An interview with
Jack J. Dongarra
Conducted by Thomas Haigh
on
26 April, 2004
University of Tennessee, Knoxville, TN

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

ABSTRACT:

Jack J. Dongarra describes his professional career to date, with particular reference to his involvement in the production of mathematical software packages. He grew up in Chicago and studied mathematics at Chicago State College, where an internship at Argonne National Laboratory involved him in the ongoing EISPACK project and fostered a life long interest in mathematical software. Dongarra then earned a graduate degree in computer science from the Illinois Institute of Technology, while continuing to work with Argonne staff including Brian Smith, Jim Cody, Danny Sorensen, Jim Boyle, and Jim Pool. Dongarra discuses the organization and functioning of the EISPACK group, the mechanisms used to produce, distribute and maintain code, and his personal role in producing test programs. After graduating in 1975 he went to work full time at Argonne, developing mathematical software. During this period he also studied for a Ph.D. at the University of New Mexico under the direction of Cleve Moler, and worked as a visiting scientist at Los Alamos. From 1976 to 1979 he worked with Moler, Pete Stewart and Jim Bunch to produce LINPACK, a widely used collection of routines for linear algebra. LINPACK also became important as a benchmark for the performance of scientific computers. Dongarra discusses the origins, purpose, organization and results of this project. LINPACK was closely tied to BLAS, the Basic Linear Algebra Subprograms. He also explores its parallels with, and differences from, the later open source software movement. Dongarra received his Ph.D. in 1980, and remained at Argonne, where he developed the NETLIB online software repository (in collaboration with Eric Gross of Bell Labs). In 1989 he accepted a joint position between the University of Tennessee and Oak Ridge National Laboratory. Dongarra discussed his creation and leadership at the University of Tennessee of what became a large research group, the Innovative Computing Laboratory, its sources of funding, and the systems it developed. His next major project was LAPACK, a collaboration with Jim Demmel and several others to create a new library for linear algebra and matrix functions that would maximize performance on shared memory parallel and vector architectures. LAPACK worked closely with Level 2 and Level 3 BLAS, giving a higher level of abstraction from hardware differences and so combining high performance with portability. This led to ScaLAPACK, a version of LAPACK usable on distributed memory systems, and to a more general interest in message passing standards for portable software which Dongarra pursued through the PVM project and the MPI standard. Dongarra also discusses then-current and recent projects, including ATLAS (a system to automatically generate optimized code for particular architectures) and NetSolve, a mechanism to make mathematical software capabilities available as services over a network.

HAIGH: This is an oral history interview conducted as part of the SIAM project on the history of software for scientific computing and numerical analysis. It's Monday, the 26th of April 2004. This interview is being conducted with Professor Jack Dongarra in his office at the University of Tennessee. The interviewer is Thomas Haigh.

Good afternoon.

DONGARRA: Good afternoon.

HAIGH: And to begin with I wonder if I could ask you a little about your upbringing and childhood interests as they reflect your eventual career in mathematical software.

DONGARRA: Right. I was born and raised in Chicago, the oldest of three boys. All of my grandparents emigrated from Sicily; my father's family arrived in America when he was a teenager. I attended a Catholic elementary school, Saint Daniel the Prophet, for eight years it was a typical parochial school: we wore uniforms- a blue shirt and blue tie - and attended the usual elementary classes. I would say I wasn't an especially good student particularly in terms of the grades that I received, most especially relating to abilities in skills such as writing and reading. I suffer from dyslexia, which was undiagnosed, and I think I was probably rated as a rather poor learner during those years. In fact, I remember trying to discreetly scoot my desk across the line from the slower learning group in the back of the room to the more advanced group in the front of the class.

After finishing eighth grade I went to a Chicago public high school, Kennedy High School, on the southwest side of the city. It was a relatively new high school at that time. While taking various courses, I realized that I had a particular interest in science, although I had always had an interest in trying to understand how things worked. I remember one time in particular when my father had just purchased a new electric drill, and I wanted to understand how that drill worked. I took it apart and exposed the inside of that drill, and the thing that I remember to this day is that I was unable to get it back together again. So here's this brand new drill which I had taken apart on my dad's workbench, and I was trying to figure out how it worked, and then was unable to get the thing back together again. My father was understanding, and went out and bought another drill, and I got to keep the one that I had taken apart. I also had an uncle, "Uncle John" who liked photography and has interested in new technology and how things worked. I always enjoyed going over to his house and learning new things. A particular teacher may have sparked my interest in science. I don't remember how to spell his name, but I think his name was Mr. Solange and he was a particularly good teacher. From the moment I started his class, I had an interest in learning more about how things worked at a scientific level. I took courses in math and did okay, but again I didn't excel in anything particular, certainly not in terms in reading and writing abilities.

After high school I decided I wanted to become a high school teacher so I attended a Chicago commuter college that produced many of the teachers in the Chicago public school system. At the time was called Chicago State College, and I think at one time it may have had Normal in its name as well, but while I was there it changed its name to Chicago State University. I was able to take education courses in preparation for a high school teaching degree in science, which was really what I wanted to do. I took math courses and science courses and had a reasonably good grasp of things. In my junior or senior year I realized that maybe high school wasn't the place where I should teach; that instead maybe I should teach at a junior college. So I was going to go off to get a masters degree, and a masters degree in physics is what I decided on. I applied to DePaul University in physics, and they were interested, but something happened in my last year of undergraduate education at Chicago State which changed my plans.

The chairman of the physics department suggested that I apply for a position at Argonne National Laboratory for undergraduates. The arrangement was that the student would work at Argonne with a scientist for one semester of the undergraduate education, earning credit for courses, although there wouldn't be actual courses but rather on-the-job training at the lab. So I applied and asked my professors at Chicago State to write letters for me. The chairman of the physics department, Harvey Leff, had a friend at Argonne, who I think had been one of his college roommates, Jim Pool. Jim Pool was either the director or associate director of the applied math division at that time, and I guess Harvey made a call or did something, explaining to Jim who this guy from Chicago State University was, and that maybe he should be given a look. Because Chicago State University doesn't have a very strong reputation in science, I was competing against people from schools all around the country. People from Stanford and other places certainly have a better shot at this sort of position. Still, Jim Pool decided to try this guy out.

So in my last semester at Chicago State, I had the opportunity to spend those 16 weeks or so at Argonne National Laboratory, living on site, working with a group in the applied math division, and spending my off hours interacting with students from other places. It was a terrific experience in terms of interacting with other students from places that I had never visited, and interacting with people at the lab in terms of development. I had the great fortune to work with someone named Brian Smith, a scientist at the lab, and the unofficial head of the EISPACK project.

EISPACK is a collection of mathematical software that was developed based on algorithms and ideas that had been through the community mainly developed by Jim Wilkinson and his colleagues. Jim Wilkinson was a very distinguished leader in the field of numerical linear algebra, and he and his colleagues had developed state of the art techniques for solving eigenvalue problems. Those methods were developed in Europe for the most part, and were written in Algol and published in a book called "The Handbook for Automatic Computation." Jim Wilkinson and Christian Reinsch were the authors of that book, and the book contained Algol programs which implemented these state of the art methods. Those methods were being translated into FORTRAN for the EISPACK project that Brian Smith was leading. There were a number of other people who were contributing to the effort. Cleve Moler is another individual who was involved in that project and was making significant contributions. So my time at Argonne came at a very important point in EISPACK's history. I was there, learning very quickly, trying to come up to speed in terms of programming and style and ability. At that stage I had only a very crude understanding of programming using APL at an interactive terminal at Chicago State University. That was my first real experience with programming.

HAIGH: Okay, let me slow you down. That's a very good start, but let's just move back over that perimeter and fill in a few other areas. So when you were younger or at high school did you have any, you mentioned that you liked taking things apart –

DONGARRA: Yes –

HAIGH: Did you have any other technology related hobbies like ham radio, or electronics, or playing with chemistry sets?

DONGARRA: Well, I was interested in chemistry. I had a very small chemistry set, and would experiment in the basement of our house in Chicago, mixing chemicals and trying things out and doing things of that nature. Now having a chemistry set often requires having a fire, that is having some source to heat things up, and my parents wouldn't let me do this in my house. I guess they knew better, or were concerned about my abilities at that time to control things, so I was forbidden from having fire in the house. That's my one memory of chemistry, and I had to do everything without source of heat, or if I had a source of heat it was through an electric bulb or something like that.

I don't think I had any ham radio or any kind of electronics of that form. I remember buying a Heathkit and assembling a stereo tuner out of that. That was a fun project to do, but my understanding of electronics was only how to solder things or how to follow instructions. There wasn't really any deep understanding about the components of how they interact and worked and provided what I was in the end going to hear. In terms of other things, I remember having a very crude set of a battery operated devices, called a computational device, with relays or switches which you would operate by turning something, that would carry out some function. It was very primitive, and again it wasn't anything about understanding how it worked, more about how to do the wiring and connect things to get the functionality there. So, I did have a number of interests but they weren't at a very deep level in terms of understanding things.

HAIGH: Was the bachelors degree that you earned from Chicago State University in mathematics?

DONGARRA: It was in mathematics with a minor in physics, yes.

HAIGH: So how did you choose mathematics as your major?

DONGARRA: Why did I choose mathematics instead of physics or chemistry? I'm not sure how that came about. It was probably because I had enough math courses so it was easier for me to get a degree in mathematics than get a degree in physics, although I took physics courses as well. I worked in the chem lab while I was a student there to earn money. There was an opportunity to help and basically be the guy who cleans up or sets up experiments for the classes, and I remember doing that for maybe two years, probably sophomore and junior year and maybe even part of my senior year. So I got to know many of the professors in those departments and interacted with the better students at Chicago State University. I'm not sure why it was mathematics though; I can't put my finger on it. I mean I took courses in all those fields and probably exhausted the courses that they had to offer. I was taking courses in the summer as well. I think I had to satisfy some education requirements to be a high school teacher, and that required additional hours over and above what I was getting. So my undergraduate degree was in mathematics. I had enough courses to actually do the education part as well, that is to be a high school teacher, except that I never did the student teaching. There was a student teaching component and that was supposed to be finished by my last semester, but I decided to go to Argonne instead.

HAIGH: Went the other way.

DONGARRA: Went the other way.

HAIGH: And did any of the courses during this undergraduate degree have any significant kind of numerical analysis content?

DONGARRA: Not particularly. I remember taking some kind of advanced course which I needed to invert a matrix. The teacher wanted us to invert a 16 by 16 complex matrix, and 16 by 16 complex matrix is pretty tedious to do by hand, so I learned very quickly that a computer was the right thing to do. We had access to this terminal connected to a computer running APL, via a time-sharing system. I'm not sure what the speeds were of the connection or even what the computer was connected to. I remember punching in the 16 by 16 matrix by hand and having it compute the inverse for me, and bringing in a piece of paper which contained the inverse, and that was accepted as a good thing. I felt I had done a lot less work than the rest of my classmates who had to sort of work tirelessly trying to get this thing inverted here.

HAIGH: And did you receive any formal instruction in computing as part of your degree?

DONGARRA: Not as a undergraduate. There were no computing courses that I know being taught at Chicago State. The only thing, and I'm not sure why I got to use it, was this APL terminal sitting in the corner. The physics department had one of the early Wang computers, a device which basically had your program on punched cards. I remember being very careful about getting the punches right for this programming language, and I'm not even sure what the language was. There was a card reader which would basically close around your card, and then this computing device would read your program and execute it somehow and then spit out the answers. I remember being very frustrated with the punching holes and making mistakes. It wasn't a very positive experience, and if anything it would discourage one from going into computing rather than to encourage one.

My real computing training, I guess, took place in my last semester at Argonne National Lab, interacting with this group who were building this collection of software called

EISPACK. It was fortunate that I had Brian Smith as my advisor at Argonne, who was very patient and would take an enormous amount of time from his schedule to teach me things. So here I was trying to be helpful, and I was causing more trouble, costing Brian more time than I was giving to this project. But he was very, very giving in terms of providing his time and experience, so the relationship that developed was very good. Brian was, I'd say, very instrumental in my becoming involved in numerical methods, and becoming involved in mathematical software overall. During my time there I learned a little bit and I probably became important because they needed a lot of testing of the software, and I think I was probably a good person to do that. I didn't need to know much about the details of the software. I could run things and report when problems occurred and perhaps fix things up in a minor way if a problem did occur. So I spent my semester there helping with that effort. I made a critical decision after that experience: I decided not to go into physics at DePaul.

I had applied for a masters program, and had been accepted there, but I decided not to take the assistantship which they had provided for me. I decided instead to apply to two schools in Chicago for computer science because of my newfound love of computing after the Argonne experience. It was pretty late in the year, probably May or June, when I decided apply to IIT, Illinois Institute of Technology, to get a masters in computer science. I also applied to the University of Chicago in the information sciences program (I don't think they had a department of computer science at that time). IIT is another case where somebody knew somebody else and that helped. At IIT the chairman of the computer science department knew Jim Pool. Jim Pool was now the director of the applied math division at Argonne, and he was the friend of the guy at Chicago State University, Harvey Left. Jim contacted the chairman of IIT and said "There's this guy who's going to apply. Do you have any opportunities? Maybe you can give him one of the positions." It was very late in the year and as it turns out they had one cancellation. One of those students who had been accepted cancelled late, deciding to go someplace else or do something else. So when my forms came in the chairman decided to give me an opportunity and I became a student with a teaching assistantship at IIT.

The University of Chicago accepted me in the program but they didn't have any financial support and I wasn't capable of bearing the cost. My parents were quite new to this whole thing of going on to high education, and I was the first person in my family to go to college. I have two younger brothers, and neither my father nor my mother had high school diplomas, although my mother had a certificate for a partial high school experience, which was common for girls in her time. My parents knew education was a good thing, so they certainly encouraged it and were supportive of it, but they weren't in a financial way to pay for my education. So during my undergraduate career I was working to support myself. For years I worked at a local pizza place and still keep in touch with some of the other guys who worked with me. I lived at home during college so it wasn't a big financial burden in that sense, and my parents certainly encouraged the whole situation.

So, let's see, going back to the IIT business. After I graduated from college, with my undergraduate degree, I worked at Argonne during the summer, and continued working on EISPACK and interacting with the people there: Brian Smith, Burt Garbow, Wayne Cowell, Jim Cody, and Jim Boyle. They were all staff members at Argonne National Lab who were involved in the development of mathematical software, or aspects of using or working with the mathematical software. The summer at Argonne was a very rich time for interaction, as there were many visitors and students coming through, some for short periods and some for long periods of time. My first meeting with people like Cleve Moler was during that summer, as well as my introduction to Jim Wilkinson, who was again this leader, this incredible person who I knew only by name. Having the opportunity to work with him was really a stimulating experience. And Cleve and I hit things off well from the beginning. I was sort of assigned to him to be the guy who would help him with using the local computer. So here's Cleve, a guy who comes in, is going to work for a month or two at Argonne, and he's never really used the computing resources at Argonne, so I was given the task of holding his hand and working with him if he had any problems, and that was a very good interchange and experience again. I really didn't know who he was and he's a very easy and comfortable person to interact with, very easy to talk to and very easy to learn from. Cleve has a very kind aspect to his personality, and is very giving in terms of providing understanding and support, so I learned a lot during that exchange with him during the summer and the other people came through as well.

As that summer ended I took my position as a graduate student at IIT, where I took courses and really got immersed in computer science, and some of the more theoretical aspects of the field . I tried to learn as much as I could about numerical methods, about computing in general, and about the hardware, software, and theory aspects of computing. So it was a wonderful time learning these things for the first time and becoming involved in them. During that period I worked at Argonne one day a week, so I'd go out to Argonne during the week, probably on a Friday or whichever day I didn't have class. I would drive out to the Lab. IIT is located in the city, near the Chicago lake front, but I lived at home, near Midway Airport, and Argonne is probably an equal distance southwest of where I lived. So it was easy to get around and to do these things, and it worked very well with my particular style of taking courses, and earning a little money, and working with people. Getting exposure to the aspects of software, getting exposure to the aspects of computers, getting exposure to aspects of the numerical calculations that were going on - all of that was very informative.

For my masters' project Brian Smith, who was the guy I first interacted with at Argonne, was teaching a course at IIT. I took his course and he was the person who supervised my masters' thesis at IIT. My masters' project was on an algorithm for reducing a large banded matrix to tridiagonal form in preparation for finding the eigenvalues of the matrix. It was using some techniques for "out of core" manipulation of the matrices, that is trying to reduce a symmetric banded matrix to tri-diagonal form where the matrix is too large to fit within memory; the matrix would sit out on the secondary storage, and you would have to swap the segments of the matrix in to core, and do the calculations, and then swap segments out again. It was interesting from the standpoint of the numerical values, and it was interesting from the standpoint of the computer in trying to optimize how one does things to be very efficient. I was there for about a year and half working. I was a teaching assistant at IIT, a graduate student of course, and one day a week I would go out to Argonne and interact with the people there. I was at IIT for three semesters and I graduated with a masters' degree in Computer Science.

HAIGH: So your resume shows that you received your bachelors in 1972, and the masters in 1973.

DONGARRA: That's correct.

HAIGH: And it has your first participation at Argonne also being in 1973. Do you think that should be 1972?

DONGARRA: That probably should be 1972. That's an interesting point…. Yes, clearly there's a mistype there. Yes, so that should definitely be 1972. Research student associate that was in 1973, that's a good point, thank you. I'll correct that on my vita.

HAIGH: So through that original semester you spent there and then the time you were working on your masters was throughout that was your main activity working on the EISPACK project?

DONGARRA: Yes. So the main thing was doing this EISPACK thing. Helping with the testing, developing some of the test material, working with Brian Smith and Burt Garbow, the two primary individuals. Burt was the person who was in charge of dealing with the assembly of the software, the final polishing and putting things together. I worked with him in terms of putting together test packages that would be sent out to our test sites, and helping with the collection of data, the generation of magnetic tapes and a number of things related to verifying that everything was ready to be sent out and to have things work that way. After I received my masters' degree in the end of 1973, Jim Pool, who was the director of the applied math group, asked if I wanted a permanent position. At that point I was prepared to look for a position perhaps as a junior college teacher in computer science. Jim asked if I was interested in continuing to work at Argonne, and working with Brian with this package called EISPACK. I though it was a great thing, and I took him up on that offer. So my first full time employment as a professional was at Argonne working on EISPACK in 1974.

HAIGH: You mentioned working with Smith, and doing a project based on that?

DONGARRA: Yes.

HAIGH: Were there any of the other courses in the computer science masters that were particularly relevant to this area?

DONGARRA: Something that happens, and you don't appreciate it at the time, is that you're forced to take certain courses and you may not fully appreciate why you're taking these courses. Throughout my education I guess, I was put in a position of having to take courses that I didn't particularly want to take. I felt they were hard and I wasn't sure how I would ever use that information. Almost every one of those courses I draw on today, at some level, using the ideas or the concepts or the ability that I acquired in those courses to help in solving problems I have. So it's a wonderful time, again, to learn and to experience things that you wouldn't normally. So computer theory courses, operating system courses, architecture courses, are all things which are very important, I'll say, to the kinds of things that we do. At that moment I was saying, "I'm don't know why I'm taking this course; it has nothing to do with the things I'll be using in the future," and I was wrong.

HAIGH: So, moving back to EISPACK, do you know exactly when the project started?

DONGARRA: The beginning of the project? There's a book which Wayne Cowell was the editor of and there's a paper in there that I have with Cleve, probably, which talks about EISPACK, and you know I don't see it on my bookshelf which is unfortunate. [EISPACK--A Collection for Solving Eigenvalue Problems, J. Dongarra and C. Moler, Sources and Development of Mathematical Software, W. Cowell, ed. Prentice Hall, pp. 68-87, 1984].

HAIGH: And, actually looking at your resume I also see from 1988 a paper with D. C. Sorensen called, A Look at Software for Dense Matrix Problems Over the Past Fifteen Years. ["A Look at Software for Dense Matrix Problems over the Past Fifteen Years," J. J. Dongarra and D. C. Sorensen, *Numerical Algorithms for Modern Parallel Computer Architectures*, IMA Volumes in Mathematics and Its Applications, 13, Springer-Verlag, 1988.]

DONGARRA: There's something called the EISPACK users' guide that was put together to help people figure out how to use the software and this is the original one; there's a revised edition of this. [*Matrix Eigensystem Routines--EISPACK Guide, 2nd Edition,* B. T. Smith, J. M. Boyle, J.J. Dongarra, B. S. Garbow, Y. Ikebe, V. Klema, and C. Moler, Springer-Verlag, Lecture Notes in Computer Science No. 6, 1976]. The original one I'm not an author on; this came right about the time I was starting at Argonne. The second edition updates and corrects a number of things that were in the original volume.

The users' guide refers to a FORTRAN set of programs, EISPACK, which describes how to use it the software and describes the accuracy, recommended path.As I mentioned, the software that's here was a translation of the software that occurred in this other book called "The Handbook for Automatic Computation", by Wilkinson and Reinsch. [J. H. Wilkinson and C. Reinsch, *Handbook for Automatic Computation, volume 2: Linear Algebra*, Springer-Verlag, New York City, 1971].

HAIGH: Yes, Cleve Moler talked a little bit about that in his interview.

DONGARRA: Right. So this is the book that describes at a very mathematical level what the methods are doing and it provides Algol routines for doing that. And the EISPACK project was to take most of this software in Algol and convert it to FORTRAN, because FORTRAN was being used as the language of choice at that time for scientific users.. It was to be done in a way that the software would be portable, and there was some hint of efficiency. One of the critical things with EISPACK is that it was an attempt to make software portable in the sense that it can go from one machine to another machine without changing much of the software. So we're dealing with numerical procedures, we're dealing with machines that have very different arithmetic characteristics in terms of floating point numbers, the representation for the numbers, and what kind of accuracy is carried in the floating point representation. For these methods it's critical to understand issues about accuracy for convergence matters or properties. EISPACK took on that challenge. As far as I know, that was one of the first packages that looked at developing a portable set of numerical routines and focused on getting the accuracy across a wide range of systems.

