, H. J. Shih, D. G. Hartzog, D. M. Herron, D. Nahmias, and W. G. Stuber), in Supercollider 2, Proceedings of 1990 IISSC, M. McAshan, ed., Plenum Press, New York, 1990.

Using Krylov Methods in the Solution of Large-Scale

Differential-Algebraic Systems (with P. N. Brown and L. R. Petzold), SIAM J. Sci. Comput. 15 (1994), pp. 1467-1488.

Avoiding BDF Stability Barriers in the MOL Solution of Advection-Dominated Problems, Appl. Numer. Math. 17 (1995), pp. 311-318.

A Linear Algebraic Analysis of Diffusion Synthetic Acceleration for the Boltzmann Transport Equation (with S. F. Ashby, P. N. Brown, and M. R. Dorr), SIAM J. Numer. Anal. 32 (1995), pp. 128-178.

Algorithms and Software for Ordinary Differential Equations and Differential-Algebraic Equations (with L. R. Petzold), Computers in Physics, 9 (1995): Part I: Euler Methods and Error Estimation}, pp. 34 -41; Part II: Higher-Order Methods and Software Packages}, pp. 148-155.

CVODE, a Stiff/Nonstiff ODE Solver in C (with Scott D. Cohen), Computers in Physics, vol. 10, no. 2 (March/April 1996), pp. 138-143.

Consistent Initial Condition Calculation for Differential-Algebraic Systems (with P. N. Brown and L. R. Petzold), SIAM J. Sci. Comp. Vol. 19 (1998), pp. 1495-1512.

PVODE, An ODE Solver for Parallel Computers (with G. D. Byrne) Int. J. High Perf. Comput. Appl., Vol. 13, No. 4 (1999), pp. 354-365.

Using an ODE Solver for a Class of Integro-Differential Systems (with M. D. Rotter), J. Comp. Phys., vol. 168 (2001), pp. 267-285.

Application of Parallel Implicit Methods to Edge-Plasma Numerical Simulations (with T. C. Rognlien and X. Q. Xu). J. Comp. Phys., vol. 175 (2002), pp. 249-268.

SUNDIALS: Suite of Nonlinear and Differential/Algebraic Equation Solvers (with P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, and C. S. Woodward), to appear in ACM Transactions on Mathematical Software, 2005.

(Various short notes and problem solutions in Amer. Math. Monthly, Comm. ACM, and SIAM Review.)

(b) Technical reports – 1963-1990 (excluding preprints of published work):

Computational Aspects of Alignment Target Design, SLAC Report TN-63-78, September 1963.

Properties of Symmetric Quadrupole Magnet Triplets (with K. L. Brown), SLAC Report No. 47, September 1965.

Double Drift Buncher (with C. R. Emigh), LASL internal report, September 1966.

Positivity Conditions in Analytic Function Theory, Ph.D. Thesis, Stanford University, June 1968.

The Forced-Collision Method in Time-Dependent Monte Carlo Neutronics, LLNL Report UCIR-437, December 1969.

Composition of Hermite Interpolatory Rootfinding Algorithms, LLNL Report UCRL-50771, December 1969.

ROOT: General Real Rootfinder, LLNL Report CIC-C2.1-004, June 1970.

GEAR: Ordinary Differential Equation System Solver, LLNL Report UCID-30001. Original report, with R. J. Gelinas, April 1971. Rev. 1, single author, September 1972. Rev. 2, August 1973. Rev. 3, December 1974.

Solution of Banded Linear Systems, LLNL Report UCID-30045, April 1972. Rev. 1, June 1978.

The Determinant Method for Calculating Small-Crystal Entropy (with W. G. Hoover), LLNL Report UCRL-51227, May 1972.

Linear Multistep Methods for Ordinary Differential Equations: Method Formulations, Stability, and the Methods of Nordsieck and Gear, LLNL Report UCRL-51186, March 1972. Rev. 1, September 1972.

The Control of Errors in the GEAR Package for Ordinary Differential Equations, Part III of Construction of Mathematical Software, F. N. Fritsch, ed., LLNL Report UCID-30050, Part 3, August 1972.

Numerical Analysis Special Interest Group Meeting: Workshop Notes (editor), LLNL Report M-00876, March 1973.

GEARB: Solution of Ordinary Differential Equations Having Banded Jacobian, LLNL Report UCID-30059, May 1973. Rev. 1, March 1975. Rev. 2, June 1977.

Banded Linear Systems with Pivoting, LLNL Report UCID-30060, May

1973. Rev. 1, May 1976.

DISBAND: Disk-Contained Symmetric Band Matrix Solver, LLNL Report UCID-30065, June 1973.

Numerical Solution of Ordinary Differential Equations: Lecture Notes, LLNL Report UCID-16558, June 1974 (216 pages).

Quality Software for Ordinary Differential Equations at LLL, LLNL Report UCRL-76671, March 1975, presented in a panel discussion at the AICHE 80th National Meeting, Boston, MA, September 7-10, 1975.

EPISODE: An Experimental Package for the Integration of Systems of Ordinary Differential Equations (with G. D. Byrne), LLNL Report UCID-30112, May 1975. Rev. 1, April 1977.