EISPACK had a long life. It was done under a project called the NATS, which stands for a number of things depending on when we talk about it. It stands for Northwestern, Argonne, Texas and Stanford, as those are the places where these authors worked. Okay, so Brian Smith, Jim Boyle and Burt Garbow were from Argonne, Yasu Ikebe was from Texas, Virginia Klema was from Northwestern, and Cleve Moler was from Stanford at that time, so that's where the NATS project comes from. The name got changed to the National Activity for Testing Software, and that's probably how it's referred to today in some kind of a document.

HAIGH: Now of those people, are you aware that there was any subset of them who had the original idea? Do you know whose idea it was?

DONGARRA: Original, wow, that's interesting. I'm not sure exactly who originated the idea. Things are a little bit fuzzy at this stage, you know. Cleve was certainly involved in linear algebra and worked with Wilkinson, and so on, so he could have been. Virginia Klema could have been another person who had the idea about translating these things into FORTRAN. They received some funding for this but I don't remember the source of the funds, whether it was NSF. Argonne was involved, so DOE was there. Jim Pool was very instrumental in getting these things off the ground. Jim Pool was the director of the division where we worked at Argonne, and he was very encouraging and supportive of these activities and felt that they were really important for the future of the whole area. When I first started I was young and naive and didn't fully understand how things worked, but Argonne went through a period of time when they were under reduced funding and decisions had to be made on who would be hired and who would be kept on, and who would be fired. Jim Pool made decisions that basically gave these guys a job, and got rid of other guys who were not doing things which were perhaps in tune with the direction for the future. I thought that was a very interesting way of doing business. Brian Smith had been hired very recently at Argonne and he was kept on as the guy who was sort of leading this effort, and other people who were there, mathematicians and statisticians, were let go. They were told to find other jobs because the Department of Energy was not focused on their activity as much as they were on some of these other things. So this shows that there were some very hard decisions to make.

HAIGH: And can talk a little bit about how this group fit into the lab as a whole?

DONGARRA: Well the Argonne National Lab has a number of different divisions and those divisions relate to different technical areas: chemistry, material science, physics, high energy physics, and so on so forth. Each division goes out and acquires its own funding from various sources, primarily the Department of Energy, interacting whenever necessary to do business. Now the division that we worked in was called The Applied Mathematics Division and we received funds to do work in applied mathematics and computer science research. Those monies came from the old program that funded the applied mathematics efforts within DOE. In addition, our division ran the computing facility so we had fairly powerful computers for those times. Today our laptops make those machines look like very slow devices, but they were very impressive in their day. When I started we had an IBM 360/75 computer, which was a pretty fast machine. People from various divisions would come to use the punch card reader, interact, and get output, so we saw people on a daily basis from other divisions.

HAIGH: Was the machine housed within your group?

DONGARRA: It was housed within our building. We had a group of people doing the research and then there was another group looking after the machine, so we didn't have to take care of the machine. We used the machine in ways which were perhaps more innovative than others. We had time-sharing, so I don't ever remember punching a card at Argonne, although I may have and don't remember. I do remember sitting in front of a terminal and typing stuff in, submitting jobs through the terminal, editing files on the terminal, and doing things like that. Other people had punch cards and paper tape which they would bring to our building and read into the computer.

HAIGH: So some of the people within that division were involved in servicing researchers in other parts of the lab, and some people were doing research of their own?

DONGARRA: Absolutely, yeah. People would come and talk to us about their problem but we weren't there to service them specifically, we were there to do our own stuff. We had our own agenda, but we would clearly be interested in listening to other people's problems, and helping them by giving them our software as well.

HAIGH: So you've said that your role on the EISPACK project was mostly in terms of testing.

DOGARRA: Well it was in terms of testing and development of the software library. I was a young guy who was just starting at that period so I certainly didn't have the skills that other people had. You have to understand that I didn't have any formal training in programming at that point and my interaction was pretty naive, I will say, in terms of the programmability of these things, and numerical understanding. Having said that I was in a situation where I was able to interact with people who were very skilled and at the top of their field, so it was very easy for me to acquire the skills. The people were there and it was easy to interact with them, they were very giving of their time and very patient with me in terms of teaching these things. I've always found myself in situations where people were very eager and helpful in terms of their resources.

HAIGH: Can you say a little bit about the main contributions of the other project participants, who was doing what.

DONGARRA: I'll try to do that. Brian Smith was in charge of the project overall. Jim Boyle was involved in the development of a system which was a high level interface to the EISPACK collection. EISPACK was a collection of software, a bunch of subroutines. I don't know the exact number, but I'll say it was made up of 50 subroutines, that a user had to call and put together in a certain way to help solve an eignevalue problem . Jim Boyle came up with a software tool which allowed a user on an IBM system to make one call to a routine called EISPAC, spelled without the K because FORTRAN has this characteristic of only allowing six characters in its name. That routine, if the user had specified key words correctly, would call the underlying individual routines from EISPACK in the right order to help solve the problems. It was a high level interface to this mathematical software, but it only worked on IBM systems, so it was very system dependent.

I don't remember what Yasu Ikebe's contribution was. He was there before I came. Virginia was there, and again I'm not really sure about her contribution, although she seemed to have more of a statistical background. Cleve had played a major role in developing some of the key components for EISPACK itself, so Brian and Cleve were the two guys I remember as being the sort of technical leaders. Burt Garbow made significant contributions as well, mainly to the look and style and form of the software itself. He was the person who had the job of putting in the documentation, making the software look presentable, making sure that things worked correctly, cleaning things up and getting them to the point where they could be made into software for the masses. So those are the roles.

There was another edition of the package that came out later which extended the package once again and presented another set of routines . At this stage, the four of us - Garbow, Smith, Moler, and myself - were the were the main contributors to EISPACK . [*Matrix Eigensystem Routines--EISPACK Guide Extension*, B. S. Garbow, J. M. Boyle, J. J. Dongarra, and C. B. Moler, Lecture Notes in Computer Science No. 51, Springer-Verlag, 1977.] Burt Garbow was a sort of leader in terms of putting things together, so that's why his name is the first author.

HAIGH: Do you know where the funding for the project was coming from?

DONGARRA: I don't specifically. Most of the funding came from the Department of Energy. The Department of Energy was funding mathematical software, which was relatively new, and Jim Pool was very instrumental in getting the funding secured for us and in place for the project. That kind of software is always one of these funny things in terms of providing research funds. If you're writing software, many people ask the question "where's the research contribution in working on software, how does it affect us and why should we be paying for this?" So that's been a constant source of problems from the political side in funding mathematical software on a level with some of the research that's done in other areas. And sometimes we'd have to hide the fact that we're writing software, so we'd say that we're developing portable techniques, we're developing techniques which are robust, and work well under fire and the software's a byproduct of that research and could be useful to other people.

HAIGH: Yes. So EISPACK was intended as a reimplementation of these techniques that had already been published by Wilkinson?

DONGARRA: Yeah, Wilkinson and his colleagues. Yeah, Wilkinson's, Reinsch's stuff in Algol, that's correct.

HAIGH: So, in the end were any additional methods added, or did it stick within that scope?

DONGARRA: No, they were added. Some new methods were becoming known and the state of the art was changing, so as that happened within the life of the project, new methods were included. Cleve Moler and Pete Stewart created a new method for finding the eigenvalues for generalized non-symmetric problems which wasn't included in the original material, so it was put into EISPACK and made available to the community through that form. There are other examples of that as well . A number of things have been changed, and the methods were also improved although they were very good from the original handbook. Improvements were made in the numerical accuracy of the methods, as well as in things that enhance the performance of those methods on a FORTRAN system.

HAIGH: You mentioned that the other main contribution, the innovative thing about the project was portability.

DONGARRA: Yes.

HAIGH: So how did you go about achieving portability?

DONGARRA: Portability meant that it had to work and had to compile clearly but it had to work from the numerical standpoint under different environments. We had a network of friends willing to engage in this testing for us. We would send out magnetic tapes of the software, and ask that they run those tests on their machines and report back to us the results. One of my main functions within the project was to put together some of the tests, to send them out to these places, and then to receive back problems and look at ways of correcting or addressing those problems in terms of getting it to compile correctly, getting it to run successfully on these systems. In those days these machines were quite distinct: the floating point formats were different, numerical precision was an issue, and the standard mode of operating in terms of single or double precision was also one of the functions. The machines we had were not terribly powerful by today's standards, but they were the machines at the top of the heap at that time. So we had a network of test sites that we used to whom we would send our tapes through regular mail and then wait for them to be run and to report back the results. We were talking about IBM, CDC, and PDP computers that are used to run these programs.

HAIGH: Do you know if there were any internal technical reports that might include more material on the early stages?

DONGARRA: Yeah, so where are those technical reports today? It's an excellent question.

HAIGH: I see there's a 1974 document listed on your resume, an Argonne National Laboratory Technical Memorandum, called Path Chart and Documentation for the EISPACK Package of Matrix Eigensystem Routines. [Path Chart and Documentation for the EISPACK Package of Matrix Eigensystem Routines, J.J. Dongarra and B. S. Garbow, Argonne National Laboratory Technical Memorandum, ANL AMD-TM-250, July 1974 (updated August 1975)].

DONGARRA: Yes, so that would be a wonderful one to have, so that would be ideal to get a hold of.

So looking here there's a reference there to Jim Boyle, an Argonne report, which talks about his program called EISPAC, and must describe the early stages in some detail. So the testing was done on IBM machines, Univac, Amdahl systems, DEC systems, Honeywell machines, CDC equipment, and at a wide variety of places as well.

HAIGH: And when you sent out the program tape were there also a set of standard exercises that they should run with it?

DONGARRA: Right. So what the test sites were expected to do was to install the library and then run a set of test programs against that library. So they would have to compile and verify that things were ready to execute, or ready to be linked in. We would also send the test programs. The test program was a program that made calls, which would read some data in. There were extensive data sets of test matrices, chosen because they provided interesting numerical challenges for the software.

HAIGH: And did you design those test matrices?

DONGARRA: They were collected over a period of time, and I don't even know the origin of those matrices, of the matrices themselves. Some of them come from a book by Gregory and Karney,which contained a collection of matrices for testing computational algorithms. I notice there that the book was Wallace Givens' book. Wallace Givens was a researcher at Argonne, sort of on a par with Jim Wilkinson in terms of his being a leader and founder of the field of numerical linear algebra.

HAIGH: Yeah. So this book is surprisingly early 1969, Robert T. Gregory and David K. Karney, and its title is "A Collection of Matrices for Testing Computational Algorithms", published by Wiley Interscience.

DONGARRA: So that was a book that we used to draw on for test matrices. Part of the deal was putting together this test package which would read the matrices from a file, then make calls to the various EISPACK routines to verify that they were doing the right thing, and then put together a set of residual tests to verify in fact that they contained the right answer. You want to have an internal consistency test so they would compute a residual based on the original matrix and the solution they computed, to verify in fact that they were doing the right thing. My main responsibility with EISPACK was generating those programs and verifying that we had all the testing done correctly so that we knew we were going to be releasing software that had some level of scrutiny or testing. So the software was put together and went through this rigorous period of tests and changes and checks, and ultimately, would be released through something called the Argonne Code Center. The Argonne Code Center made the tapes and made it available to their subscribers. That was one of the official repositories for Department of Energy software, and it was the way in which people were expected to get the software. At some point I think it was free and later they started charging for their services, and charging for their services I'm sure led to their decline.

HAIGH: Do you know if the project had been influenced in any way by any earlier project to gather together, or distribute, or standardize scientific routines?

DONGARRA: Jim Cody had a package which was called FUNPACK, but I'm not sure which came first, that project or EISPACK. Jim Cody had a set of routines which were basically to perform elementary functions and calculations. He had test routines and so on, and went through a similar process that we went through with EISPACK, in terms of beta testing, so that may have predated us slightly. He was just down the hall from us and we exchanged ideas all the time on ideas for portable software.

HAIGH: My impression is that that was simultaneous or slightly later. I'm aware that in the late 1960's there was a project at Bell Labs where they attempted to put together routines for a portable library.

DONGARRA: That's right. So Bell Labs had a thing like that and Phyllis Fox was involved with it.

HAIGH: And it's not quite the same thing, but I know SHARE set up a numerical project, also in the mid 1960's, where they were attempting to test different kinds of routines and come up with some kind of best of breed collection.

DONGARRA: That's correct. SHARE had a bunch of things, and one of the guys at the University of Chicago was instrumental in that. Kuki was his name, he was a collaborator with Jim Cody on some of these things.

HAIGH: They have some papers from that project in the archives of the Smithsonian, I should be able to read some of the correspondence and look at the original reviews.

HAIGH: But it's occurring to me that this might be the first project which constructed and gave out to people this kind of standard set of computerized routines. Do you know if that might be true, or are you aware of any earlier?

DONGARRA: I don't remember anything earlier being discussed. So, what you're saying is that this is the first such collection of software which is portable, which contained a set of test routines, which would verify the fact that these routines were doing what they were said to be doing, and that was freely available to the community.

HAIGH: Yes. I need to do some research to find out whether this idea of a suite of compliance tests for computer language specifications predated this, but at least it's the first I'm aware of in this scientific software area.

DONGARRA: It could be, it could very well be.

HAIGH: Comparing this to the more modern kind of model for an open software project, it seems that at least some of the pieces are there, inasmuch as it's being tested by a wide group of users on different platforms, and of course the source code itself is available. On the other hand it seems, and I'd be interested to hear if you agree with this, that there's a difference because the users would report bugs and then a small centralized core of people would be involved in fixing them. So the idea that some group of the users themselves will contribute additional modules or would correct bugs isn't there.

DONGARRA: I would say that that's probably the way it happened for the most part. That is people would find situations where either the code didn't compile on their machine - could be a compiler bug or obscure issue, it could be that the routine didn't do what it was supposed to do from a numerical standpoint for our tests, in which case we would look at it; or it could be that they had a matrix where the routine behaved as expected, so it would either give an inaccurate or faulty result, in which case they'd report to us that test matrix. Sometimes the users would figure out what was going on, they had the source code, they were able to track it and they were able to pinpoint precisely where the problem was and report it to us and we would take some action based on that. And other times, and probably the way it happened for the most part, we would end up tracking down what was going on and what went faulty with the routine and try to correct it from a numerical standpoint back at the ranch.

HAIGH: But at least in some cases users contributed their own patches along with the bug report?

DONGARRA: In some cases that was correct, but I wouldn't say that would be the normal case. Most people said "your routine doesn't work and here's the matrix where it failed" or "the routine didn't compile on my machine" for whatever reason, and we would try to figure out why and try to work around it. Many of those times it was because of the compiler not doing something correctly. We tried to write the software in a way that was not using any exotic features of the language. Maybe that's the wrong word to use, but using pretty standard vanilla FORTRAN, so there was no COMMON used, no EQUIVALENCE, and a number of other things we tried to stay away to avoid potential problems.

HAIGH: So I believe that EISPACK itself was released in 1974.

DONGARRA: 1974. Probably right, the first “official” released.

HAIGH: Now after it was released did people continue to send in bug fixes, or bug reports and suggestions?

DONGARRA: They do it today. That’s something that goes on and on, and today it’s often because of misuse of the software. People pick up a piece of software and try to use it, but they haven’t read the documentation, so they don’t understand fully how the calling sequence works, and then they say your software doesn’t work, and here’s the example. It’s obvious that they’ve just not read something and yet they expect us to debug their program, so that becomes a very frustrating thing. Early on, there were reports that revealed that some things were incorrect, and we fixed them. Occasionally we came across a situation where a matrix provides a challenge for the routine and may cause the routine to do something we didn’t expect, and then we would think that at this stage we’re probably not going to make changes to EISPACK; but we think about those changes in the context of the algorithm, about how we may change the algorithm as it is embodied today in software that we might run. So in the course of things EISPACK has a living complement which sits inside of LAPACK, which is another package that I’m sure we’ll talk about.

HAIGH: Yes. And do you have a sense of what kinds of people were using EISPACK in the early days and approximately how many installations it was supplied to?

DONGARRA: I don’t remember this number, but we would get reports from the Argonne Code Center about who was acquiring our software, and at one time I thought it would be interesting to have a record of that, so I had a map of the world on my wall of my office and I would put a pin in for each instance. I can’t recall exactly where pins were in this map but it was quite striking, many of them in the U.S. but also outside of the U.S. as well. There were hundreds of places that had the software, but I just can’t recall the number of places that had requested the software from the Argonne Code Center. Today this is a little artificial in the sense that we have this thing called NETLIB. NETLIB is this repository that tries to contain all the software and EISPACK is collected there. The part that’s artificial is this number: today there are 1.4 million hits to the EISPACK collection here in Tennessee for things underneath the directory called EISPACK. So as we may learn later, NETLIB is this repository of software which sits in a number of places around the world today, and the one here in Tennessee and the one at Bell Labs are the original ones for that and we get a lot of traffic for that software.

HAIGH: So the people who would directly request and install the package presumably would be staff working at some kind of computer center?

DONGARRA: Right, that’s right.

HAIGH: Did you ever hear from the people who would then come along to the computer center and have a matrix they needed to do things to using the package?

DONGARRA: Occasionally, and sometimes they would contact us directly, sometimes through the person who did the installation at the test site. On every routine in EISPACK is the name of an individual… so Burt Garbow is the individual who was listed, he was the guy who would respond to these things, he would be the interface to this stuff. And he would have the charge of sorting that out, trying to figure out what the real issue was. Here it just lists his name, "Questions and comments should be directed to Burt Garbow, Math and Computer Science Division, Argonne National Lab" and doesn't give his phone number, which is a good thing for Burt.

HAIGH: And in order to use the routines, am I correct, someone would write a FORTRAN program and then they would link to that library, and in their program they would include some calls to the library?

DONGARRA: Right. So EISPACK is composed of 70 or so routines. The way this works is that the user has to put together a set of these routines to solve a problem. Wanting to find the eigenvalues of a non-symmetric matrix may require putting together four or five routines to solve that problem. One routine does some balancing, another routine reduces a matrix to a Hessenberg form, another routine computes the eigenvalues of the Hessenberg form, another routine back transforms the eigenvectors, another routine undoes the balancing. So there would be a string of routines that the user had to call, they had to get that sequence correct, get the arguments correct. Finally we put together a set of driver routines, "one stop shopping" as we called them, where a person calls one routine which calls all these other guys internally, and then solves the user's problem. And that made things a little bit easier for the average guy to call these routines, but everything was there so if somebody wanted to experiment and replace a routine with some other version of the routine they could actually do that.

HAIGH: Do you know of cases in which users did that, where specific installations might change or replace some of the code?

DONGARRA: I don't know of a place exchanging a routine. I would say that researchers in the area would think of putting their own version of a certain routine in place and then testing it out, trying to come up with a method which was better than something that was in EISPACK. EISPACK was a standard that you had to match your method and software against, and people tried, of course, to produce routines which were better, better in the sense of more accurate, or better in the sense of faster, than the EISPACK routines.

HAIGH: So in that sense there were two groups of users: the people who actually had a problem to solve, and the people who wanted to use EISPACK as an experimental framework within which they could try out new methods?

DONGARRA: Right. So the latter is a smaller group I would say. Those people who are interested in numerical software and methods for solving problems in linear algebra would develop a method they thought was better, they would implement it in FORTRAN, match it against the EISPACK package, and we would hear about if they had a method which was better. We would look at it and then there was a chance that it could find its way into the collection, for some future version.

HAIGH: So from 1975 onwards you would be working full time at Argonne National Lab?

DONGARRA: Yes, that's right. I worked full time on developing mathematical software for the most part, mainly on EISPACK. At this period of my employment and education I was learning a lot, I was using my skills to develop software which people were using, and we had visitors coming to the Lab all the time. So Cleve Moler, Jim Wilkinson, Christian Reinsch, Pete Stewart, and other people, would come to the Lab and I would interact with them. These guys would say, "You're doing some interesting things here, this is great stuff." They knew my education background, and because I was a young guy at that time they said maybe you ought to think about going back to school and getting a Ph.D.. I thought that would be an interesting thing. I never thought I would be focused on education. I was surrounded by people who had Ph.D.s and felt maybe slightly inferior to them, because I didn't have the same training or background or understanding that they did, and having a Ph.D. would certainly make me more marketable and would enhance my skills and other things along the way. Argonne had a program for people to go back and get their degrees, so I was encouraged by my colleagues at the Lab, and those who came to visit the Lab, and when I talked my supervisors, Jim Pool and Brian Smith, they thought it was a great idea.

When I looked around to understand how I would do my degree, there were several choices. Because of my background I'd say I probably wasn't qualified to go to some of the top tier university; but, I had people who were very supportive of me and I wanted to work with somebody who would be easy to work with and learn from, and that's where Cleve comes in. Cleve was a natural for me in terms of interacting. We had worked as colleagues and interacted at a different level than student/teacher, and I knew that could be a problem, but I was certainly willing, given Cleve's personality, to be engaging, and his willingness to help was important. He was in the math department at the University of New Mexico at that time. Now I don't consider myself a mathematician, and the University of New Mexico didn't have a Ph.D. program in computer science at the time. Cleve knew that, and knew my limitations in terms of my math background, but he said we'd try to make it work. "We'll try to construct a program that makes sense for you," is the way he put it, and he said that I wouldn't have to do all the crazy math stuff, and that we'd make it so it fit into my interests and provided me with reasonable knowledge to work with in the end. That worked out well in the sense that I was able to move into that program.

I should back up and say that before I started my first real semester there at the University of New Mexico, I was able to arrange for a position at Los Alamos National Lab. Cleve said I should do this. I had talked to the people at Los Alamos and said that maybe what I should do first is to go off to Los Alamos National Lab, work there with some people and take a University of New Mexico night courses which was taught at the Lab.

HAIGH: And your resume shows this visiting scientist, Los Alamos Scientific Laboratory, 1977.

DONGARRA: Yes, that was a wonderful experience. It had been arranged that I work with Bill Buzbee, who was in charge of a group there called C3, they with developing the numerical methods used by Los Alamos National Laboratory. I worked there with a number of people. It was a very critical period in terms in computing, critical in the sense that Los Alamos and Livermore have always had the fastest biggest computers, and the biggest and fastest machine at that time was a Cray 1 computer, which was at Los Alamos. I had a wonderful experience learning about this computer, using it and trying to make our software run fast on this machine. It was a very critical period in terms of being exposed to vector computing for the first time, trying to understand how that hardware worked, and what we could do to make our software match better to the architecture of that machine.

HAIGH: How would you compare the kind of culture and concerns of people at Los Alamos with the environment you were used to at Argonne?

DONGARRA: How would I compare it? There were people at Los Alamos who understood and were supportive and were doing things similar to what we were doing at Argonne. There were people working not necessarily on numerical linear algebra but on other things related to high performance mathematical software. For instance, there was a researcher at Los Alamos who was doing work on elementary functions similar to Jim Cody’s work. So there were similar ideas in terms of developing portable software that would run efficiently or robustly on a number of different machines.

HAIGH: And were there any important packages that came out of Los Alamos?

DONGARRA: Yes. There’s an elementary function package which has its origins at Los Alamos. Wayne Fullerton was doing some of these things. His software is available today, in fact it must be here somewhere in this library.

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

HAIGH: We’d got as far as your first visiting position at Los Alamos, which you said was just before you started the Ph.D. program.

DONGARRA: Right, that’s right. I lived at Los Alamos and worked there before formally starting the Ph.D. program, although I was taking courses. The University of New Mexico had an extension program at Los Alamos which provided night courses, so I took two math courses during that first period in New Mexico. That was a really good experience as it got me in the mood for doing mathematics again, and being at Los Alamos provided this incredible stimulation to work on some of the fastest computers, to interact with a different group of people, and to experience New Mexico, which is a delightful place. I worked with a number of people. Bill Buzbee was sort of the boss, and a guy named Tom Jordan was very influential in terms of my understanding the vector supercomputers. Tom Jordan was using the Cray, and developing ideas and techniques for making things very fast on that machine,. We had long conversations and he would share his experiences and I would try to do things and to extend some of his ideas, so it was a lot of interaction. It was a great period of time.

HAIGH: And were you on leave from Argonne at that point?

DONGARRA: I think I was, I may even have been paid by Argonne at that time. So I was still an Argonne employee working at Los Alamos, on some kind of exchange program. I don't think I was receiving a check from Los Alamos. I think the official way it worked out was that I was receiving a check from Argonne National Lab, living in Los Alamos, working at the Lab, but still an Argonne employee. Argonne had a program that paid a partial salary when I formally started school. It wasn't a lot, and I can't remember the exact number, but maybe it was like ten percent of my salary which I received while I was a student, under the assumption that I would return to Argonne after I received my degree. When I formally started on campus at the University of New Mexico, Los Alamos hired me as a consultant. I worked there one day a week and would fly up on Fridays on Ross Airlines. That was an airline that serviced the Lab and flew from Albuquerque airport to Los Alamos' airport and back only, maybe four or five flights a day, I got up early in the morning, almost before the sun rose, and drove to the airport, took a flight up to Los Alamos, took a taxi over to the building I was working in, and flew back at night when the sun set again. It was just a terrific, terrific experience. I had a lot of fun while I was a student, although I didn't feel much like a student compared to some of my peers at the University. My situation was rather unusual as I was a graduate research assistant to Cleve and I worked with him 20 hours a week, I went up to Los Alamos one day a week, and took courses and received some pay from Argonne while I was doing this. It was a very unusual trajectory for a student, I'll say. I wasn't quite the poor student living in the student ghetto, but it worked out fine in the end.

HAIGH: And what kinds of things did you work on with Cleve?

DONGARRA: Well, yes, so we have to unwind things just a little bit now because the EISPACK project was, I wouldn't say it was finished, but it wasn't the big project anymore. LINPACK was the big project now. LINPACK is a package of mathematical software for solving problems in linear algebra, mainly dense linear algebra problems, and that project had four primary contributors: myself, Jim Bunch from the University of California-San Diego, Cleve Moler who was at New Mexico at that time, and Pete Stewart who was at the University of Maryland. That project had its origins in 1974, when Jim Pool started a series of informal meetings to discuss the possibility of producing this package.

HAIGH: So LINPACK was Pool's idea?

DONGARRA: I'll say that Jim probably had the idea of putting that all together. Jim was instrumental in EISPACK, in getting things going in EISPACK, and also instrumental in terms of LINPACK. The interesting thing about LINPACK was it was funded by both

DOE and NSF. A proposal was written and NSF funded the project for three years, beginning in 1976. That project, again, had these ideas of trying to understand the mechanics of software production, understanding the issues about portability, trying to put together packages that were efficient and robust in terms of being able numerically to solve generally difficult problems, and producing documentation that could be used and made freely available.

HAIGH: So officially the research was being supported more as software engineering than as mathematics?

DONGARRA: Correct, yeah. It was a software engineering project at some level, but it was a way for us to produce software and perhaps hide it under that cloak so that it was freely available.

HAIGH: So the release date you have here for LINPACK is 1979. Can you remember when work started?

DONGARRA: Well, let's see, it was a three year NSF project, and that started in 1976, so 1979 would be the culmination of that effort I would guess, in terms of releasing software. So from 1976 to 1979 we worked in trying to put this stuff together, working on it individually and then coming together in the summer. So those four individuals - Bunch, Dongarra, Moler and Stewart - came together in the summer at Argonne, to collectively put together various pieces that they had worked on over the year and trying to make it all work. I remember those meetings as being very intense in the sense that these guys would come for a summer period, probably six weeks to two months, and work very hard. I remember many nights working with Cleve in particular, sitting around a terminal or an early workstation, trying to figure out some aspect of the software that we were developing. And it could also be very contentious as these guys would have been off doing their own thing and then they'd come together, having very distinct ideas about how the software should be written. We'd have loud discussion with yelling and and perhaps screaming, well not quite screaming, but we got quite excited and animated about our ideas and expressed them forcefully. Whenever we got together over the summer there was an initial period where there were a lot of issues that we had to get out on the table and work through.

HAIGH: Did you start off by drawing a list of modules and splitting them up between you?

DONGARRA: I can't remember exactly how that worked but it was probably similar to that approach. We each had our area that we were going to work on and extend. Everybody had their ideas of where they wanted to focus. Pete wanted to work on issues relating to linear algebra applied to statistical methods, including these squares problems and singular value compositions; Jim Bunch wanted to work on symmetric indefinite solvers; and Cleve wanted to work on dense problems. I was working with him on some of that, and also trying to develop a framework for the testing of all the software and how that would work in general.

HAIGH: With EISPACK you were able start off with a single source for the methods. Where were the methods in LINPACK coming from?

DONGARRA: Yes, that's an interesting aspect. So we never talked about the EISPACK issue and how that worked. So let me just back up a second and talk about EISPACK. With EISPACK we had our software, which was supposed to be portable, but, because it was going to run on different systems and those systems had different numerical accuracies, we actually had a master file of the software. The master file contained IF DEFs, but it had code in it which could be selected out if you wanted to gain a certain version of the code. So if you wanted the CDC version we would use a program called the selector and say CDC and it would draw out of this master file the code. The code was, for the most part, the same on every machine but it would also have certain statements which were specific to the CDC. By this I mean it would have statements about the machine's floating point precision, the floating point accuracy of an arithmetic that's critical for convergence of some methods. For CDC the floating point precision was different than IBM, and it was different from Univac. Each of those things had to be selected in a certain way. On IBM you wanted to get a double precision version, on CDC you wanted a single precision, so those had to be selected out. This program was a way to draw out the specifics for these "portable" programs.

With LINPACK we structured the codes a little bit differently. We wanted to have a code that "lived" in four different precisions: single precision, double precision, complex and double precision complex. So each routine in LINPACK has four flavors associated with it: single, double, complex and double complex. A user who had complex arithmetic could call on a routine and that would have a very similar calling sequence to the real counterpart, so if somebody knew how to use one they could easily use the other one. That wasn't true in EISPACK but it was true in LINPACK and it made things work very easily for that. We had ways of doing the extractions which were similar to what we had done before but more complicated now because of these four flavors. Ideally we wanted to have one version where it was possible to extract the single, double, complex and double complex versions, and that was an interesting aspect of the project.

HAIGH: This is Session 2 of the interview with Jack Dongarra conducted by Tom Haigh, this session is taking placed on the 27th of April, 2004, and it's taking place in Professor Dongarra's home, which is in Oak Ridge, Tennessee.

Now to continue where we had reached last session. You'd began to discuss your Ph.D. at the University of New Mexico, and the consulting work that you did for Los Alamos around the same time, and then you'd realized that you needed to back track into LINPACK.

DONGARRA: Yes.

HAIGH: So, I think you'd described the original motivation for LINPACK, you'd talked about the four collaborators and I think that was about where we'd stopped.

DONGARRA: Right, right. So, as I was mentioning the original idea for this project, I feel, comes from Jim Pool, who was the director of research for the applied math division at Argonne National Lab. At some point in 1974, he brought together a group of individuals who had an interest in linear algebra and software, and we held sort of a group or colloquium on the topic. I guess a consensus was reached that a package would be a good thing, and roughly in February of 1975 a group of participants met at Argonne to lay the ground works for a proposal. In August of 1975, NSF agreed to fund that project for three years, from 1976 onward. The Department of Energy also supported the effort, and the funds were used to conduct research into the development of mathematical software, the development of this package. The funds went to four institutes: Argonne, New Mexico, Maryland, and the University of California-San Diego. The people who received funding were Jim Bunch at California, Cleve Moler at New Mexico, Pete Stewart at Maryland, and I received funding at Argonne. That’s sort of the mechanics of the project, in terms of how things got going. We began the project and then we met at Argonne during the summer of 1976 to understand where we were going and how the software would fit and work together. In the fall and the winter we worked independently, more or less, and then met again at Argonne in the summer of 1977, trying to put together the software documentation and the test material. In the fall of 1977 that software was sent to our test sites, a group of friends who would exercise the codes for us, make the software available to their user community, and provide general feedback about how that software was working. Based on that testing, we made some changes that were incorporated back into the package in 1978, and by the end of 1978 the codes were ready to be released. The package was sent to both the Argonne Code Center, which had changed its name to the National Energy Software Center, I think that’s what it was called, and also to IMSL. IMSL was a mathematical software company that was leasing numerical libraries, and they had agreed to distribute our software for a modest fee, I believe it was about $75. For that price they would make the tape and send it to you.

HAIGH: So other than the involvement of IMSL was there anything else important different in the testing and distribution process from the one that had previously been used with EISPACK?

DONGARRA: Not really, I remember things pretty much following that same pattern or footprints that we had used with EISPACK. The software was put together on mag tapes and sent to our test sites, they ran our tests at a minimum, and hopefully they would make the software available to their user community and let their users then get a chance to see it. They would support the software locally, providing the help they could, and if users had problems they would then feed those back to us.

HAIGH: Yes. Now you had mentioned earlier that the sessions when the four of you met to hash things out could involve quite a lot of animated discussion and argument. Can you give any examples of the kinds of things that would be controversial or argued about?

DONGARRA: Well, they ranged from things that were critical to how the package looked to things as silly as what we called certain variables and how the layout was done for the documentation. So it was a whole range of software issues relating to details of the numerical aspects of the software, to stylistic issues which are very subjective and individual, to documentation layout and the level at which the documentation should specify issues about the algorithms. There were a whole range of issues that had to be sorted out, and I would say that Cleve and Pete were perhaps the most outspoken and opinionated of the group. They were the ones who became the most animated, and perhaps led to a number of interesting discussions about the style and the issues of form for the package. So I would say that those were very positive discussions, but at times there were very heated exchanges. You can imagine somebody working for six months off at their university, developing software of a certain style and form and then coming during the summer and realizing that decisions had just been made to basically change all the stuff they had worked on, so they had to go back and reengineer some of that and retool it to fit into the new style.

HAIGH: And were there any areas there that you think, in retrospect, exposed significant philosophical differences or were they all issues that seemed fairly import at the time but proved ultimately arbitrary?

DONGARRA: So there were many arbitrary issues that had to be settled and there were a number of issues, I would say, that were more substantive than choosing how to spell a certain variable that's going to be used. There were many issues about how to do the convergence, how to specify the data structures that were going to be used, how certain routines were to be used in terms of calculating certain quantities; all of those things, you know, were on the table and had to be addressed. We wanted to do it in a consistent way throughout the package, so it was important that we had consensus and that it be followed with actual implementation.

HAIGH: So, we should talk a little about the use of LINPACK. You've followed through to the testing process, and then you said, was it 1978 that it was released?

DONGARRA: 1978, it was finished, 1979 was probably the official release date.

HAIGH: Yes, 1979 is the official release date. So did it go to all the same kinds of installations that would already be using EISPACK, or was there something about the audience it appealed to that would be different?

DONGARRA: Since linear systems have perhaps a broader impact, LINPACK was going to a wider audience, and we felt that it would have a larger acceptance. This package was designed at a time when the biggest computers available were the vector computers. The vector supercomputers were just coming onto the scene, and the package was designed with vector computers in mind, so the package was designed to rely on an underlying set of routines called the BLAS (the Basic Linear Algebra Subprograms). The BLAS are a set of kernels which form the computational core of LINPACK; they are the vector operations that are going to be done over and over again in the package. The BLAS were a set of standard routines which were formed right before LINPACK was really kicked off, and we made a decision to use them.

HAIGH: So whose idea was it originally to develop the BLAS?

DONGARRA: I credit the BLAS to Chuck Lawson and Dick Hanson. They were both at JPL at the time, and were interested in mathematical software (their interests were in solving these squares problems). They understood enough about software to realize that we had to standardize and get reuse of software anyway we could, and that coming up with this core set of routines was going to be critical for the community. So they embarked on a path to standardize a set of routines, get the community involvement, and through consensus of the community build this set of routines. They had a series of informal gatherings that took place during conferences, and they would try to explain what they were doing and solicit input, looking for criticism and improvement to what they had put in place. They had a small set, maybe ten, fifteen routines which would be the core again for numerical linear algebra problems, dealing with vector operations, doing inner products, taking norms, doing a scalar multiples of vector added to another vector, doing things like summing the elements of a vector, finding the largest in magnitude, and the hope was that if we could isolate those critical parts and implement them very efficiently on each architecture then packages built on top of them would run very efficiently. These were vector operations and LINPACK was built on top of vector operations, so using them would enable us to achieve "portability" and performance at the same time. The hope was that they would be implemented efficiently on different architectures. Vector machines were just coming out, and vector operations were the core of the BLAS. The thought was that if LINPACK used the vector operations it would run efficiently, and that's exactly what we did.

HAIGH: So there would have been a different implementation of the BLAS that was tuned to the vector architecture?

DONGARRA: Correct. So each machine would have its own implementation of these kernel routines, perhaps produced by the vendors, that would be optimized for the architecture and they would be linked into LINPACK, and then LINPACK would work efficiently because they used these lower level routines.

HAIGH: Is this the work you previously alluded to doing while you were consulting for Los Alamos?

DONGARRA: Right. So at Los Alamos we were interested in trying to get efficiency and trying to write software so it could be efficient even in the high level language. At the time most users had to resort to something called CAL, Cray Assembly Language, and that CAL code was rather tedious to write, hard to debug and was only for that one architecture. We were thinking of trying to write something in FORTRAN that would capture most of the efficiency that we would see in CAL, and we had some modest success in doing that. Some of the success came about by taking loops and unrolling them to expose more operations and letting the compiler see the operations and have a hope of the compiler being more efficiently from an execution standpoint. Writing software that's efficient, you're trying express the code so the compiler does the right thing when optimizing. You don't write codes specifically for the machine you write it for the compiler, which then hopefully writes it for the machine. Getting the compiler to recognize what you're doing and express it in a way that is efficient on the architecture is a constant struggle, even today.

So, yes, there was a set of BLAS, they were written primarily to make the package run efficiently, and we were in a better position because of using this underlying set of routines. These BLAS routines, at the time, were just vector operations. We later came to understand that that wasn't enough,so probably 10 years later, we developed another package which is called the Level 2 and the Level 3 BLAS which have the same kind of ideas but are expressing higher kinds of operations. Instead of vector operations, they express matrix-vector operations, and matrix-matrix operations, and that's critical to gaining performance on machines that have a memory hierarchy architecture, that is they have main memory, they have Level 1 cache, Level 2 cache, vector registers, and so on. By using these higher level BLAS we're able to get more reuse of data in that memory hierarchy and allow the algorithms to run much more efficiently overall.

HAIGH: Yeah. So the dates you have here are 1987 for the Level 2, and 1989 for the Level 3 BLAS.

DONGARRA: That's right, yes.

So let me just back up and say at least one more thing about LINPACK; it's a historical point which I think deserves some attention. LINPACK was a software package and along with that software package we produced a users' guide. So we had a set of software and accompanying document which described how users should, how users could use the package effectively. It described a calling sequence, it described various examples and how a user could in fact use this software to solve these kinds of problems. That was a document that was produced and published by SIAM. The first edition came out in 1979. [*LINPACK User's Guide*, J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewart, SIAM Publications, Philadelphia, 1979].

In the appendix to that book I collected some timing information about various machines. The idea was to give users a handle on how much time it would take to solve their problem if they used our software. In the appendix was the table that listed the solution time for solving a matrix of order 100 using the LINPACK software, and that problem size was chosen because I was able to accommodate that matrix on all the machines that we were testing on, from a Cray down to a DEC PDP something or other. I was able to use that same matrix size, and we received the execution time and translated that into an execution rate. Based on that there was a little table published which said, "on this machine it took this long, this rate of execution, for this problem" and maybe ten or fifteen machines were listed there. So that's the origin of something called the LINPACK Benchmark. That LINPACK benchmark has a life of its own today, but it carries on from the origins in the appendix of the users' guide. Today there must be thousands of machines on that list, which provides the performance for those machines running that same problem, that matrix of order 100: solving that problem on their machine, going through the FORTRAN compiler and looking at the performance for it.

HAIGH: So other than its portability is there anything that makes LINPACK a particularly meaningful benchmark?

DONGARRA: LINPACK is a benchmark that people often cite because there's such a historical data base of information there, because it's fairly easy to run, it's fairly easy to understand, and it captures in some sense the best and worst of programming. LINPACK is not particularly well done for modern architectures, so it doesn't have a very good access pattern in terms of referencing data in memory, and as a result the performance is really quite inferior to what we can do today. That's one reason why we built other packages subsequent to LINPACK, and many applications are written the same way as LINPACK. So some people feel that LINPACK captures the essence of their application, not necessarily what it's doing but the performance level that it's achieving, so that's another reason why it perhaps is something that people resonate with when they look at performance.

HAIGH: Yes. I'll have to back up and ask you a couple of follow up questions on this. Was LINPACK the first package to have used the BLAS?

DONGARRA: Ah, that's interesting. So I'm not sure, it may have been. There may have been other uses of the BLAS before LINPACK but it certainly is the first "major" package that embraced the BLAS and used it. There was help on both sides because we used the BLAS, so the BLAS got more exposure, and people understood what it was and it helped the BLAS get going; on the other side, since the BLAS was getting more exposure, and because it was used in LINPACK, people looked at LINPACK as a good thing .

HAIGH: Yes. So LINPACK would have played an important part in persuading people to tune and optimize BLAS for their machines?

DONGARRA: Absolutely. Once it received wider recognition, or wider acceptance, then people were keener to optimize the kernels of the routines, so it certainly had a very positive effect in terms of trying to produce this portable efficient set of underlying routines.

HAIGH: And leaving on from that. Your work at Los Alamos was that literally that you were working there with their Cray to make the BLAS efficient on it, or were you just doing similar kinds of work?

DONGARRA: So, I don't remember if I was given a specific charge, or if I was working on a specific application. I was there trying to understand better how to use that machine and one of the ways that I was looking was in respect to our package, LINPACK, and also the BLAS in trying to get things to run efficiently on that machine.

HAIGH: Was adapting software for the vector architecture akin to the work that had already been done on double and single precisions? Or did it require some more fundamental change in the actual mathematical methods being used?

DONGARRA: Well, it required a fundamental change in the way in which we construct the algorithms. The mathematics has to be translated into an algorithm, and we need to implement that algorithm in a computer language in a way that captures the numerical form but also expresses things in a way which is efficient on the underlying architecture. Then the computer language is converted by a compiler into some assembly language of the computer and finally executed on the hardware. Getting each step right is a complicated process, and getting it right so it can be portable and efficient, across different machines, is a very critical thing. With LINPACK the resulting programs were not very efficiently. That is we cannot write the algorithms very efficiently if we express things only at the vector level: it doesn't expose enough of the operations so we can effectively use the memory hierarchy of modern machines or let us capture reuse of data in a way that can get to the higher level of execution that these machines are capable. That doesn't relate just to vector computers, it relates to all computers which have a memory hierarchy. With vector machines:, there was a big gap in performance between effectively using this machine in vector mode and running a scalar speed, so there's a large gap between getting high performance by effectively using the memory hierarchy and not. With some of the packages like LINPACK we see the software is not getting very high efficiency as a result of poor structure of memory reuse of our algorithms. And it's not just about vector machines, it pertains really to the memory hierarchy on the vector machines and that's something which is true on all of our machines today.