GEARS: Solution of Ordinary Differential Equations Having a Sparse Jacobian Matrix (with J. W. Spellmann), LLNL Report UCID-30116, August 1975.

GEARV: A Vectorized Ordinary Differential Equation Solver (with D. B. Morris), LLNL Report UCID-30119, October 1975. Rev. 1: GEARV and GEARST: Vectorized Ordinary Differential Equation Solvers for the 7600 and STAR Computers (with D. B. Morris and P. F. Dubois), December 1977.

Preliminary Documentation of GEARIB: Solution of Implicit Systems of Ordinary Differential Equations with Banded Jacobian, LLNL Report UCID-30130, February 1976.

The LLL Family of Ordinary Differential Equation Solvers, LLNL Report UCRL-78129, April 1976, presented at the AFWL/AFOSR Conference on Stiff Systems of Ordinary Differential Equations, Albuquerque, NM, May 6-7, 1976.

A Comparison of Two ODE Codes: GEAR and EPISODE (with G. D. Byrne, K. R. Jackson, and H. G. Brown), April 1976, presented by G.D.B. at the AFWL/AFOSR Conference on Stiff Systems of Ordinary Differential Equations, Albuquerque, NM, May 6-7, 1976; U. Pitt. Res. Report. 76-10.

EPISODEB: An Experimental Package for the Integration of Systems of Ordinary Differential Equations with Banded Jacobians (with G. D. Byrne), LLNL Report UCID-30132, April 1976.

A Proposed ODEPACK Calling Sequence (with G. D. Byrne), LLNL Report UCID-30134, May 1976.

DEC/SOL: Solution of Dense Systems of Linear Algebraic Equations (with L. J. Sloan, K. W. Fong, and G. H. Rodrigue), LLNL Report UCID-30137, June 1976. Rev. 1 (with L. J. Sloan and P. F. Dubois), December 1978.

Preliminary Documentation of GEARBI: Solution of ODE Systems with Block-Iterative Treatment of the Jacobian, LLNL Report UCID-30149, December 1976.

Nonsymmetric Conjugate Gradient and Chemical Kinetics-Transport (with G. D. Byrne), NMG Tech. Memo. 77-1, January 1977.

Solution of Block-Tridiagonal Systems of Linear Algebraic Equations, LLNL Report UCID-30150, February 1977.

Comparative Test Results for Two ODE Solvers--EPISODE and GEAR (with G. D. Byrne, K. R. Jackson, and H. G. Brown), Argonne National Laboratory Report ANL-77-19, March 1977.

On Error Estimation in the BDF Method for Ordinary Differential Equations, LLNL Report UCRL-80247, October 1977.

A Tentative User Interface Standard for ODEPACK, LLNL Report UCID-17954, October 1978.

A Collection of Software for Ordinary Differential Equations, LLNL Report UCRL-82091, January 1979, presented at the ANS Topical Meeting on Computational Methods in Nuclear Engineering, Williamsburg, Va., April 1979.

Method-of-Lines Considerations in ODE Software Design, LLNL Report UCRL-82406, February 1979, presented at the 1979 SIGNUM Meeting on Numerical Ordinary Differential Equations, Urbana, Ill., April 1979.

A Study of a Family of Singular Second Order Ordinary Differential Equations, LLNL Report UCID-19207, September 1981.

TORANAGA (with T. S. Axelrod, P. F. Dubois, R. B. Hickman, and J. F. Painter), LLNL Report UCID-30190, January 1983.

A Model Approach for Simulating the Thermodynamic Behavior of the MFTF Cryogenic Cooling Systems - A Status Report (with S. B. Sutton, W. Stein, and T. A. Reitter), LLNL Report UCID-19878, August 1983.

Matrix-Free Methods in the Solution of Stiff Systems of ODE's (with P. N. Brown), LLNL Report UCID-19937, November 1983.

Note on a Householder Implementation of the GMRES Method (with H. F. Walker), LLNL Report UCID-20899, October 1986.

MSRS: A Fortran Code for the Numerical Dynamic Simulation of Solid/Fluid Reactions in Nonisothermal Multispecies Porous Spheres with Stefan-Maxwell Diffusion (with S. H. Johnson), LLNL Report UCRL-21022 (Subcontract 8979505 Final Report), February 1988.

KRYSI, An ODE Solver Combining a Semi-Implicit Runge-Kutta Method and a Preconditioned Krylov Method (with Syvert P. Norsett), LLNL Report UCID-21422, May 1988. Also Norges Tekniske Hogskole Mathematics of Computation Report 8/87, Trondheim, Norway.

Modeling and Simulation of Multispecies Reaction/Diffusion Transients (with S. H. Johnson), LLNL Report UCRL-102327, November 1989, for presentation at the AIChE meeting in San Francisco, November 10, 1989.

Index and Consistency Analysis for DAE Systems from Stefan-Maxwell Diffusion-Reaction Problems, LLNL Report UCRL-ID-103198, March 1990.

Note on CVL Plasma Field Equations, LLNL Report UCRL-ID-105180, October 1990.