HAIGH: So those same issues that first became really pressing with the Cray's would today apply to a humble Pentium class chip?

DONGARRA: That's exactly right. The chips in our laptops are capable of high performance;for instance, if we do the right thing with respect to the memory hierarchy we can see really significant speedup over doing it in a naive way.

HAIGH: And the last question I have on that area would be, it appears from what you've said that as well as portability one of the big contributions of the earlier EISPACK project had been to get these newer and better methods into wide use. Now with LINPACK you've already talked about portability, about this modularization of the core functions into BLAS to allow tuning, and about the documentation. Was there anything new in the actual mathematical methods in the package, perhaps some methods there that might not already have reached a broader audience?

DONGARRA: As the state of the art progressed, newer ideas come into play and those were embodied in LINPACK as well. Methods relating to the singular value decomposition were improved. LINPACK deals with solving systems of equations and it deals mainly with dense problems and those are handled for the most part with direct methods. But LINPACK also has a component which deals with finding the singular values of a matrix and that algorithm is iterative in nature, and some of the ideas related to improving the convergence for the iterative methods were incorporated into the LINPACK structure.

In terms of the direct methods, many of the ideas related to performance and how we could structure things for portability and that was critical for some of those methods. Also how the data was laid out in memory was critical. We do compressed storage in the sense that we have a symmetric matrix, and we want to represent just half of the matrix and not waste the other half of the array. So we examined how the algorithm can be written to effectively use that compressed storage. There is generally a two dimensional array containing the matrix, so we take the data and compress it into a linear array and now we have to write an algorithm to effectively access that linear array and carry out the computations in a very efficient manner.

One of the other things about LINPACK was the fact that each routine existed in four flavors: single precision, double precision, complex, and double precision complex, and that was the first time that a package of this size really had that level of specification. We see that also in the BLAS, as the BLAS have single, double, complex, and double complex as well, and that sort of led our decision to go in that direction. At a software engineering level we produced one version and held that in some, what I would call, abstract form and through a software tool called TAMPR, we were able to extract from the abstract form each of these four precisions automatically. So in essence we would write one version in a way that was general enough, and we could capture the abstract form and then be able to generate from that form the complex version or the single precision or the double, or the double complex version from that one instance.

HAIGH: Can you remember who it was who produced it?

DONGARRA: Jim Boyle at Argonne National Lab was the author of the TAMPR package, and let's see, what does TAMPR stand for, something I'm sure with transformational in it somewhere. [Haigh: Transformation-Assisted Multiple Program Realization system]

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

HAIGH: That's covered all the issues I had in mind about the developments of LINPACK. I do have a question about the use of it, again as a precursor for today's open source software. Was the pattern of use pretty much the same as EISPACK in terms of the level of feedback you'd get from the users?

DONGARRA: Yes. Users would install this package on their machine, they would make the documentation available. Everything was free, everything was open source, there was no licensing that we even considered with the software. There's no copyright statement in the software, there's no issue about free use or anything like that. It was just available, and if ever the question came up we would tell people that it was developed using government funding, it was freely available and could be incorporated widely with their software. So the software was made freely available and people use it, and they would then send us questions and comments mainly through the form of regular mail, is the way I'm remembering that. Now this was before we really had a network so we didn't have e-mail in its form that we know it today. So back in 1979, 1980, the question is how did we receive feedback for this software collection? People sent letters or telephoned us to say things were working or not working. We would hear comments that the software was running much better than what they had been using before, or that this software covered an area which they were interested in, or why don't you include this kind of routine in the package next time. Then there would always be comments that the software doesn't work, and we'd have to sort out if the guy was just misusing it or if there was a problem in fact with the software. Early software of course had problems and we made changes and fixes and bugs got corrected along the way. We tried to have an extensive test collection, and that test collection exists today and people still use it to check their software that they're writing which would improve on certain aspects of the package in a way to verify in fact that their stuff is correct.

HAIGH: Were there any cases of people other than the four original authors using the package and wanting to contribute new or improved routines?

DONGARRA: Oh, yes, yes. We're the people who designed the original package but I'll say that we had a lot of feedback and input from the community on these things. This was not four people sitting in a closet designing it totally in isolation. We did have feedback, we talked about it whenever we could, and after the package was exposed to a much broader community we did receive feedback and suggestions about how to extend the package and make things better. Some of those suggestions were filtered into the code over time as we released bug fixes and so on, but I don't remember major changes, major extensions, being made to the package after it was finished.

One of the problems that we have with this kind of software is that we receive some research funding, and we engage on a project to develop software. (The proposal that's submitted usually doesn't say we're going to develop software, it says we're going to do research in this area here, developing new numerical methods and understanding how we can improve the state of the art of these other things, but we develop software in the end). The problem is that the grant goes on for a certain amount of time, usually three years, and after that time we're finished, and then we move on to something else. However, the software doesn't stop; that software carries on for an extended life. We have software from these packages now which is close to 30 years old, and that software's still being used today, and we don't have support to maintain the software in the sense of providing answers to questions, providing improvements in the documentation, providing fixes for bugs that come up, providing additional capability and minor extensions to those packages. So there's no mechanism in place to help users beyond the existing grant and the grant is usually not written to develop software, it's usually written to do something else and the software becomes a byproduct of that. In some sense it's a legacy that we have and I occasionally get e-mails saying, "This piece of software doesn't work and it has your name on it." Of course I'm interested in understanding why it doesn't work and I want to fix it if there is something broken, although it's pretty unusual that there's something wrong with software that's been in use now for about 30 years. In many cases it's the user who has not read the documentation correctly and has some problem with understanding the interface to it, so perhaps we can do a better job of writing the documentation . The main problem is that there's no support, there's no funding, there's no mechanism in place with these legacy packages.

HAIGH: That's a very interesting point. So if you wanted to make substantial improvements then you'd need to get another grant, and to get that grant you'd need to promise to do something newer and more exciting than just LINPACK 1.5?

DONGARRA: Right. So you couldn't get another grant to do software maintenance, because NSF or DOE doesn't fund software maintenance, they fund research. People are not interested in funding that kind of effort, and to be honest it's not the most exciting kind of work either, so to find the right people to engage in an effort of maintaining software over some life time is a hard task, I would think.

HAIGH: Now in some ways it seems like that's the situation which people have claimed that open source software has an advantage, because once you have a core of a package there then users will go away and allegedly produce their own improved versions and the central coordinators can just sit back and patch new code in occasionally. Do you think something could have worked for LINPACK at that point in history if the idea had been tried?

DONGARRA: Right. So there's no central coordination as there is with the "open source" movement today for our numerical software, there's no centralized repositor of the software which has a mandate to look after the software, and to take input from the community and feed that back into the software the way the open source has done it. I think it's a fine way of doing business: it provides a mechanism for many hands to improve the software, and a very well understood path for incorporating those changes back into the collection. So that's a wonderful thing, we don't have it today with our numerical software, we rely on a number of other mechanisms which are not very rigorous in that same way, and are prone to the whims of people, basically, on whether or not something gets implemented.

HAIGH: And do you thing there's anything fundamentally different between mathematical software and, for example, operating systems or compilers that make one approach more sensible for one area, and the other for the other?

DONGARRA: I don't see it, not at the 10,000 foot view. I think they both can be done through the same mechanism. Obviously the people have to have different qualifications, for numerical software one has to understand the numerics and understand what the implications are of making subtle changes to the structure and the conditions that exist in the code in order to do the convergence. So there's a slightly different, a basic understanding is different, but beyond that I don't think so.

HAIGH: Well I had formed the impression to some extent from what you were saying, that it might be more possible to say that routines in something like LINPACK are definitively finished and debugged than it could be in something like an operating system where there would be constant need to rewrite things for new hardware and add new features and to change things around.

DONGARRA: So LINPACK is perhaps a bad example to look at. But if we take a look at a more complicated package I think we start to see the similarities, and LAPACK is a more complicated package. It's more complicated because its level of modularity is quite a bit more extreme, it has more involvement with the performance which changes from machine to machine and from architecture to architecture, from time to time. It has a number of other characteristics which try to adapt themselves to the underlying structure, so I would say it's a more complicated situation there. And as we go up in the chain there's another package called ScaLAPACK, which is one used for parallel machines, distributive memory parallel machines, and there it's a greater level of potential problems and optimization and improvements could be made to the software.

HAIGH: Alright. So I'll talk to you later in more detail about those packages. But the lesson we take from this part then is that as the numerical packages get more complicated, then the issues faced become very much closer to those on operating system type projects.

DONGARRA: I would agree.

HAIGH: Okay. Now returning to your own career. In parallel with LINPACK you had been working with Cleve Moler at the University of New Mexico on your Ph.D.?

DONGARRA: Yes. I spent what amounts to one semester at Los Alamos, about 16, 20 weeks, working there with people about half a year, and then started the real academic side of my Ph.D. at the University of New Mexico in Albuquerque. I was resident on campus for two semesters, so again my trajectory here is not a common one I would say. The way this worked was that I went away to Los Alamos, and spent basically one semester's worth of time taking two courses in evening school there, and the next semester I then went down to campus and went to work taking my courses and preparing for the qualifying exam. You had to take a certain number of courses, and a very small number it was, and then take some set of exams to qualify for the Ph.D.. You then had to do a dissertation and defend that and do some other things. I remember having to take courses so that I could be examined in three areas. I don't remember the areas, but they were probably something like function analysis, numerical linear algebra and maybe differential equations. So I took courses that would prepare me for those exams and, I also had to take a foreign language exam. I had Russian as an undergraduate so I decided to take Russian as my exam. So those were the four hurdles I had to overcome in order to qualify to do the dissertation, and my schedule said I should do this in one year.

I was taking courses and doing all these things in preparation for finishing in a year, and the plan would be to work with Cleve on the dissertation and do that for the third semester, and after that I'd go back to Argonne and finish up my dissertation. So the other part of the story is Cleve was on campus in New Mexico teaching for the first semester I was there, and the second semester while I was on campus he went to Stanford for his sabbatical. He said “if you pass your qualifying exams at the end of this semester you can come and join me at Stanford and we’ll get you started working on the dissertation.” I said that was great. I had a path which said I had to finish in this time, and I took courses and took the exams, and the following semester went out to Stanford. Cleve was there for the one year period. One semester had already gone by and I stepped in at that point during his second semester, were he was working with the computer science department - working with Gene Golub and the other students and people there. And that was a wonderful experience. I feel very fortunate for that opportunity as it allowed me to intersect with a number of people who are still friends and colleagues today.

HAIGH: And your resume shows that as being visiting scholar at Stanford University in 1979.

DONGARRA: 1979, that’s right. It was right after LINPACK was released. We were finishing up that and thinking about new things. One of the students there at Stanford was Eric Grosse. Eric and I talked about a number of things and he subsequently got his Ph.D. and went off to work at Bell Labs. And one of the things that Eric and I eventually decided to put together was this thing called NETLIB. NETLIB is this repository of software, we did that in… I want to say 1984, but it may have been a little bit earlier than that. My point here is that a number of interactions were begun at Stanford that continue to today. So I was at Stanford for that one semester, finishing up Cleve’s sabbatical there, and then I went back to Argonne. I went back to Argonne as a staff member, and was still working on my Ph.D.. The agreement I had with the Argonne was I would get half of my time to work on my dissertation and the other half of the time to work on lab related stuff. And I must have finished probably after nine months or a year on my dissertation, and then got my Ph.D. in 1980.

HAIGH: Yes. And what was your dissertation?

DONGARRA: So my dissertation was on improving the accuracy of eignenvalue calculations. The idea is you can go off and compute an eigenvalue and use this technique to enhance the accuracy of that eigenvalue calculation for little cost. So basically the eigenvalue calculation overall is an n cubed process and the improvement and the accuracy for each eigenvalue is of order n squared. So you can think about doing the eigenvalue calculation overall in a lower precision and then going back and selectively compute the eigenvalues that you want in a higher precision.

The topic was suggested by Jim Wilkinson who was this leading figure in numerical linear algebra, while we were at Argonne. Jim had given a talk about this idea and I had been floundering around looking for a topic. Cleve and I had talked about a number of things and they weren’t panning out. After this lecture by Jim, Cleve pulled me aside and we talked to Jim a little bit, and Jim thought that would be a good topic. Jim said he wasn’t interested in pursuing it himself, but he thought I could do a fine job looking after it. So I was let loose on that and talked with Jim Wilkinson and Cleve Moler, and came up with a strategy and approach on the implementation, and some theory. I was blessed then with a Ph.D. in 1980, and had a publication with Jim and Cleve on that topic. The paper appeared somewhere.[*Improving the Accuracy of Computed Eigenvalues and Eigenvectors*, J. J. Dongarra, C. B. Moler and J. H. Wilkinson, SIAM Journal on Numerical Analysis 20(1):23-45, February 1983].

HAIGH: Did that method find its way into any of the packages later on?

DONGARRA: So it did actually, not into a package, but it found its way into this collection that we have based on the ACM TOMS Journal. There's a set of algorithms and software that TOMS maintains, and that's one of the algorithms that's there. I don't remember which one but there must be a publication that points to it. I'm not even sure how many people have downloaded it, but that would be an interesting thing to find out.

HAIGH: I have the first publication here which I think you alluded to a second earlier, "Improving the Accuracy of Eigenvalues and Eigenvectors", J.J. Dongarra, C.B. Moler, and J.H. Wilkinson, SIAM Journal on Numerical Analysis, Volume 20, Issue 1, Pages 23–45, February 1983. And there's actually a couple of other publications in 1984 also with Wilkinson.

DONGARRA: Yes, a very interesting time.

HAIGH: You'd done all this work in EISPACK and LINPACK. Wasn't there anything you'd already done, that you could just take, write up, and call it a thesis?

DONGARRA: Well, yeah. Why was that? It was because there were contributions made by too many people. A lot of people had their hands in the software, and I was in the math department, I wasn't in a computer science department or a software or computer engineering department. I was in a math department and my degree was going to be in applied mathematics, so software is one of those things for which people don't perhaps have enough respect . There's an issue about the math department doing stuff on software, which back in the 80's was not given a high level of respect, so it was easier to turn to the numerical methods and to come up with something fresh. It was fun doing that, and it was a good thing that I had done it rather than to just put a stamp on something that we had all had a hand in doing, so I don't feel bad about that at all.

HAIGH: Now your resume shows that in 1980 you received your Ph.D. and it also shows that you were promoted to a senior computer scientist at Argonne in 1988, where you remained until 1989. So how did your position at Argonne change, and what kind of projects did you work on after your return there?

DONGARRA: How did my job change when I became a Ph.D.? It didn't which is sort of an interesting thing. I wasn't treated any differently, I didn't change the uniform I wore, I didn't get a better parking spot or anything. I was just another guy, and continued to be another guy there. It's not exactly true because I certainly became more mature and understood how things worked at a different level, so I'm sure people treated me differently. From my first time there as an undergraduate until I left it was a very, very positive experience and one in which we all worked together and did things in a very common way. The work environment was terrific, very rich and stimulating for a young person, and it led to association with many individuals who came through the Lab It never felt like I had to go to work, it felt like I was going and experiencing something new almost every time I would go into the place. It was almost like being at Christmas all the time, opening things up and finding new exciting things there, so there are a lot of very good memories about Argonne.

HAIGH: And was it also similar to the academic environment in that you would mostly just sit by yourself and work on the things that you were interested in?

DONGARRA: I certainly had the options of doing that. I could also interact with people at the Lab, or with people outside the Lab, and I would say I did all those things. I engaged in activities with my colleagues around the world, I engaged with my friends at the Lab, and I also worked by myself sometimes.

HAIGH: Were there people akin to research assistants who you could order to help you with things, as opposed to colleagues that you collaborated with?

DONGARRA: "Ordered" is the wrong word. We had people who would help with our projects, clearly, and we had students who would come by and interact with us at various times as well. We would get them to work with us and do some of the work that perhaps was more tedious to do and more experimental in nature. I wouldn't say we ordered anybody around to do those kinds of things. And I've certainly collaborated with many colleagues there at the lab in the division.

HAIGH: Okay. So you can talk about some of the projects that you worked on there after 1980, and the release of LINPACK?

DONGARRA: After 1980, there was a sequence of projects that took place. There was NETLIB, a repository of software that Eric Gross and I put together. Gene Golub suggested it would be a good idea to have some place where we could electronically store programs that were being developed by the community. Gene is a professor at Stanford, he had a number of students who produced software, and after the student left and found a job, they weren't interested in maintaining that software. They went off and did something else, perhaps totally different from what their dissertation topic was, and as a result the software would fall off the road. Gene felt it would be a good to reposit the software and let people get access to these great ideas over time, that it would be beneficial for everybody. So Eric and I worked on this system called NETLIB, which was by today's standards a rather crude mechanism. Understand there was no web in place at that time, there was no network to speak of. We had electronic mail, so we decided to use electronic mail as a mechanism for exchanging programs. We set up a repository at Argonne and one at Bell Labs. An individual would send electronic mail to that repository, and software would intercept the mail, try to parse the mail and find out the request, and then send back to the sender the requested piece of software. That worked very well, and in fact it’s still in use today. There’s also another mechanism of course which is web based and makes it a lot easier to point and click.

HAIGH: Can you remember when Argonne first got e-mail?

DONGARRA: You know I can’t. I struggle to remember when I sent my first e-mail message. My whole time at Argonne I was sitting in front of a console, typing things in. I never used punch cards, but the first time that e-mail was sent I don’t remember. Initially it was sent internally, so we were probably doing that in the early 80’s I’m sure, sending electronic mail, even in the 70’s… but I don’t remember exactly when mail going outside was first done.

HAIGH: And can you remember when work on NETLIB began?

DONGARRA: When it began, yes. This is a paper which describes NETLIB, a CACM publication, this is on my web site. So I scanned in the paper and have that on my web, and it must say there someplace where we first started doing this. So the date of the paper is 1987 –

HAIGH: Yes. And there’s also NSF agreement number on that that I can just about read –

DONGARRA: Yes. So if we go to the end there what does it say?

HAIGH: It was received in 1986, I think, there’s some acknowledgements –

DONGARRA: Yes. So I have another paper here which talks about things on a more informal level, sort of an advertisement for NETLIB.

HAIGH: Well you do have a graph showing request service and the graph begins in September 1985.

DONGARRA: There you go. So that’s probably when we opened shop, 1985, How many requests were there, very small number…. under 500 for that month.

HAIGH: And then it seems by about January of 1986 it was up to pretty much the general level of use that would achieve over the next couple of years.

DONGARRA: Yes.

HAIGH: So that’s quite a rapid adoption.

DONGARRA: Rapid adoption by our community, and these are requests for individual routines from a package for the most part. These are real hits; they’re not like web page clicks that you’re getting all sort of GIFs and other things. These were actual downloads of individual items of software, and done electronically.

HAIGH: For the transcript I should say this article is called "Distribution of Mathematical Software by Electronic Mail", it's by Jack J. Dongarra and Eric Grosse, and it's published in May 1987, Communications of the ACM, which is Volume 30, Number 5.

DONGARRA: So that was one publication. There's another one here but this has less information, this comes in February of 1989. [*Shopping for Mathematical Software Electronicall*y, J. Dongarra and E. Grosse, IEEE Potentials 8(1):37-38, February 1989, ISSN 0278-6648]. This is some IEEE publication, but it doesn't have much real content. The CACM is the real one.

HAIGH: What was the Bell Labs involvement with the project?

DONGARRA: Eric worked at Bell Labs. We decided to do this thing and he decided to set up shop there. Bell Labs also had this history of mathematical software going back to their PORT library, so they were certainly savvy in terms of mathematical software and trying to promote the use of mathematical software as much as possible. I think Eric was given a free hand to do whatever he needed to do. There was no money to be made off these things, and again we didn't have any issues about copyright or selling. I never asked anybody at Argonne if I could do this, we just set a machine up, put it in the corner and let it go receiving e-mail and responding to mail as it came in. There was no official sanction given by the Lab. It was one of those things where if I had asked the question I'm sure I would be told I can't do it, so rather than ask, I just did it, and nobody ever objected.

HAIGH: And another, probably the earliest, historical precursor would be the SHARE library.

DONGARRA: SHARE library, yes.

HAIGH: Now was there any kind of quality control of the contributed routines?

DONGARRA: Absolutely, yes. So Eric and I were the gate keepers. We were the ones who said what got put into this repository. People would ask, "could I put stuff in?" We'd look at the software and say yes or no depending on what we felt was the quality and the overlap and use of the software itself. So we did turn people down. This paper talks about putting in place a set of editors who would review software and who would be the gate keepers for certain areas. And I think we actually implemented that at one point, but it wasn't done to the same extent that we would referee papers in journals, and it wasn't as persistent and as openly known that that's how it worked.

HAIGH: So essentially you and Eric were acting as editors, deciding which things to include. Now you mentioned looking at the software, you also mentioned the question of overlap, so what kind of criteria would you use in looking at the software itself to decide whether it was worthy of inclusion?

DONGARRA: Well we were looking to see if it was addressing an interesting area, if the software worked, did it compile, was it portable, was it addressing an interesting problem area, how much overlap did it have with software that was already being collected there, and was it an improvement at all on the software that was there. We also considered what we felt we could expect in terms of the author of the software supporting it and answering questions and being able to live up to some expectation at least from users. Based on all that, we would allow the collection to expand to include the software or not. This paper contains a list of that early software, so here’s a list of packages and directories that were done in 1985, and if we look today we would see a much expanded list. So that would be one way to see what was enhanced over that original package.

HAIGH: That’s Table 1 on Page 406. So if someone contributes something to NETLIB today do you still look at things that come in?

DONGARRA: Yes. So Eric and I look at the material. He’s still at Bell Labs or whatever they’re called now. We don’t get many requests and that’s okay. Sometimes we seek things so the nature of how we do business has changed. We don’t have a central repository. That idea was useful in 1985, but today we have the web, so people set up their own shops. I maintain a list which is sort of in the same spirit for linear algebra. I have a survey of freely available software which I maintain, and that is a list of software that people can download. It should be open source software, they should be able to get the source code for the software, and they should be able to incorporate it into their own package. I’m not very sticky about the licensing agreement that the package has, but, the software’s there and I try to categorize software in a number of ways to make it easy for people to find it. So that web site has had a lot of interest, and it sort of, in some sense taking freely available software for linear algebra and trying to have a focus for it within the community.

HAIGH: So, that in some ways would serve the same function. The difference would just be that the software isn’t copied locally because whoever wrote it will have it on a web site already.

DONGARRA: Right. So there’s no vetting of quality there other than somebody saying I have this package and want to make it available to the community and could you help get exposure to that software. Their name and e-mail address goes on the line so they’re not going to call me; they’re going to call that individual about it. Periodically I check to see if that software is still there. If bit rot occurs and we remove the link, we try to figure out what happened to it, and try to keep everything fresh; we put new ideas into it as well, so every year I send out a mailing and ask if there are any additions or corrections to it and I occasionally get things back.

HAIGH: Can you remember when you started maintaining that list?

DONGARRA: That’s a very interesting question. So I’m going to say about six or seven years ago, that’s when I started maintaining that list.

HAIGH: So that would be circa 1997.

DONGARRA: 1997. I was getting questions from the community saying “do you have this piece of software” and I got tired of answering that question. So I set up this web site which has basically the answers to the questions that people ask most commonly, and put the pointers there and exposed it to people.

HAIGH: So at that point it happened because you’d already been personally recognized as a gateway to software resources?

DONGARRA: That’s right, yes. For better or worse.

HAIGH: Now returning to NETLIB, I wonder if you could compare and contrast this and other kinds of mechanisms that were out there already for software, and as well as the early SHARE library and the Bell Labs project. One thing that comes to mind is the ACM algorithm series and then John Rice’s ACM Transactions on Mathematical Software, and another thing comes to mind is the Argonne Code Center, which you said had been renamed something by that point, so clearly there were already mechanisms there by which software was distributed, there were already mechanisms there by which software was, in the case of transactions, more formally peer reviewed and by that point there were also the IMSL and NAG libraries which you could buy commercially. So what would be the similarities and differences you think of NETLIB and these other mechanisms for obtaining software?

DONGARRA: Right. So the primary difference comes about, in my mind, with support. NETLIB is there and the software’s available and whoever downloads it has to realize that the author of that software may or may not respond to their questions. With NAG and IMSL if I purchase their library I’m guaranteed that if I have a problem somebody will address that issue. That is, I can go back and say “I have a problem with this routine, it’s not working right, I think there’s a problem you should respond to it.” And I paid them some money and I’m getting support in return for that, as well as getting documentation which is maintained and kept up to date. I’m assured that if I buy a machine next week, and that machine is from a popular vendor, that software will eventually be available on that machine and run correctly. The software that we have here, maybe it will, maybe it won’t, maybe the author’s available, maybe not, maybe somebody’s around to answer, maybe not, if I have a bug I’m on my own. I do have a source code however, so I can see the insides. With NAG and IMSL it’s a lot harder to get access to the source code. Sometimes you can but they’re not going to make it available to everybody. So that’s some of the main issues or differences.

With respect to the Argonne Code repository, I think there were restrictions on who could access the code, so I would say that even in those days there were issues about requests outside the U.S. There were issues about paying money as well, so originally is was free and then they started charging for their services, and I’m not sure if it was cost recovery, but whenever you start charging for stuff you have an uphill battle. This is especially with software, especially if that software can be found some other way, maybe not as good, but by some other mechanism where it's freely available. NETLIB is always freely available; there's no issue at all with charging. We didn't keep records on who individually got items of software, we didn't expose that list to anybody else, and it was used and embraced by the community, at least by our community at that point.

HAIGH: Yes. And how about the alternative to contributing something to NETLIB, of publishing a routine in something like *ACM Transactions on Mathematical Software*? Perhaps some would do both?

DONGARRA: When we first began with NETLIB we did not have TOMS in the collection, but we did later, so today TOMS is part of it. So actually when people publish in TOMS that software finds its way into NETLIB. We have an agreement with them to help promote the software in TOMS, and I would say that that was an uphill battle to get to that point. Every time a person retrieves something from TOMS there would be a cost associated and there was concern that the ACM would be losing money over this, but in the end enough people in our community helped to overcome that issue of money. I think John Rice was instrumental in that. John was the person who originated the journal and basically was the founder of this idea of mathematical software.

HAIGH: I've been somewhat surprised talking to people by the extent to which the same code can find its way into a bunch of different places, and sometimes into commercial products. Do you think any of these same routines that were contributed to the NETLIB would also have found their way into the IMSL and NAG libraries?

DONGARRA: I'm sure of it, yeah. In some case they helped, they helped that code. So a good example of that is LAPACK. LAPACK is this package which takes the ideas from LINPACK and EISPACK, updates them to present a package which runs efficiently and is portable and a number of other things. That package was built by about eight people, and two of the people come from NAG; Jeremy DuCroz and Sven Hammarling. They're NAG employees, and they worked with us, not just occasionally, but as collaborators; they were part of it, they put their ideas into it. After that package was finished, and this was not a surprise, they took the software and embedded it into their library, feeling it was of a higher quality than what they had initially.

HAIGH: Alright. So NETLIB has no kind of copyright at all, you can take, retrieve something from it, build it into commercial project, change it?

DONGARRA: That's the way we would like to have it. Now sometimes somebody puts something into NETLIB which has a clause in it which says for researchers only, and people are on their honor to adhere to that, and not to embed it into software which is going to be commercialized, but there's no rigorous check on any of this. There's no shrink wrap click through kind of thing that you have to agree on.

HAIGH: Okay. So the author could impose conditions but NETLIB itself would not do anything.

DONGARRA: Correct.

[End of Tape 2, Side B] [Start of Tape 3, Side A]

HAIGH: Jack Dongarra interviewed by Tom Haigh. This is the beginning of Session 3, it's the 28th of April 2004. This session is taking place in Professor Dongarra's office on the campus of the University of Tennessee at Knoxville.

Now having had a chance yesterday to consult the book, *Sources and Developmental Mathematical Software*, there were a couple of small other questions about EISPACK that was suggested by it. One of those was that in Chapter 4, in which you discussed the development in general of function and organization of EISPACK, and you described three different versions of the package having been released over time –

DONGARRA: Yes, that's correct.

HAIGH: With Version 2 as a major upgrade, Version 3 primarily to increase portability?

DONGARRA: Yes.

HAIGH: Did these take part very close to the original release, and were they funded as part of the same project or with separate additional funding?

DONGARRA: They took part at different time periods; that is they were separated by a number of years. The exact timing escapes me at the moment, but I would say it's more like three years apart, so the first release occurred, then in three years the second release, and then three years later, we made some enhancements to correct some things, and as you pointed out, to enhance the portability and the performance of the package itself. They were funded by the same effort. The EISPACK project was funded in large part by the Department of Energy and that funding was used for this package.

HAIGH: And the same people were involved in all three releases?

DONGARRA: Not quite. So the later releases, the consistent authors were Burt Garbow, Cleve Moler and myself. I think Jim Boyle was also involved in the second and third releases. In the original one it was Brian Smith who was the lead on that project and there were also a number of other authors on that.

HAIGH: And on Page 86, the article concludes with the observation, "We believe it's time for a major new edition of EISPACK, a complete revision should have its expanded capabilities, new user interfaces and uniform naming conventions. Some algorithms should also be altered to take advantage of vector machine architectures and paging operating systems. The program should be written in FORTRAN 77 with a mind to future versions of FORTRAN, such a project would require careful planning and extensive resources."

DONGARRA: Right. So that was a statement that was made in that paper. That was followed by discussions which led to a draft project, called EISPACK 8X. EISPACK 8X was intended to do just that. It was intended to update EISPACK, have a uniform naming convention, embrace FORTRAN's new standard, and follow more or less the same structure in terms of the problem area that was addressed within EISPACK. The people discussing the project were Cleve Moler, the two people from NAG, Sven Hammarling and Jermey Du Cruz and myself, but as it turns out that project never really came together for a number of reasons, and was put on the shelf. Later a charge was taken to develop a package which had, in some sense, a broader mission to look at both linear systems and eigenvalue problems, and that was the beginnings of a project which became known as LAPACK.

HAIGH: Okay, so we'll pick that story up later. Now I also noticed that in W.J. Cody's article about FUNPACK a package of special function routines, he talks a little about the EISPACK experience and he says whereas it had originally been hoped that the test sites would contribute a lot of insights and suggestions, that in fact early experience of EISPACK however dashed these hopes. On page 61 of that article he writes that the volume of material to be tested overwhelmed most sites, personnel assigned to testing often lacked the numerical training necessary to produce independent tests or were busy with other responsibilities, in a few cases field testing was carried out as hoped though most sites' responses were simply reports of the program's compilation.

DONGARRA: Right. There are a number of reasons for that. Jim points to some of them in that article. We sent them an enormous amount of material that had to be run and what we asked them to do was to compile it and run it, that was the minimum, and most of them did that. Most of them compiled it, reported back that there were or weren't any compiler problems, and then ran our test cases. They were then asked to install it on their system and then make that software available to their user community. Some of them did that, some of them didn't. Of the ones who did, some actually reported experiences with the code by users on real problems and gave us that feedback, but as Jim points out, the number of actual instances of that happening was really quite small. Part of the reason is that they were overwhelmed with material, they were volunteering their efforts, they did have other responsibilities, and in many cases the people were not highly trained numerical analysts, they were people running the computing center and had responsibility for insuring software support at that center. So they really didn't have the expertise to dive into the codes and fully embrace them, and understand the capabilities and limitations that those codes might have over existing routines.

HAIGH: Were there any sites that you remember as being exceptionally helpful?

DONGARRA: I can't, it's been so long that I can't really recall instances or individuals. Michigan-Ann Arbor, perhaps, is one that comes to mind where they were particularly helpful in terms of providing feedback. They had a long tradition there of helping and providing insight into some of the codes. And maybe also the people at Waterloo. None of the other ones really stick out in mind as doing everything that we asked and more.

HAIGH: We've already discussed LINPACK, but wrapping that up, my final question, which we didn't quite get to, is: as your work at Argonne continued through the 80's, was there any point at which anyone in the lab had the idea that they should be doing something more to safeguard intellectual property, or that there were things being produced that should be commercialized?

DONGARRA: Not at the lab per se. We tried to keep a low profile in the sense that we were concerned that somebody might feel we were infringing on the commercial side. Occasionally questions were raised about our software, including claims that our software was being produced by the Government, it was freely available, and it was competing with private industry. In particular, I remember some discussions of IMSL where they felt threatened by our software, in the sense that they may be losing customers to this freely available software package and questions were raised: why was the Government effectively competing with private industry. In defense we said that we weren't providing the same level of service that a commercial company was providing; that there was no support guaranteed. While we were producing software, IMSL was benefiting from that software in the sense that they were taking it and using it within their library, adding value to it, and providing documentation and support and services along with it. Most users who purchased their library understood that and were thankful that somebody was doing that for the long term.

HAIGH: So those were questions that were literally asked of the LINPACK team by members of the lab's management?

DONGARRA: I wouldn't say lab's management, but I would say that were asked. Maybe the lab's management, indirectly. I think I remember an instance where IMSL, or the people who funded IMSL, had raised objection with, I'll say, some congressmen or senators, and those senators then asked the Department of Energy why there was government competition with these private companies, and we had to respond to it in some way. So I remember drafting material for the response that came, I presume, from some administrator within our lab.

HAIGH: Then you sent the response and didn't hear anything more about it?

DONGARRA: I think our response was enough that everybody felt satisfied that in fact the company was adding value and that people should recognize the value of that and that there was a certain service being provided by the government in the sense that a basis for a commercial company was being laid, and that software was fairly high quality, perhaps a higher quality than had been seen by other government efforts, but that that software was then in a form which could easily be used and adapted within a company. So that happened within IMSL, that is, they benefited from this but also within other companies such as NAG.

HAIGH: You have said also previously that your role, and your kind of day to day existence within the Lab didn't change a great deal from the beginning, but certainly it must have when you returned full time in 1980, through to the end of your time there in

1989. I wonder if you could talk about any additional projects or initiatives that you were involved with there after LINPACK?

DONGARRA: Right. So after LINPACK there were minor projects that we worked on. I struggle now to remember all of them. The Lab was interested in high performance computing. The Japanese had introduced vector computers, and one of the things I was tasked with was understanding more about the Japanese computers and how they related to the computers that we had at our Department of Energy laboratory. So I remember going off to Japan for a couple of weeks and running benchmarks and studying and talking to the Japanese about their machines, and looking at the capabilities and limitations. We were interested in enhancing the performance of the software that we had produced and we realized that there was a rather complex mechanism there that had to be understood and could be used to do that enhancement; and that the way in which the data was organized in memory and the way in which the data was accessed from memory into the hierarchy of the system was going to be critical. I had done some studies about how to effectively address a matrix element and how that could be laid out so the compiler could effectively understand and generate code. I remember collaborating with a colleague at Argonne, Allen Haines, we jointly wrote a paper about unrolling loops to enhance performance on scalar machines. It was a very simple paper, and today seems quite obvious and trivial, but at that time it was there to enhance and to boost performance and that enhancement led to maybe ten, twenty percent improvement in performance on some of those scalar machines

We were also starting to look at parallel methods and trying to understand how to write things to exploit some kind of parallelism, even if it was very minor. And I remember in LINPACK looking at that for tri-diagonal matrices and there was a rather interesting algorithm in LINPACK that I guess I have my fingerprints on. It goes like this. One’s interested in solving systems with tri-diagonal matrices, they occur quite frequently, and on the surface it looks like a rather serial algorithm. You start at the top of the matrix introducing zeros below the main diagonal and you continue that process until you reach the bottom of the matrix, and then you have an upper triangular, actually upper bi-diagonal matrix, for which you can easily solve systems. The part that was exploited was that if you start this decomposition at the top, but also at the bottom starting at the upper diagonal, you can effectively have two factorizations going on simultaneously within the matrix. This leads to an algorithm which starts at each end of the matrix and sort of works away to the middle, one zeroing elements on the lower diagonal and the other one zeroing elements on the upper diagonal till they meet at the middle. That was an algorithm that we called the “Burn At Both Ends Algorithm,” and that was able to enhance performance, even on serial machines because of this exposure of independent operations, by a factor of approximately 20-25 percent. There were other projects related to software and enhancements, and making software available, so that relates to the NETLIB project, that was already talked about.

HAIGH: Your growing interest in parallelism, was that something that by the early 1980’s was emerging as a major concern for the community as a whole?

DONGARRA: Well it was clear, I would say, to some of us that parallel processing was going to be the way in which we did out scientific computing in the future. Individual processors had certain limitations in terms of performance and if aggregated together we could gain more computing power by using that sum of processors to help in solving our problems. That left us with the small problem of developing algorithms which effectively utilized that capability, and we started looking at ways in which we could do that. Some work had been done in the past in terms of numerical methods that worked in parallel but now we were embracing this with full knowledge that these would be the machines of the future. They would have very impressive floating point characteristics and these machines which have very high potential have a very unstable characteristic in the sense that small changes to the way in which you describe the algorithms can make dramatic changes in terms of the performance that you see. So you have to get it just right, and it may be a portability issue, in the sense of working on one system so that you do get high performance, but the way the exposure is done doesn't have the same effect on another system because of the way in which the memory hierarchy is laid out, because of the way in which the processors are connected because of the compilers, and other issues which could complicate that performance profile. So we needed to understand a number of issues there and it was a great time for learning and trying out new ideas.

HAIGH: Are there any major understandings or new insights in this area that you or a team that you were involved with, came up with?

DONGARRA: Because were interested in parallel processing and looking at how to write our algorithms to use parallel processing, it became clear that we had no way of really exposing that in our programs. The programming models that we had were quite limited. We used FORTRAN, and C was just becoming popular, and neither of those languages have parallel constructs within them. So we needed to think about extending the language in some way and having the ability of expressing parallelism within the languages. We dabbled with a number of mechanisms for doing that. My colleague, Danny Sorensen, and I put together a package that we called Schedule, which was a way of essentially graphically programming, which allowed the user to specify nodes in a graph, and those nodes would be executed once their data dependencies were satisfied. So think about a computational graph with nodes in it, these nodes are ready to execute but they're waiting for their inputs to arrive, and now think about constructing a numerical algorithm based on this, where the nodes then that have their inputs can fire or can execute. They would be scheduled on some parallel processor for execution and we would have a system, basically, an operating system which would effectively schedule, recognize, and do the execution of the nodes on some parallel machine. So that was a very early attempt at exposing parallelism, it is the parallelism in some sense, the user didn't have to engage in explicitly passing messages, this system was originally designed for shared memory systems but later some of those ideas went to distributed memory systems as well.

HAIGH: And approximately when would that have been?

DONGARRA: Probably in the mid 80's I'll say is when that mechanism was put in place. If you can't find it there I'm sure I can in my things here, I'll look. There's a paper that was written with Sorensen, it appeared in a book, the characteristics of parallel algorithms, that was a tool that we had, that we had constructed. [*Schedule: A Tool for Developing and Analyzing Parallel Fortran Programs*, J. Dongarra and D. Sorensen, The Characteristics of Parallel Algorithms, L. Jamieson, D. Gannon, R. Douglass, eds. MIT Press, pp. 363-395, 1987].

HAIGH: Yeah, I don't think you've talked about Sorensen yet. Was he a colleague at Argonne?

DONGARRA: Absolutely. Danny and I didn't share an office but we had offices next to each other. Danny's background is in numerical optimization, and numerical optimization has a lot of overlap with linear algebra, so we share ideas, and we started at Argonne roughly at the same time. We worked together on a number of things because of his interests in linear algebra. He was a contributor to some of the later packages that we developed, and certainly was a major player in some of the ideas that were coming about at that time. We wrote our first paper together in 1985, and that was on "Algorithm Design for Different Computer Architectures." [*Algorithm Design for Different Computer Architectures*, J. J. Dongarra, B. T. Smith, and D. Sorensen, IEEE Software 2(4):79-80, July 1985] It's a very short paper, I think it's a couple of pages, that was done with Brian Smith as well, and we were just exposing some ideas for writing numerical methods on different machines. Danny and I had done a whole series of papers, I'll say, working on these methods, current algorithms, again, for matrix factorization, and so forth. The paper that I remember as having the most impact, almost a critical paper, in terms of parallel processing and linear algebra, was an algorithm for finding the eigenvalues of a symmetric matrix using a parallel method, which was published in *SIAM Journal on Scientific Computing* in March of 1987.

HAIGH: I see, that's coded as No. 21, and it's called "A Fully Parallel Algorithm for the Symmetric Eigenvalue Problem". *[A Fully Parallel Algorithm for the Symmetric Eigenvalue Problem*, J. J. Dongarra and D. C. Sorensen, SIAM Journal on Scientific and Statistical Computing 8(2):139-154, March 1987].

DONGARRA: Right. So that paper, as its title implies, exposed a lot of parallelism in a problem which until then was unclear that a lot of parallelism could be exposed. It used a divide and conquer technique, and sort of set off a wave of interest in parallel algorithms for the eigenvalue problem in the community. So I would say that was one of the papers that is sort of a landmark in my mind in terms of the work that was done in linear algebra and parallel computing: it is often cited, it has interest today, it is embedded in our software in the ScaLAPACK project, it is well used and represents, in some sense, the state of the art of doing these kinds of computations.

HAIGH: Yeah, I'm noticing in general that from I think December of 1982, onwards, you start publishing very prolifically. Whereas, in your first eight years or so at the Lab, you hadn't.

DONGARRA: I guess that was because I was a student. As a student I was more or less learning, more or less trying to understand what was going on. My career at Argonne as a professional guy, that is as a card holding Ph.D. guy, starts roughly in the early 80's. A sequence of research projects began during that period, focusing on improving performance on software for linear algebra in terms of portable libraries. The BLAS came into play, we had another rather interesting paper, which had a lot of exposure, published in SIAM Review in 1984. [*Implementing Linear Algebra Algorithms for Dense Matrices on a Vector Pipeline Machine*, J. J. Dongarra, F. G. Gustavson and A. Karp, SIAM Review 26(1):91-112, January 1984]. That paper talks about linear algebra algorithms for dense matrices on vector pipeline computers, and it laid out some of the issues and problems with the algorithms that we were using. It characterized some of the basic computations, things like matrix multiply and LU factorization in terms of a loop structure, and how you represent the loops. Today you hear people talking about ijk algorithms and that comes from this paper, in terms of explaining how these algorithms behaved and how the data is referenced from memory. So that was an interesting paper and, again, it's one that's often cited. Not only that, it led to the community being involved in this area, so that sparked other people to look at and also to start understanding what the implications were of the memory hierarchy in terms of matrix algorithms.

HAIGH: Would you say that you've had the same level of pressure to continue publishing at the Lab that you would have done in faculty position?

DONGARRA: I never thought I was pressured at Argonne, and at the university here I don't think there is pressure either; but, I think some of my junior colleagues feel the pressure more than I did Publishing was thought to be a good thing, but I don't know that I remember it as being critical for my advancement, although I'm sure it was at some level; but I surely didn't feel any pressure. and it was just not something that I thought about It was more of that we're doing these great things and we needed to expose them and talk to other people about them. Sometimes they were recognized as being good things and other times we had rather mediocre ideas and those ideas really didn't pan out in some sense.

HAIGH: So the drive was really coming from inside?

DONGARRA: Yeah, oh yeah. It was motivated and driven by our desire to understand something. We were at the crest, we were at the point where other people hadn't arrived yet in terms of parallel processing, vector processing, in terms of portable software, and we were exploring a lot of things. Maybe you can call it low hanging fruit, but there was a lot of it and we were very eager to test ideas out and to explore things and to expose those ideas to other people.

HAIGH: What led you then in the end to leave the lab and move into the university environment?

DONGARRA: That’s a good question. It wasn’t an easy decision to arrive at. I had lived in the Chicago area my whole life, and now I had worked there for some 15 years roughly, and an opportunity came to move to a different setting. I had been very successful at Argonne, and to be honest, I think I could have enjoyed working there for my whole career, but an opportunity came about. The opportunity was to work at a national lab and also at a university. The position that was proposed was a joint position at University of Tennessee and Oak Ridge National Laboratory (ORNL). Now the University of Tennessee is a fine place but it hadn’t really been on my radar in terms of a place to move. But Oak Ridge was a place where I had colleagues; I understood how the lab worked more or less as it was very similar to Argonne, so I felt very comfortable moving into that position. It was something very similar to what I was experiencing at Argonne, it was expanding my scope in some sense being a university professor, and it was time for a change. I had been at Argonne for a long time. It was a decision point. I guess at the time I was feeling that if I didn’t make a change I was never going to make a change and that would be okay, but not having experienced being a professor at a university would be an unanswered question in my mind.

So it was an opportunity to do something, it was for a position which was comfortable in terms of the research setting, and it would allow me to be a professor and to have students and to do other things that were perhaps not as easy to do at the lab. At the lab I never felt pressure in terms of funding but it was clear that your funding came from the Department of Energy and I knew at the University I would be a free agent, free agent in the sense of being able to acquire funds from different agencies. Now having said that, at the Department of Energy I was in a rather unique position. I actually had grants from The National Science Foundation, and to this day people are wondering how I ever managed to accomplish that. I have to say I’m not sure, but we had ideas and it seemed like the right place to fund them was through The National Science Foundation. NETLIB was funded out of The National Science Foundation, projects like LINPACK also received NSF funding to some extent, and so I had exposure with writing proposals and submitting them to other agencies. It was harder to do that within the Department of Energy, within Argonne at that time, and moving to the University was a natural thing to do, so I felt quite comfortable.

HAIGH: Yeah. In your resume I think lists five, six, yeah five small, well one $567,000 grant from NSF during your time at Argonne.

DONGARRA: So, it was something that was possible and certainly I felt like it was the right thing to do.

HAIGH: Had any universities made attempts to recruit you prior to that?

DONGARRA: I don’t remember it, and I don’t remember even being interested in other universities. The attraction was not to go to a university, per se. The attraction with the position I currently have is it’s a joint position between two places. One place I felt comfortable existing in, and the other thing being something new I could try. I guess in the back of my mind I always felt if the university thing didn't work out I would have the lab to fall back on.

HAIGH: How has the arrangement with Oak Ridge worked out?

DONGARRA: Well that's an interesting story. At Oak Ridge things are similar to the way they were at Argonne but a little bit different, and everything is about money in the end. At the university it became very easy for me to reach out and attract research funding, to write proposals to gain interest by various agencies and to build a sizeable program there. At the lab it's a lot harder to strike out to different agencies. It kind of gets a little bit easier, but at the time I started it was much harder. And funding really came from the Department of Energy, solely, and those came through programmatic funds and, I'll use the word, more or less like an entitlement. The Department of Energy goes through cycles where they have good times and bad times, and if you're in a bad time at the Department of Energy it's going to be hard to make ends meet and to keep everything going the way you would like it to go. And having a portfolio which had different places as sources of research funding was certainly the right mechanism. There are issues about overhead: at the university our university overhead rate is roughly 45 percent. At the lab the overhead rate is much higher in the sense that it's like 200 percent, so your money goes a lot further at the university. So if I know I'm going to be able to attract, we'll say a half a million dollars, I get much more for my money at the university than I would at the lab. So there's sort of a natural binding to the university for research dollars because of that overhead, plus the bureaucracy I find is a lot easier to work with at the university. Mind you, at the university it's no picnic, but it is easier than working through the lab in terms of getting things funded and having everybody agree. So in the end my research portfolio was built at the university, the lab felt I wasn't doing my share and it put pressure on my joint position, and in the end I actually gave up my original joint position at the lab and the university. I still have a joint position in the sense that I go to the lab one day a week to work and interact with the people there, but the original position that I started with here in Tennessee has been changed so now I'm paid full time as a university professor and consult one day at the lab.

HAIGH: So over time the balance shifted very much then in the direction of the university?

DONGARRA: That's correct, and that was because of a number of issues primarily related to funding and how things are more streamlined at the university or, allow me to act as an entrepreneur, I think that's a better way to say it. So at the university you're an individual and you get to do the things you want to do, and you're judged on that; at the lab there are programmatic efforts and you have to fit within one of those efforts and find your funding stream in that direction, and over time projects may change and as a result you have to find new places to fit within the changed structure. At the university you do your thing and as long as you can attract research funding nobody's upset with the way things are going or your progress.

HAIGH: Right. Were there any issues that you had to negotiate with the university or with the department when you first arrived?

DONGARRA: The critical thing at a university is space, so negotiating space is a crisis. When I started here I had one person. Today my group at the university has 50 people, and finding space for the 50 people in a way that's conducive to carrying out our job is always a struggle. I constantly have issues with the university over space and guaranteeing that we have enough good space for us to work in. So negotiating at that level is an ongoing effort, negotiating in terms of teaching, you know, that's an issue that was handled and dealt with, so I would say the university's been pretty good in terms of providing for our needs and allowing us to carry out our job.

HAIGH: Now looking at the list of grants at the end of your resume, it seems like after you arrived here they began to pour in relatively quickly. Two grants totaling about $430,000 a year from NSF, 1989 and 1993, and then more than a three million dollar grant from DARPA in 1991,

DONGARRA: Right, so that was a big grant. That was the grant that led to ScaLAPACK. My colleague, Jim Demmel and I were the major architects for that. Jim Demmel's a professor at Berkeley, and a numerical analyst as well, and he and I set out to develop a software package that would basically take EISPACK and LINPACK at an abstract level, and develop an updated state of the art package that would effectively be a replacement for those packages. Enough time had passed and enough new ideas were out there, so software engineering things needed to be done. We wanted to insure portability and efficiency at a number of levels, and we thought we had a way of doing that. DARPA was very interested in the development of high performance architectures and machines at that time. We approached them with an effort to develop a numerical library for those machines, and we were successful in that attempt and had a project which probably had a five to six year lifetime, maybe even a little bit longer, to develop that library. It was a wonderful project, it involved a number of people at various places, it involved the community at a number of levels, and it produced software that was probably the first portable numerical library for parallel machines. It did it based on a certain standard which was just coming about at that time, so everything sort of fell together at the right moment. ScaLAPACK works on distributed memory machines and uses a number of things that we had developed with the BLAS. It uses a set of parallel BLAS that we developed; it uses a message passing library that the community was developing called MPI; and it was built also in a way that could use another parallel library that we had developed at Oak Ridge, and Emory and the University of Tennessee called PVM; so it was capable of using parallel systems and reaching high levels of performance, and doing it in a way that was parallel across different machines. So that was being at the right place, I'll say, at the right moment with software which allowed users to fully exploit the capabilities of their high performance machines.

HAIGH: So that's ScaLAPACK. Now you haven't talked in any detail yet about LAPACK, and the two projects seem to overlap somewhat chronologically and in terms of people.

DONGARRA: Yes, that's right. So LAPACK was, again, this effort to take EISPACK and LINPACK and make it work on serial machines and shared memory machines. There's explicit message passing going on within LAPACK. It's intended to update the algorithms, it's intended to be portable across shared memory machines, and it's intended to achieve very high levels of performance on those architectures and do it in this portable way. So LAPACK was structured on top of the BLAS. It was focused whenever possible in terms of Level 3 BLAS, matrix-matrix operations, to expose the high levels of performance that one can get out through matrix multiplication and its related operations. The idea was that if we can write those at a very efficient implementation for a given architecture the whole package would run very efficiently. So that was a project which received NSF funding for the most part, and DOE funding, and was joint between Jim Demmel and myself, people at NAG, and a number of other individuals. From the moment it was released it was the standard for numerical software. It exposed issues that were of interest in terms of parallelism, in terms of performance, in terms of portability, in terms of software engineering, and allowed us to very easily and quickly deploy that package across machines and let our users get experience. We used the same basic structure in terms of test sites: sending things out and getting results back. We had evolved a little bit in terms of our technology of doing that, so things went a bit smoother in terms of using the network to convey our algorithms and software and get results back, and even for us to run things in a remote setting now was a more routine feature.

HAIGH: So that's the LAPACK project. Now actually backtracking and picking up on one thing that you said there. You've discussed the origins of the original BLAS, which I think you said came out of JPL.

DONGARRA: Right. The BLAS routines are sort of critical in all of the software development. Chuck Lawson, Dick Hanson, and a few other people came up with the original idea. They got the community together, had discussions, got input, developed a set of routines that had an interface and a reference implementation. So the key thing there was not the implementation but the interface. This approach was simple to use, covered a lot of options, and would be exposed in single, double, complex, and double complex arithmetic. So now there was a method to doing it and the reference implementation specified what the operation should do and now people just had to implement that efficiently on their machine, and people started to do that.

HAIGH: According to the list here, Level 2 was 1987, and Level 3 was 1989.

DONGARRA: Right. So it became clear that these routines that Hanson, Lawson, and colleagues, produced were not enough. They weren't enough to get at performance, so we had to enhance the areas that we looked at. We were developing or thinking about developing LAPACK at this time, and we wanted to structure a package in terms of these higher level routines and we clearly saw the need for matrix vector operations doing things like multiplying a matrix times the vector, operating with triangular matrices and vectors and rank one updates. So the Level 2 BLAS routine encapsulates that sort of level of operation, working with order n squared pieces of data, and order n squared operations, so matrix vector is sort of the cornerstone implementation there. And we defined an interface and had a reference implementation and test routines for that, for that package, and that was done in 1987.

And then after that it was obvious that we had to extend that, and the Level 2 and Level 3 BLAS followed what the original things had done in the sense that we got community input on all of these efforts. So it wasn't just a small group of people doing something, it was a group of people working in concert with the community to try to develop a standard interface for these things. We talked to vendors, we talked to users, we talked to colleagues about how to do that. From those discussions came the final standard (these are all de facto standards, they're not standards in the sense of being sanctioned by any international organization, they're sanctioned by the community) which was embraced by the community primarily because after they were defined the vendors went off and implemented them in an efficient way and users could now see that if they used that standard, they would have a way of getting portable performance out of their algorithms. That was done for the Level 2 and as well as the Level 3 BLAS, following that same template we had in place. So that was a very good model to use and we did it successfully for both of those efforts.

HAIGH: And is this switch to, essentially, a higher level of abstraction, related to the techniques that you discussed earlier for taking advantage of the characteristics of vector architectures and memory hierarchies?

DONGARRA: Right, yes, exactly. So we wanted to focus on that as a way of generating these very efficient operations. Our software packages get structured on top of these BLAS operations, and we hope that the vendors will go off and write a good implementation of the BLAS. In many cases they did, in some cases they didn't, and sometimes the software was generated but not made freely available. That caused some stress in the sense that we didn't have a way of getting portable performance out of our software, and we started thinking about how we could implement a portable implementation of the BLAS which would be efficient.

Understand that the BLAS is a definition, semantics, and a reference implementation. The definition describes the way in which the routine is called, the semantics describe what should be done for these operations, and then a reference implementation does the operation in a very straight forward way. If you use the reference implementation you're not going to get the performance. And we began to try to figure how we could implement something which would be very efficient across all the machines, and that led us to a quite recent project called ATLAS. It led to the community starting to explore this space of self adapting algorithms, self adapting numerical algorithms, implemented in software, that would try to be portable and efficient across all the architectures. So ATLAS is an effort in program generation. ATLAS is going to generate the BLAS for a specific architecture; that's all it does. The BLAS consist of maybe 50 different routines, and what ATLAS does is to go out on the machine that the package is going to run on and probe the system to try to figure out the characteristics of that system. Characteristics like the size of cache, how many levels of cache it has, how many floating point units it has, does it overlap loads and stores with floating point operations, how long is the floating point, how long does it take to do a floating point instruction, are there certain other characteristics of that architecture which would be exploited.

The compiler is on the critical path to performance, and in many cases it doesn't do a very good job of optimizing. We can help it by structuring the code in a certain way and that's really what ATLAS is trying to do. So the first stage of ATLAS is probing the system for architectural features, and it goes through a series of experiments to do that. Then, based on the experiments, it's going to implement something like matrix multiply. It's not going to implement one version of matrix multiply with the knowledge it has; it's going to implement thousands of versions of matrix multiply for that machine, each of which has different settings according to the parameters that it just found out about. So it generates thousands of versions of matrix multiply with different settings, and then runs each of them and measures the performance, and then it will pick the one that has the highest performance on the machine. So it's going to generate software by carrying out a series of experiments to probe the architecture, and then by carrying out a series of experiments to determine which implementation is best for this machine and compiler.

HAIGH: You said that's a recent project.

DONGARRA: The project is still going on. One of my former students still works on this project today. There's a great deal of interest in the community in this approach, and a lot of people are trying to do things similar to it today in terms of self adapting to the architectural features.

HAIGH: And who were the other people working on that project?

DONGARRA: The primary person is Clint Whaley, so today he's a student at Florida State University today, and Antoine Petitet who's an employee at Sun Microsystems in France today. They were both here as students and workers and then had gone off, and continued their interest and involvement and actual use and development of that software. There are other projects similar to that: my colleague Jim Demmel has a project called BeBOP, which tries to do the same kind of thing; there were similar projects with respects to Fourier transform; there's an effort called FFTW which does the same thing; and an effort at the University of Houston which also follows along those lines. FFTW does it for FFT's, our stuff does it for matrix operation.

HAIGH: Okay. And then moving back again slight in time, was there anything in the Level 2 and Level 3 of the BLAS that would help in unlocking the benefits of parallelism?

DONGARRA: Well, the BLAS and LAPACK were designed for shared memory machines so we looked at exploiting parallelism at that level within those packages, but it became very clear to everyone that shared memory had limitations, and those limitations were going to prevent us from going to hundred or thousands or tens of thousands of processors. That led to exploring a space of distributed memory software and applications. In that case we needed new message passing, and the numerical libraries that we had didn't do message passing at that time. We had another problem in that there was no standard way of doing message passing. We had parallel machines, and each machine had its own way of expressing its message passing characteristics, but there was no portable way of doing it. That led to a project called PVM, designed by people here in my group at UT, at the Oak Ridge National Laboratory, and also at Emory University. We received some money from the Department of Energy to do this project called PVM, Parallel Virtual Machine. The idea was to come up with a framework to tie together workstations, and also to allow that same software to be used on a parallel machine where we had multiple processors sitting in a box and one could use them to carry out a simple operation like a matrix computation. And PVM allowed us to run a program on these machines and do it in a portable way so it would run not only on a set of workstations but also on this multi computer, and do it in a way that allowed fairly high efficiency. So PVM was an effort to do this parallel processing and it was one of the first packages to do it. It wasn't the only one; there was great interest in the community in developing such a concept, and we were at the forefront of that activity.

PVM was done by a small group of people and clearly these ides were in demand by the community at large, and that caused us to generate a community effort to define a set of message passing routines called MPI. We started something called the MPI Forum, I was the chair of the first MPI standards group, and within a two year period that forum had specified what message passing should look like on parallel (distributed memory) machines, and that led to this thing that we have today called MPI, which is the standard for doing message passing on all of our parallel machines. A lot of vendors implement that today.

HAIGH: And who were your collaborators on the PVM?

DONGARRA: So PVM was done jointly by people at UT, ORNL, and Emory. Adam Beguelin and Robert Manchek were the guys from my group here, Al Geist was the person at Oak Ridge National Lab, and Vaidy Sunderam was the one at Emory University. PVM was done jointly by the group, but the code for the most part, was done by a student of mine, Bob Manchek, a superb programmer who reached the level of expertise here that really made that package shine, makes it very useful even today, and clearly has a life beyond its original concept.

HAIGH: And was PVM itself used for ScaLAPACK?

DONGARRA: PVM was used as an interface for ScaLAPACK. ScaLAPACK was a modular designed project. It had a series of levels in that modularity, and at a certain level message passing had to be done, so we defined an interface within the ScaLAPACK package that would do the message passing. It exposed a level of notation in which we could easily write our software, and that was something we designed, and then below that was the real message passing system, and we could plug PVM into it, we could actually plug other systems into it as well, so we had interfaces for PVM and MPI for ScaLAPACK, and things came together, again, at the right moment. So PVM and MPI were being designed and implemented and being used at the same time when our software was being ready for release, so we have one of the first packages built on top of these de facto standards for doing message passing.

HAIGH: So was PVM superceded by MPI, or are they different levels of abstraction?

DONGARRA: PVM came first. PVM provided a lot of ideas that found their way into MPI. MPI was designed by a committee of twenty people from commercial settings, academic institutions, research centers, and vendors, all pouring their ideas into this package, and because it was a committee it was done by consensus. It has a lot of ideas in it, some of which are really terrific and some of which are rather obscure, but that’s the nature of committee work in some sense. The package goes well beyond PVM in terms of its interface and in terms of flexibility. Again, PVM was done by a small group of people sitting in the mountains of Tennessee is the way I like to describe it, where MPI was done by a community effort to develop a standard that would be good for not just workstations, and multi computers, but for parallel processing in the large.

HAIGH: Now the period around 1994-1995, was obviously a traumatic time for the vendors of larger and more expensive supercomputers. The various Crays weren’t doing so well, and Thinking Machines and KSR folded. Did that have an impact on work in this area?

DONGARRA: Well it was perhaps a natural thing to have happen because the market for high performance computing is rather small. We had a lot of vendors, a lot of startups, engaging in the development of parallel machines, so they saturated the market, and because of that saturation they couldn’t all exist, and then there was a big shakeout which is perhaps still going on today. So we came through a period where there were a tremendous number of vendors, then that number was dramatically reduced and today things are starting to build back up. Actually today I would say we have a pretty vibrant field for machines and companies interested in high performance computing. Some of those are led by cluster based, commodity based technologies, which are quite popular for their low entry price point for high performance computing.

HAIGH: Okay. So maybe to shift back and look at a bit more detail at ScaLAPACK. You’ve explained that the key difference between that and LAPACK is the difference between the shared address and the distributed address architectures of the systems. Other than that do they provide essentially the same functions?

DONGARRA: ScaLAPACK and LAPACK are very similar for the well used functions; for the most part they have the same functionalities. There are certain differences and LAPACK is a super set of what we have in ScaLAPACK. ScaLAPACK doesn’t quite provide all the functionality; it tries to cover the more important aspects of the package, but there are certain things which because of time, funding, and interest, were not fully developed.

HAIGH: Has there been any kind of feedback from this interest in these more parallel architectures back into the mathematical research community to go out and look for methods which would work better on these machines?

DONGARRA: Oh, sure, there's always this feedback that goes on in terms of methods that work better with interconnects, with the processor architecture, with the number of processors, with the characteristics of certain things. Today there's continued interest in parallel processing and some of the things that we're exploring today relate to issues of fault tolerance. So today we're seeing machines that are being used at our largest centers, having tens of thousands of processors, and we know about machines that are hundreds of thousands of processors which are on the horizon, and our programming model doesn't really provide any mechanism for recovery from a fault. So if with a hundred thousand processors a processor will go down, we can say, "within the next four hours we're going to lose a processor." If you have a long running computation that's running on this machine we want a mechanism to be able to sustain that failure in the presence of those errors. We have no mechanism in our programming languages for doing it outside of doing the check point and then recovering from the check point. We'd like to be able to sustain a failure and carry on the computation while that failure is there, we'd like to be able to express that within our algorithms, and we'd liked to be able to express that within our programming model, so people are pursuing that. One of the other problems we face, with respect to software, is the fact that we haven't changed our programming model in the last thirty years. FORTRAN and C are the programming paradigms that we use, and we still use, and they have no mechanism for doing parallel processing, so all the processing comes about through functional calls that are made to do the message passing. We really need to think about new languages that can effectively exploit parallel processing. Today we have to program at a very detailed level. The new language would allow us to express that level of parallel processing at a deeper level within our programs and then do other things, like recovery and fault detection. I won't say there aren't any new languages but those languages have not been embraced by the community and are not widely in use today.

HAIGH: One of the interesting things that Cleve Moler said was that back in the early days these new methods, eigenvalues and so on, were extremely broadly applicable to a very wide range of users, but he felt that over the last decade or two that most of the methods that a system like MATLAB needed had been pretty well understood and had fairly well implemented, and that he didn't think that current leading edge research in numerical analysis necessarily needed to be fed back into the product at regular intervals. Obviously MATLAB is being a rather different constituency from these extremely high end kinds of system. Now in this kind of area you're dealing with, with these LAPACK and ScaLAPACK, do you feel that there are still major unsolved questions in numerical analysis that you wish you had the answer to, and if you had the answer that you would rush straight away to plug them into the packs and make them available to a broad range of users?

DONGARRA: So I would say the answer is sure, yes, we would like to do that. Some of the issues we're looking at relate to performance from the standpoint of robustness of algorithms, from the standpoint of efficiency in exploiting large numbers of processors, and also efficiency running onto these modern processors. So just getting back to Cleve's point, which perhaps is valid for a number of users, MATLAB is a wonderful tool for exploring new ideas; I use it all the time, and I'm happy to pay for it. I have MATLAB on my laptop; it's my favorite programming language. A lot of software that I wrote is in MATLAB, it uses our stuff and even though it uses our stuff, the value that MathWorks has added to our software makes you want to pay for it to use it, so that's a rather strong statement about MATLAB's usefulness. But one of the problems with MATLAB is that I'm limited to the machine I'm running on, and what I think a next frontier for systems like MATLAB would be is to tie the ease of use, or ease of expressiveness, that we get from MATLAB with the power of some parallel machine. Allow users to easily write, construct, and use MATLAB, but have that execute on some device that's out there which has thousands of processors and do it in a way that is efficient. There are a number of groups looking at that today, and that's one of our projects as well: exploring that space of how to tie together these interactive systems with, say, some cluster which has the capability of large numbers of processors and very efficient execution.

HAIGH: So conceptually speaking, would some relatively loosely coupled cluster of commodity machines be a special example of a "distributed address space computer" that ScaLAPACK would work on, or does that need a fundamentally different kind of approach?

DONGARRA: I'm sorry, say that one more time.

HAIGH: Would something like the Virginia Tech celebrated collection of G5's, constructed with a few thousand commodity machines, conceptually be a special case of a distributed address space computer that ScaLAPACK would work on, or would it require on a more fundamental level a different kind of approach?

DONGARRA: No, no, no, that's exactly what we're intending. The people working on the G5 collection at Virginia Tech (by the way that machine is no longer there, they're replacing it with some other device) ran our code on their machine. There's a benchmark we have called the LINPACK benchmark which measures the performance of systems and then reports them; and then there is something called the Top 500 which ranks the 500 fastest computers that we have today based on solving a system of equations, running something that looks at lot like ScaLAPACK. So there's a way for people to run a benchmark, to measure the performance and then for us to report the results. I've been fortunate that people have taken interest in running that software and reporting the results on their machines, so I've been able to get performance numbers on high performance machines, and have acted as sort of the bookkeeper of that list in the sense that people send me the results and want me to list their machine to show the speed of their system. That gives us an opportunity to look at some of the issues of performance, and to understand where some problems lie in terms of lack of good performance. It also allows us to get exposure of our software on a wide variety of machines which helps us in terms of developing new things.

HAIGH: So this top 500 list, did that grow naturally out of people sending back their test results from the testing stage of these packages?

DONGARRA: Not really. It grew out of the LINPACK benchmark collection, which was an attempt for people to easily measure how fast their machine runs, not necessarily to test our software, so it's a slightly different measure, although they both had the same origin.

HAIGH: So, moving back again slightly to return to LAPACK and ScaLAPACK. Now you've said that during the testing cycle and the development period it was pretty much the same process as the earlier PACKs except that you made increased use of the net for collaboration and code distribution. Now how about when those packages entered use. Was there anything different in the way that you would handle feed back with the end users, or the way in which interim releases and bug fixes were handled, that was more fundamentally different?

DONGARRA: So I'm not sure which question you're asking. So when packages like EISPACK and LINPACK ended their life did we do something different?

HAIGH: No. The question is about the processes by which the software is released, used, improved. Now in an earlier answer you'd said that nothing fundamental had really changed in the way that the team collaborated internally, except that the net helped a bit, and that nothing really fundamental had happened with the way that the test sites were used except that you swapped data electronically. I was wondering how about after the first release, did anything change in the way that you got feedback or bug fixes from users, for example, did anything change in terms of it being easier for people to contribute ideas and improvements? I noticed, that each package shows a number of releases, first release, second release, third release, did you, for example, move to something like today's commercial software where there are frequent tiny upgrades and bug fixes between major releases?

DONGARRA: At the 10,000 feet level things are not changing too much, of course it does change. The web and the way the Internet works today has greatly simplified how we do business. Today I wouldn't think of making a tape and sending it to somebody:, I would point them to a website and say download the software from here, do this kind of thing and send the results back to us using this web interface. The web simplifies the mechanisms we use to gather the information, it makes it easier for us to run on remote machines, and it simplifies the time that it takes to do these things, so I would say everything's been enhanced. But then enhancement also occurs with the frequencies of which we make changes to our software, fix bugs, and the way in which those bug fixes may be distributed through patches and so on. So all those things are true: it does make it easier to release software, to do our business and to do our job.

Those are tools that we're using but is there a fundamental change in how we do business? Again at a high enough level the answer is no: we have our test sites and we have our friends that we send code to; they run it on their systems and test it out on their problems; they send back ideas, changes, corrections and enhancements to our software; those enhancements get made here or through the contributions of individuals at other places, and then we update the package in that way. It's the same sort of mechanism that we used before, but now we're using the tools that we have available today, which certainly simplify things and make our life a lot easier, a lot faster; but I would say we haven't made major changes in the way we do our business.

HAIGH: Yeah. So to return then to the comparison with the free software movement. It remains true that the actual core development is very much centralized within the team, and that users don't spontaneously contribute appreciable chunks of code or expansions to the package?

DONGARRA: I would say that's true: there are not major chunks of things being contributed by users to these routines. If there were then we would include them in the team. So that's the other way of doing that task. And that has certainly worked for our community and worked for us in this way. It's allowed us to have control over the software, to guarantee a certain level of integrity and portability, and all the other qualities that we wanted in our software, and provided the community with a service and a way of making improvements and enhancements to that.

HAIGH: Now with those two projects you'd already mentioned Demmel as a key collaborator. Who were the other important people on those projects and where were they coming from?

DONGARRA: Right. Jim has been a close collaborator ever since he began his career. He was a student at Berkeley, went to NYU, and did his postdoc out of Courant, then went back to Berkeley and has been there ever since. We've collaborated, I think, ever since he graduated or before he graduated at Berkeley. We developed this LAPACK stuff with the help of very talented people. Ed Anderson a student, who worked here at the University of Tennessee, was probably the first person who worked with me here. Zhaojun Bai was a collaborator with Jim, and today is still an active contributor with our activities. He is now at Davis, University of California Davis. Chris Bischof was a postdoc at Argonne and he continues to have interest in this area. Jeremy De Cruz and Sven Hammarling are close colleagues from NAG, and Jeremy is now retired but Sven is still active and collaborates and contributes to these ideas, both in terms of LAPACK and some of the other packages that we have. Anne Greenbaum is a professor today at the University of Washington, and is still interested in these areas. Al McKinney was involved but I've lost track of Al; he is somewhere I'm sure, but I've just lost track of his activities. Susan Ostrouchov changed her name to Susan Blackford, and is still active in this area. She was very instrumental in terms of the mechanisms for software for the release for the documentation, and so on. You need very talented people for that and she was our talent. And Danny Sorensen is just a very clever mathematician and software developer, and has helped with all these projects.

HAIGH: Yes, you had discussed him yesterday.

DONGARRA: Yes.

HAIGH: And the team for the ScaLAPACK project?

DONGARRA: Right. ScaLAPACK is made up of a lot of the same characters, with some new faces as well, including a couple of postdocs who were here at UT: Jaeyoung Choi and Allen Cleary. David Walker originally from ORNL, was involved and is now a professor at Cardiff University. Clint Whaley, who was a student here, worked on ScaLAPACK in terms of a distributed communication library and was very instrumental after that in developing this ATLAS package for our collection. Antoine Petitet played a big role in developing this net of parallel BLAS that we use within the ScaLAPACK collection, and he earned his Ph.D. here with me. Greg Henry, who works for Intel, was here as a visitor, and was very helpful in the numerical methods. Ken Stanley was a student at Berkeley and Inderjit Dhillon, who's a professor at Texas, was involved in some eigenvalue calculations. So we had a well rounded group of people, some very experienced, some postdocs learning how these things worked, making major contributions to the package overall.

HAIGH: Yes. Now I noticed in the book "Sources in Development of Mathematical Software", the article that you wrote with J.W. Stewart, called "LINPACK – A Package For Solving Linear Systems", you make a similar comment in there to the one that you made in our interview about the, I think you called it "bickering", that took place between the four collaborators on LINPACK. [*LINPACK--A Package for Solving Linear Systems*, J. J. Dongarra and G. W. Stewart, Sources and Development of Mathematical Software, W. Cowell, ed. Prentice Hall, pp. 20-48, 1984]. And on Page 46, you say that the major difficulties with the way the project was organized was that there was no senior member with final authority to decide hard cases, and you say that something like a court of last resort should be available to issue arbitrary rulings on things –

DONGARRA: Right –

HAIGH: Now is it fair to say in these later packages that you yourself have taken on that role more and it's been less of a completely collegial collaboration.

DONGARRA: So I would say on LINPACK and EISPACK that's probably true, in the sense that there would of course be a discussion about issues and hopefully consensus would be reached from that discussion. But occasionally you don't have that sort of coming to a conclusion and somebody needs to make a ruling, and the rulings at a certain level become arbitrary. I would take on the role of saying, "it's got to be done, let's do it this way here, we can argue about it some more but I'm going to make a ruling that this is the way it's going to go and we would do it that way," and that seemed to work fine. So perhaps we learned some lessons from LINPACK where we had a group of people who would come together and with strong ideas and opinions and would sort of sit in a room and fight about it, and maybe stew about the decision after it was released without having some final authority to say this is the way it should be done. We learned our lesson from that and have a central person, who in the end, can make these arbitrary decisions.

HAIGH: And in discussion of free software, that position has sometimes been called the benevolent dictator. Is that a fair description?

DONGARRA: I think so, yes, that's right. Somebody needs to make that last call, somebody who everybody respects and won't challenge. I'll say that works as long as there's been an open discussion and that people have had a chance to air their opinions and can, in fact, appeal at some level, and then live with the final ruling.

HAIGH: You'd said that intellectual property wasn't an issue during your time at Argonne, except for the issue of publicly funded competition with private industry. Now moving to the university environment have there been any issues with the intellectual property licensing or technology transfer staff at the University. Obviously in recent decades, there have been people much more concerned with patenting things, exploiting intellectual property, generating revenue streams. With these later projects, is there anything that has or could have been patented?

DONGARRA: I try not to get into it. I'm not asking questions and I wouldn't approach a lawyer and ask them how I should do business here at the University. I'd rather carry on in the way that we've been doing things over the last 20, 25 years, and not ask those questions because if I ask, I'm afraid I know what the outcome will be: I'll be forced to engage in a license that I don't agree with, and then I would probably seek other ways of distributing our software. The university wants to make money: at the core of its concern is how they can reap profit from the intellectual property of its professors. Maybe that's being slightly unkind, but that's at the core of what its interests are, and I don't think there is enough money to be made to warrant that level of interest. I think the community is going to benefit far more from the funding poured into the effort through the National Science Foundation, Department of Energy, and other sources, through an open source effort from the community, rather than by laying claim to ownership.

Doing that, I think, would reduce the availability of the software; it would decrease the visibility and effective use of that software, and it would position things so that people would try to write their own software to do these operations. Perhaps in many cases they can do a fine job, but in other cases a tremendous amount of experience and knowledge has gone into certain issues of accuracy and performance of our software, and it would be hard for everybody to get that right if they were to go off and do it themselves. So I think through a community effort we can provide software that has a very high level of quality; that is robust, efficient, and portable, and do it in a way that involves the community and doesn't really engender a lot of cost from the government along the way.

HAIGH: You had said that the earlier packs included no kind of copyright statement at all –

DONGARRA: I believe that's correct –

HAIGH: Is there a copyright statement or any kind of license associated with the more recent projects?

DONGARRA: Maybe there's a copyright in some of them just because people want to see their name in there. It's hard sometimes to object to having somebody's name associated with the software, especially the author's name. So I think there could be a name and there may be copyright in some of our software. I try at almost all cost to say our software is licensed with a BSD style license, so it's freely available and people can use it and embed it in their codes and transfer that code to other codes and commercially sell that code. We're happy with that, as long as they describe that our software is being used, and that if they change it, they make it clear what they've done to it. But, again, the license doesn't infect their code in any way; it has free use associated with it.

HAIGH: That's interesting. A moment ago you sounded quite ardent about the benefits of freely distributing software avoiding any duplication of effort for those kinds of things, yet unlike some participants in open software you don't seek to force people who would take and use these packages to release in turn the enhancements back to the community. Is that because you think it's unrealistic?

DONGARRA: I would certainly encourage them to release enhancements back to the community, but I wouldn't force them to do that. That is, if a commercial company finds a way of improving something and they want to use that to their benefit, and they don't want to bring it back to the community, I guess I don't find that objectionable. I would hope that they would, but I'm not going to force them into that position with their software. And I think that maybe we'll learn something or try to work a little bit harder to figure out what they've done in that case, but it's not a situation where I want to force their hand to release something that they've uncovered.

HAIGH: Okay. Now looking at the list of major projects on your resume. The final one, and one I don't think we've talked about yet, is NetSolve.

DONGARRA: NetSolve was an attempt to merge a few things. Today, one of the hot areas is grid computing. Grid computing has tried to use distributed resources in a transparent way where the user doesn't have to be active in all of the details of getting the job started on some remote platform. With NetSolve we've tried to put in place a mechanism to expose our original library software to users. Today with one of our packages they have to take our software, download it, install it on their machine, get it ready to run and then link to it, and execute that software. NetSolve is trying to make it simpler in the sense that we've already installed NetSolve on some machines out there on the network, and they can make a call to NetSolve and specify the routine they want to use. NetSolve will find the best routine, best in the sense of the closest and the one that will provide the fastest execution for that user, and do it without the user having to be involved with any remote procedure calls, or any remote resource discovery, or any details about accounting, and so on and so forth. So it's a way of getting access to software which they don't have currently available on their machine and doing it in a transparent way using the software on some remote platform.

NetSolve is more of a software accessibility mechanism than it is about getting access to hardware. When people talk about the grid, they're talking about getting access to resources, and they mean resources in the sense of hardware resources. I'm referring to NetSolve as a mechanism through which you get access to not only hardware but also software. I can have a machine in the corner of my room which is serving out LAPACK, and people around the world can submit their matrices to my machine and my machine will solve them for them. Now that's one way it can be used, and that's perhaps a very socialistic way of using it in the sense that I don't have to have an account and I just use your machine to get the answers back. But, we can think about another situation where I leverage the resources I have on campus and set up machines on our campus which can serve out that software and only our users on campus would have access through this mechanism, and in fact that's what we have here at UT. You set up a number of NetSolve servers, they serve out our software, and people around campus can then get access to our software without having to download and install that software on their machines. Again, the motivation originally was numerical libraries and trying to get use of that software for their problems without having to go through the effort of downloading the package and doing the installation.

HAIGH: So conceptually that would be very similar to what, in the business environment, would be called an Application Service Provider?

DONGARRA: Yeah, that's right. So here I don't have to know where my problem is going to be solved. I have to say I'm going to call a function, which is going to do a remote procedure call, and I don't have to know about the details of the remote procedure call.

HAIGH: Right. Kind of parallel with that, I know that in the areas that John Rice has been working in, a big problem is helping mathematically unsophisticated users figure out what the appropriate method to use is. Now my impression is that in the linear algebra and matrix area you're working a lot less with approximations and much more with definite answers, so that's less vitally important. Do any of these packages help people select what methods they should use for their specific problem?

DONGARRA: Right. One of the efforts that we have here, under the umbrella of our self-adapting numerical software effort, is doing just that. It relates to a user having a large sparse matrix problem. You want to solve a system of equations perhaps, and there are many methods out there and actually many software collections for doing this, but choosing the right one is difficult, and for a naive user it's walking through a forest, basically, not knowing where you're going to end up. So making decisions for the user in this case is going to be a critical thing in the success of these packages. They're not guaranteed to converge, and a user just blindly making a choice would have a hard time choosing preconditioners, making the right choice in terms of a method to match their specific problem. So our self-adapting effort tries to understand something about the data that the user has given us, and then make choices based on the user's data, the size of the problem, and the machine it's going to run on, which software package, method, and preconditioners are going to be the best ones, and then make that choice for the user. So I think that that's another area which has a lot of interest today and is right for research projects.

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

HAIGH: This is a resumption of the third session, starting at noon, on the 28th of April 2004. It's again taking place in Professor Dongarra's office on the campus of the University of Tennessee at Knoxville.

As far as I can see we've dealt with the major projects. So what I'd like to do now in the final session is to return to look at your career more generally over the past 15 years within the academic community and as an institution builder and academic entrepreneur here with the centers. So I suppose to start most generally. You moved here from Chicago, where as you said you'd lived all your life, to Tennessee. Were there any kinds of adjustments that you and your family had to make?

DONGARRA: Yes, I would say the move to Tennessee was quite an adjustment for all of us, but I mean that in a positive, growing sort of way. We were very happy living in the Chicago area and loved the city, our home, and our life there. Chicago is a wonderful city with a rich ethnic mix and many things to offer culturally in terms of restaurants, museums, art, sports, and so on .It's very large and easy to find things that you might need. Both my wife and I were Midwesterners all of our lives, so it took us a while to adjust to the more relaxed pace of life in Tennessee. Although we miss the ease of access of things that we had in Chicago, and of course we miss being near family, we have really enjoyed living in East Tennessee. The natural beauty of the area is something that we have grown to love, and we have learned to appreciate the friendly, laid-back nature of the people. Our home is in Oak Ridge, which we have found to be a very fine community in which to raise our three children. All in all, it has been a very positive experience living here. So I would say, as with anything, there are positive and negative aspects of change but overall,, I'd say the positive things are in abundance here

HAIGH: Now when you first arrived here, I think you had mentioned that you'd negotiated and received space for you and one other person, forming the nucleus around which your group grew. What kind of, how would you characterize the state of the computer science department in general at that point?

DONGARRA: Well it had the feel of a relatively new department although the department had actually existed for quite a while, having been spun out of the math department,. It was modest in terms of its size, located in the College of Arts and Sciences here, and it was intended to grow at a certain rate. We never achieved that ultimate growth, although we did add new positions to the department and I think we've made a number of good decisions in terms of hiring people. There are probably fifteen people in our department. All the basic areas of computer science are taught and we have a graduate program so students are receiving masters and Ph.D.'s, Ph.D.'s out of our department itself.

HAIGH: Are there any other faculty members within the department with whom you've collaborated or worked in other ways?

DONGARRA: Oh yeah, absolutely. We have strong collaborations with a number of people within our department, working on a number of areas, including numerical methods systems design, parallel processing and grid computing, so there's been a very active involvement in research and successful collaboration on a number of projects here in the department, and I'll extend that, as well with Oak Ridge National Lab.

HAIGH: And have there been any collaborations with members of other departments within the university?

DONGARRA: Yes. We have an active program with a number of colleagues in other departments including the math department. We run an organization called the Center for Information Technology Research which is intended to provide an interface to information technology research not only here on campus, but across the whole University of Tennessee system. It is a place where people from throughout the UT system can gain help and entry and perhaps assistance in terms of developing a successful research program and acquiring research funding.

HAIGH: So there are two groups here right. There's the Center for Information Technology Research, and there's another center which is, is it called Innovative Computing? I'm afraid I'm not seeing –

DONGARRA: So the confusion I think is that there's my group called ICL, Innovative Computing Laboratory, and the Center for Information Technology Research. ICL is my research group, active in our research program, and consists of about 50 people including research professors, postdocs, research assistants, graduate students, undergraduate students, programmers, an artist, secretarial staff, people who take care of our computers, and our support staff as well. That 50 member group is paid for out of what we would call soft money, so out of that group of 50 people only one person has a job for life: that's me because I'm a tenured professor in the department, so as such I have a job until I quit, die, or do something stupid. The rest of the people are on soft money, so they're paid out of money from grants.

HAIGH: And would you have liked the university to make additional tenure track hires in related areas?

DONGARRA: Oh, sure, yeah. We are always striving to get positions, find people who are at the right level to fill those positions, and then build a program from that.

HAIGH: So let's move back then a bit in time from the current situation time of 50 people, to when you first arrived. Now quite soon afterwards it appears that you got the two grants from NSF and one large grant from DARPA, which you had mentioned, funded what became the LAPACK project –

DONGARRA: That's ScaLAPACK, so DARPA was ScaLAPACK one –

HAIGH: So who funded LAPACK then?

DONGARRA: LAPACK was funded primarily from the National Science Foundation and the Department of Energy.

HAIGH: Now when you came here did you have a reasonable confidence that you'd be able to bring some big grants quickly?

DONGARRA: Well, we always had aspirations and hopes that we could do that. It's a matter of timing, having good ideas, and being able to put together successful proposals. Those are the key factors in this whole mix. We thought we had good ideas, it turned out we were at the right place at the right time in terms of having proposals in an area which was hot, significant funding was available, and we were able to construct a compelling case for that proposal. All those things happened and we were successful indeed. Now did I know that was going to happen? I hoped it was going to happen, but I don't know that I knew the chance of that happening. It was probably a high percent because we had good people, we had good ideas, and we had done all of our homework in trying to get things organized and arranged so that it would work out in our favor.

HAIGH: And after you received those early grants how big did your group become at that point?

DONGARRA: I don't remember the size, what path we were on. Certainly we grew from two, to five, to ten people in a short time, and then we grew over the next few years maybe to 15 and 20 people. We stayed at 20 people for a while, and within the last five years we've expanded up to 50 people, and today I don't think that our group can grow any larger. It's a group that's almost out of control at this point and having it larger would make it it unmanageable. At this point we have an infrastructure in place so that we have people leading teams and groups within teams and a whole organization that tries to make things as efficient as possible in terms of running the programs and getting the results and doing all the other things that you need to do to be a successful research program.

HAIGH: Oh sure. And with 50 people on the payroll it must be larger than the computer science department itself.

DONGARRA: Yeah, yeah. I think we bring in the most money of any group on campus, so we're larger than physics, chemistry and so on. The only group that is larger in the UT system, and brings in more money, is the medical school in Memphis.

HAIGH: Did you have trouble getting space on campus?

DONGARRA: Oh sure, we always have trouble because we always want more space. Space is a commodity that is in great demand and one of the issues is that if you have research funding, space should be provided for you to do your business. We never have enough space so we're constantly putting pressure on the administration to provide additional space in some form. We're stressed whenever we have visitors, especially long term visitors, who need space to do their work, so we generally have to double up or find some other creative solutions. So, yes, we're always looking for additional space.

HAIGH: Would you say you've been generally satisfied with the relationship you've had with the university?

DONGARRA: Oh, sure, yeah. As you know, the university reacts to situations: if you make a request they try to satisfy your request. It's clear the more money you bring in the easier time you're going to have making your case and getting it satisfied. As we've been successful, it's been easier for us to be in that position of making a good case and coming out successful in the end.

HAIGH: What do you think the advantages and disadvantages have been of having your group here rather than at a university that might have a larger number of extremely large prominent research groups?

DONGARRA: Right. We're a big fish in a small pond; that would be the expression I want to use. So here there are not a lot of groups that are of our size. If you went to someplace which had a reputation for doing top research, we would be in a mix with a number of other groups competing with them for space and other resources. So perhaps that would be to a different situation.

HAIGH: You have mentioned earlier that it's always been hard to get research funding, to develop mathematical software, per se. You mentioned in the early days that the first projects were actually funded essentially as experiments in software research, and you'd also mentioned that, particularly within mathematics, where you did your Ph.D. work in this area wasn't seen as an intellectual contribution on the same level as proving a new theorem. So would you say that that situation has changed over time, and how have you dealt with it here?

DONGARRA: Well, it's changed a little bit. It's gotten easier to some extent, but it is always hard to justify a research project that's going to be involved in developing software, and then it's harder to get funds to take software that has been developed under a research program and harden it so it can be used by a wider audience. And then it's been difficult to get funds to take that hardened software and move it into a form where it really can be embraced by the community and used by everybody.

There's a path of sorts for software, from a research component out to a commercial end, and as you move towards that commercial side it becomes more and more difficult to attract research money. It becomes a challenge to get funds to develop the software, to harden the software, and then to maintain the software over the lifetime of its existence. That's the struggle, and it is particularly true in our case, because we have developed software which has a life associated with it. The software was designed and put in place under a research program and there's no mechanism to provide for long term support of that software, and that perhaps is what's missing from our community.

HAIGH: So leaving aside the support issue, you have been remarkably successful in attracting grants to support this work. Is that because you've been able to persuade people that mathematical software is important or is it because you've been good at convincing people that this project that you want to do is related to whatever the priorities for research funding happens to be that year?

DONGARRA: I would say we're good at putting together a case that the research area is important, and that the outcome of that research will be software which will benefit a large group of people; then we've been successful in developing software that fits that story. So not only have we put together a story, we've delivered on that story, and delivering on a story, of course, makes it easier the next time you go and ask for that kind of support.

HAIGH: So the track record helps?

DONGARRA: Oh, it does help, indeed.

HAIGH: Now looking at the list of grants, I also see that DARPA/ARO remains a major part of funding over the years, there's a string of NSF grants, and there's also some grants from other places, such as the DOD/Nichols Research Corporation, and a few from computer vendors. How would you categorize differences in dealing with these different kinds of funding sources, do you approach them all the same way or do you have to tailor the approach depending on what kind of group you're dealing with?

DONGARRA: It's important with a group of our size that there is a portfolio of efforts under way so that if funding is lost in one area, for whatever reason, we can continue with other areas. And different agencies have different ways of doing business in terms of how they solicit proposals, how the proposals are judged and ranked, how the money is actually given and what reporting is required after the funds are delivered. It's easy to write a proposal for some agencies; all the work is done before the proposal is written convincing an agency that this is going to be important stuff, and then the proposal is a formality after they have been convinced, and then you receive the money. On the other side, there are agencies where you submit a proposal, and that proposal gets reviewed by the peers and ranked and only the best proposals are chosen. So there you don't do your work up front, you have to develop a story within the proposal that is compelling and interesting from the research side, so that your peers will decide that this is a very attractive proposal and will lead to new research. The process of reporting what you've done, how you've done it, and what goes on after you get the money varies quite a bit from one agency to another.

Some of our grants are not grants but contracts. With a grant, you say you're going to do this work and you start off on a course of doing it and you may or may not end up with results at the end which match what you said you were going to do. A contract is more of a formal thing where you say you're going to do this work, and you may be doing it in terms of developing something for a specific project, and you have to deliver on those items with very definite milestones, and deliverables; if you miss some of those things, or if you miss too many of those, they can stop your contract and suspend your work at that point. It's a case of being held very closely to what you said you would do. A grant to an agency, on the other hand, is a research project: I don't know how it's going to turn out; I think I know, but I don't know for sure, and I will certainly live up to whatever we said we would do initially, but it may not turn out to be such a great thing in the end (or it may turn out to be better than we said it was going to be).

HAIGH: And would I be right to think that NSF would fall more into the second group you mentioned, of peer reviewed, very detailed proposals, and that DARPA would be more in the first group of somewhat less formal?

DONGARRA: Yes, that's very correct, in the terms of how things work. NSF has a very rigorous peer review, so even if you convince a program manager he may not be able to fund that because of the reviewers of that proposal, and the way the proposal was written.

HAIGH: And can you give an example of contracts that you've taken on?

DONGARRA: Well some of the work that we've done for the Department of Defense relates to contract work in terms of providing classes and material for their centers, so we have some work with the defense modernization program which falls under that category. They would like classes on how to use certain kinds of software, and would like software installed on certain machines. We understand enough about that software and have the ability to do that, so it becomes an easy case for us to deliver on those contracts..

HAIGH: Do you think there's anything in your personal background or skills that has made you particularly good at writing grant proposals and soliciting funds?

DONGARRA: I don't think I'm particularly good at writing grant proposals, and wish that I was more skilled in that area. There are usually very talented people, colleagues at various universities, helping us, and I also have people on my staff here who are very good at writing proposals and the mechanics of putting all of that together. What we do have is a certain talent for writing software, for developing software, and we have a certain ability for understanding what's important at the moment and what the sense is in the community of what's needed. So we're able to put together a number of things which are important for software needed on today's high performance equipment, and to be in position to tell a story which can be compelling from that standpoint.

HAIGH: And how would you say your personal role has changed as the group has grown from two people to 50?

DONGARRA: That's always a struggle in terms of maintaining enough balance. A large group of people has to be managed at some level; when you have one person it's a lot easier to manage than if you have 50 people. I don't manage 50 people. I interact with a large group of people, and have people that I interact with do the day to day management, but it is a struggle to maintain the levels of involvement at a technical and detailed level with all of these things.

HAIGH: Do you still ever write code?

DONGARRA: I don't write code in the same way I did back when I started, so very little code gets written today. Coding is a very labor intensive effort and can be very consuming of your resources in terms of ensuring that all the pieces are managed correctly and come together and are efficient, and so on. So I rarely do code development today, I may play around with an algorithm and implement that algorithm in MATLAB at a high level to understand something about some certain feature, but as for actually implementing the details of FORTRAN or C program, I do very little of that today.

HAIGH: Do you miss it?

DONGARRA: I would say so. It's always a rewarding experience to write a certain piece of software and to get it working, especially if that software has some unique features. It was always a pleasure to put something together and realize at the end that you had something that nobody else had, that it was able to carry out a certain operation and do it in a certain way which produced better results than anybody else had been able to achieve with their methods and approaches. So, yeah, I would say that's a missing component.

HAIGH: And as the group's grown, you've suggested that there's a shift towards a more managerial kind of role. Is that something that you've found yourself enjoying, do you think it's something that you're good at, or do you delegate a lot of the managerial roles to some kind of executive director or chief of staff type position?

DONGARRA: We have a structure in place where somebody else is given charge over certain areas and they go off and do that. We have meetings to report and talk about the issues. Is it interesting and rewarding to do some of the things we're doing? It certainly is a challenge, so the fact that we've been successful is a reward in itself. That I've been able to take on a certain responsibility, use that responsibility to develop a program, and at the end of that to come out with a very beneficial set of tools that people can use - from algorithms to software, to ideas, to sharing of information - I would say that's all very positive and it's very rewarding to go home at the end of the day realizing that you've helped other people do their job, and to do it effectively.

HAIGH: And your work as a supervisor of students. Now I have a list of the MS and Ph.D. students, so I think the general questions there would be, do you find that not being located in one of the top 20 universities makes it harder to attract the Ph.D. students that you'd like to work with the group. Also, more generally, are there any other students that you think have made particular contributions?

DONGARRA: I would say we have a harder time of attracting the quality of students that they attract at some of the top schools. I visit my colleagues at Stanford or Illinois or Wisconsin or at Washington or San Diego, and see really outstanding students with impressive backgrounds at those places. But I would say that we've been fortunate in terms of the students that we've had here. I've had a really great run of students over the past 15 years. We've had about 20, maybe 30 students, finish with a Ph.D. or masters degree here. Among them have been some outstanding students who have made really significant contributions which have found their way into the fabric of how people are doing computing today, so obviously it's a wonderful experience when that happens. And in so many cases one person can make the difference between a successful project or an unsuccessful project, and having that one person involved in an effort at the right moment is critical in terms of how that project ends up. We've had a great run of those individuals, so it's very positive and rewarding to see that happen and to watch those students evolve over time, build their own programs .

HAIGH: So switching now to consider your broader academic career. You'd earlier identified a couple of key papers from the 1980's. I know a lot of the enormous volume of things that you've published and presented since 1990 have been concerned with the project you'd already spoken about. But I was wondering are there any papers you think stand out as intellectual contributions from the last 15 years?

DONGARRA: Last 15 years, wow. It's always hard to point to specific things but in terms of work that has had the most impact on people, colleagues, and so on, things like PVM, which everybody has used over time for message passing, and MPI, another very critical development in the ways in which we do parallel processing, both come to mind. LAPACK and ScaLAPACK are also very influential packages that have had major impact in terms of the use of mathematical software. Many of the papers, and especially the users' guides, have been very well received. The project I mentioned called ATLAS has been a real cornerstone of how effective numerical software could be written based on self adaptation, and that's another very important paper.

HAIGH: Alright. Another area that you've been very active over the last 15 years, is in organizing conferences and editing proceedings. So what's attracted you to spend so much time and energy in this area?

DONGARRA: Well in a certain sense, it's trying to give back to the community what I've been fortunate to take away from it. I find myself in a position where I've benefited in a very substantial way because of my publications getting out there and in the hands of other people. I always feel like I'd like to give that back to the community, give back those things in the sense of being able to help other people in trying to achieve the same kinds of things. So putting on conferences and organizing things and being editor of various journals, are ways that I feel I can give some of that back to the community.

HAIGH: And are there any conferences you've been involved with that you think you've had a particularly important effect?

DONGARRA: I've been involved in a number of conferences of course, and they all have slightly different characteristics. There's a conference called Super Computing Conference, held every year in November. It brings together the super computing community, and I've been involved in that conference from the beginning at some level. I helped in terms of forming it initially, back in the 80's. Today I'm one of the co-chairs for technical programs for the meeting that will take place in November of this year. I was involved with the starting up of a conference called The International Conference of Computational Science, and that meeting is being held in Krakow, Poland, this summer, and will involve four or five hundred people. They had to close the registration early because they had too many people signing up for the conference. It will have four volumes in the Springer-Verlag series collection of computer science. I’ve been involved in SIAM conferences from the beginning, some of the early ones in terms of parallel processing. These are all very interesting conferences and it’s exciting to see the work that people are doing, especially the work that young people are doing in terms of carrying out some of the ideas that were laid down many years ago by my colleagues and myself when we started out.

HAIGH: So the international conference series you’ve mentioned has obviously been very successful. What niche did it fill, what did it do that wasn’t being adequately done by existing conferences?

DONGARRA: The International Conference for Computational Science fills a niche for people interested in high performance computing who were involved in this interdisciplinary area of computational science. We’re involved in the Super Computing Conference because it is the premier conference for supercomputing, not necessarily at the level of computational science only, but also looking at various areas of computing in general: at storage and networking, grids, and other things. We’re involved in smaller conferences that fill the niche regarding message passing and how to expose research ideas that are in that area as well.

HAIGH: Alright. And you’ve been for a while, and still are, editor in chief of the *International Journal of High Performance Computing Applications* from 1992 onwards, and of the *SIAM Series on Software Environments and Tools for Scientific Computing* from 1994 to present. What led you to feel, and to persuade other people, presumably, that those two areas deserved respectively a journal and a book series.

DONGARRA: The journal on high performance scientific computations is filling the gap between the more numerical analysis kinds of journals and the journals that deal with parallel processing specifically. The SIAM book series is trying to look at areas of software and tools and environments that are useful and influential. I felt that there was a need, within SIAM, to put together a series that talked about the software aspects of things, and looked at some of those areas. Some of our books and users’ guides have appeared there. We have a book which is an interesting idea, a sort of cross between taking the algorithms and actually developing the software; it’s an idea we had called templates, templates for linear systems. The idea is that in the book enough detail is described so that somebody could write software, but the software is not actually written in the book itself. The idea is to expose state of the art techniques and describe the techniques at an adequate level to give enough of a background so someone can take these techniques and embody them in software. So it’s sort of a primer maybe for the software that could be written.

HAIGH: Actually picking up of a slightly related question that I realize I didn’t ask earlier. You’ve been interested in a lot of time of producing user guides to go with software, once you'd find yourself in an academic environment did that translate into anything distinctive in your teaching approach?

DONGARRA: I'm not sure. We certainly use software and try to expose software to the students throughout the course that I might teach. The idea is to make people aware of the software, some of the performance issues, and about trying to write things in a portable way, so perhaps some of those ideas rub off on the students. The techniques and features that we try to expose in our software have been provided to the students.

HAIGH: And you're also on the editorial board of a very large number of publications. Are there any of those where you think you've had a particularly active or influential role in shaping the culture of the publication?

DONGARRA: Oh, I don't know that that's true. I certainly try to do my part and to, again, give something back to the community in terms of what we're doing here. I don't know that I've tried to influence or push certain journals in certain directions, as it's really up to the editorial guy and chief.

HAIGH: And you served as a member of the ACM SIGNUM Board of Directors from 1985 to 1989. What was the group like at that point and what was your involvement within it?

DONGARRA: I don't remember much about it, to be honest. I don't know that I had any great insight or impact.. SIGNUM was an organization, but I don't think they exist anymore.

HAIGH: No, they folded. I think it was disbanded around about 1999.

DONGARRA: It disbanded and that was perhaps a mistake, but any way, it was an organization that was intended to be this crossing point between the numerical people, the software people, and others, and relate ideas to them.

HAIGH: Okay. And you also served as a member to the SIAM council, from 1985 to 1991, and as chairman of the SIAM's SIG on super computing from 1985 to 1988. Do you have anything to say about your involvement with SIAM?

DONGARRA: SIAM is a wonderful organization. I would say SIAM is my premier organization, so if I had to list organizations in terms of the those that closely match my background and ideas, SIAM would be the first one on that list. I am a member of SIAM, and ACM and IEEE as well, but SIAM is the one I would describe as my closest organization, and that's why I was involved in those activities. It's a case of trying to give something back to the organization, trying to help in any way I can to provide assistance or help lead visions within that organization. And, you know, I'm delighted that I was able to do that, and I hope I helped in some small way to accomplish that.

HAIGH: So what is it about SIAM that makes you feel most at home there?

DONGARRA: I guess it's a number of things. The area of applied mathematics is something that's very appealing and attractive and really is the center for my colleagues in the area of numerical methods. The size of the organization is not too large, it's not terribly small: it's the right size, and perhaps the informality of the organization is appealing, as well.

HAIGH: So moving now to wrap up, and obviously if there are any things that I don't ask, please feel free to volunteer them. But it's occurred to me listening, that there's a sense in which overcoming difficulties has been a theme. You started off talking about growing up with dyslexia, having a family where further education hadn't been a tradition, overcoming limited financial resources, then the first university you attended was not an elite school, and obviously you've overcome all these things. Now, looking at your resume I have to say it's the largest vita I've ever seen, fifty something pages, you're still going if you continue working hard, you know, it could double in size over the remainder of your career. Do you think that's a sense in which you've been driven to be so enormously productive, sit on so many editorial boards, organize so many conferences, almost in reaction to these earlier difficulties?

DONGARRA: I don't think so. There certainly wasn't any conscious attempt at doing that. I think I found myself at the right place at the right time for many projects and efforts in which I've engaged. Maybe I'm more of an opportunist than others, and I'll use that in a positive way, not in a negative way. That is, I see an opportunity and realize, gee, I could do that; that's something for which we have the ability and the techniques and the opportunity to help. We go for it, and we extend our group and our vision and our capability to accommodate that. I probably have a hard time saying no to certain things, and that's also led to some of that activity that you see here, but, in general, I do things because I think we can improve the situation, I think we'll have fun doing it, and overall we're going to make a contribution in a positive way that perhaps others couldn't, because we have a great set of talent.

I would say that I've been fortunate to work with very giving people as well. Almost everybody who has been formative in my career has been willing to work and to provide a little bit extra in my case, and I'm not sure why, but they were willing to be tolerant of my inability to do certain things and to help in terms of providing me with more that I'd be able to understand, or that I could grasp. That extra attention has been a very positive thing in my whole development. The fact that these people took the time and explained things in a different way so that I could understand the important aspects of what was being described, was critical when I started out in the field, and remains critical today. I think about my initial interactions with Brian Smith at Argonne, then with Cleve Moler as time went on. Those individuals have been tremendous, positive influences in my career, as have my interactions with Jim Wilkinson, Gene Golub, and others. I view them as my mentors and still see them in that way today in terms of my life and my career.

HAIGH: So would you describe the numerical analysis and mathematical software communities as being unusually open and supportive places then?

DONGARRA: So I would say yes, but, you know, it depends who you're working with. So it's a question of the area and individuals, of course, that make all the difference and having the right group of people is going to be critical to whatever gets done.

HAIGH: Sure. Now looking back over your career so far, if you had to single out one specific achievement that you're proudest of or you think that has been the most important contribution, what do you think it would be?

DONGARRA: One single thing, it's hard to point to one thing. If I think about software the most used pieces of software that we have would be with LAPACK. If I think about parallel processing, PVM and MPI would the useful projects. In terms of other aspects, just having these BLAS routines there would be a tremendous aid, and ATLAS is something at the forefront of all of this work today. So there are a number of things and I find myself fortunate to be able to have made a contribution. Understand that I'm not the only person who worked on these projects, that they were developed by a team of people; having that team accomplish what it has and be successful has been very rewarding.

HAIGH: Right. Actually not that I would dream of suggesting this is the most important achievement, but it does occur to me to ask whether EISPACK was the first, you know, was the beginning of this PACK naming convention.

DONGARRA: Oh, yeah, absolutely. That had a tremendous impact in terms of the community and in getting mathematical software off on a foot that exposed things of the nature of high quality performance and portability as being very important for numerical software. It sort ofset the standard for these things, raised the bar, if you will, and is something which is still seen today as having done that.

HAIGH: And do you know if it was the first package to have PACK in the name?

DONGARRA: Yeah, that's right. So I would say if it wasn't, it was either FUNPACK, and I don't remember which one, maybe EISPACK was the first one, so FUNPACK was built by Jim Cody about the same time.

HAIGH: Alright. But you wouldn't know who'd come up with the name?

DONGARRA: PACK, I'm not sure, that's right.

HAIGH: I hope to talk to Cody and I'll ask him –

DONGARRA: Talk to Jim Cody, Jim Cody may know, he's got a better scope of that.

HAIGH: So anyway, the flip side of the last-but-one question, is there anything that you look back on and regret, or you wish that you'd have known earlier or done differently?

DONGARRA: Done differently, or regret? I don't have any regrets along these lines. We've done many, many things, and I would have done them maybe a little differently, but I don't know that I would have done things much differently. So I don't know that I have many regrets.

HAIGH: And are there any particular plans or hopes that you have for your career in the future?

DONGARRA: We're going to continue to do the kinds of things we're doing today as far as I can tell. I don't see any major changes or shifts that would occur in terms of how we do business. So I would say things will be more or less continuing. Throughout the course of my career there's been a thread and that thread has been looking at numerical linear algebra software and trying to understand how to write that and expose it to a large group of people. So early, that was EISPACK and LINPACK; then it was things like the BLAS, LAPACK and ScaLAPACK, and during the course of developing those packages we needed other things that weren't available to us, and because they weren't available we created them: things like parallel processing, PVM, and MPI, ATLAS, and things like, another project we didn't talk about, PAPI, which is an interface that gives you very precise information about the software that's running on a machine in hopes of improving that software. So we've had this thread of developing software for linear algebra and other things were necessary and because they weren't available we helped to develop them.

HAIGH: And it's been driven primarily by changes in computer architecture?

DONGARRA: Changes in computer architecture, absolutely. That's been the motivator for the enhancements.

HAIGH: Are there any current changes or changes that you foresee that would require further generations of software?

DONGARRA: Absolutely, yeah. There are many challenges to us, many unanswered questions, primarily dealing today with the large numbers of processors. Fault tolerance is going to be a big issue, and developing that within the software and coming up with algorithms that can sustain faults will be critical for all these things, so we're looking at that for the future. Self adapting to the environment will be another big area.

HAIGH: I wouldn't necessarily want to use the word competitors, but within other universities or national labs and other publicly funded institutions, any other groups of comparable scale working on these kinds of topics?

DONGARRA: Not scale wise, and, you know, my feeling is that if there's a group out there doing something, and they're being very successful, I don't want to compete: I want them to join us, or we'll join them. So, you know, you can fight them, join them, or ignore them, and I would choose to try to join them.

HAIGH: Well that's it for my questions. Do you have anything that you'd like to add?

DONGARRA: Not at this point, I guess I feel exhausted.

HAIGH: Well thank you very much for finding the time then, and participating over three days.

DONGARRA: Very good.