Proceedings of the Conference on the Validation and Distribution of Computer Software

March 30-31, 1972
Boulder, Colorado*

edited by

Professor Lloyd D. Fosdick
Department of Computer Science
University of Colorado
Boulder, Colorado

Report #CU-CS-004-72

August 1972

* work supported in part by NSF Grants No. GJ 32760 and No. GJ 31681.

Proceedings of the Conference on the

Validation and Distribution of Computer Software

March 30-31, 1972
Boulder, Colorado

edited by

Professor Lloyd D. Fosdick

Department of Computer Science
University of Colorado
Boulder, Colorado

Introductory Remarks - Gordan Sherman, National Science Foundation

As you know, the National Science Foundation has current interest in supporting certain worthy things in the area of computing, including networking and computer aided instruction. One of the areas which is important and we would like to do something about is the quality of software available to research people in the country. By quality of software, we mean accuracy, robustness, wide ranges for input parameters, ease of maintenance, availability, and portability. There have been several grants made in this area over the past year. Many of these are fairly recent. Different philosophies are represented in the projects the NSF has supported, and in order to get the maximum benefit out of the money spent in this area, we felt it would be a good idea to get you people together who do have a common interest to discuss your particular interests, what you propose to do, progress you have made, problems you have come up against, perhaps helping some of the others who are here with the problems they have come up against that you may know something about. It isn't clear what can be done with improving software. It may be too early in the game to do an awful lot. The quality I would like to see may not be in the nature of the beast. We're going to try to find out.

An Advanced Computer System for Social Science Research - Dr. Frederick Thompson, California Institute of Technology, Pasadena, California

My purpose obviously is to review essentially the character of the work we are doing so we can have a basis of discussing underlying problems. I am sure that like many of your projects, mine, the REL project, standing for Rapidly Extensible Language, is probably new to you. In the short time that I have to do so, I will try to cover five different questions. What is REL? Who are the users of the system, who do we anticipate will use it? What is our time schedule, where are we in the development of our work? What are our problems of validity and distribution as a producer of a system? Finally, what are our problems of validity and distribution?

The objective of REL is to produce a system which handles very, very high level languages, where we think of Fortran and Basic as very low level languages. So, what we're speaking of is languages which are natural in the sense of natural English, or natural in the sense of mathematical languages used by mathematicians, or natural in the sense of a language for a composer or an artist for him to express himself facilely to a computer. In the REL system we seek extensibility at two levels. First, all of the languages evolved are typically extensible in the sense that the user sitting at a console linked to the computer can easily define new terms and new syntax. The users that we have had find this a very natural and typical way to use the system, the vocabulary of the system rising right on top of the conceptual development as they deal with their data or their environment. Second, we want to language capability extensible in that it will be easy to implement a very high level language. We are looking at the type of thing where we could implement a very, very high level language for a new user in the matter of a week or so. The emphasis in the system is to provide a highly idiosyncratic capability, tailored and built around the efficiencies of the underlying logic of the particular application, and thus provide a language capability that is very natural for those who use it. Procedural languages do not fit well into our scheme.

Let me give you an idea of the system architecture. We are at this time operating in a 360/370 environment, and we are operating on top of OS 360. We were in operation with an on-line interactive multi-programming, multi-processing system in the summer of 1970. In order to get the efficiency we needed, we were on a 360/50 and we programmed at the machine level, way below

OS 360. Our installation at the California Institute of Technology, like many university installations, had to be cut back because of the reduction in federal support of science and research. Consequently, they got rid of the 360/50 and with that four years of programming was essentially wiped out. We then had to regroup and find out where we were going. We will be getting a 370/130 in July of this year, and we will be back in business then. This time we are not going to be below OS 360, we are on top of OS 360. That poses a lot of problems, but even there one can do a lot. On top of OS 360 we have a general operating system, our own time sharing system, and our atlantic processor. Our system has a single language processor, an extremely powerful and efficient one, and it is important for us that it works very closely with the operating system, because the large data bases addressed exceed the core storage capacity.

We have what we call language data base packages. In our system the user's language and his data become one, so that the total package is the important thing. The system supports as many of these data base packages as you please using a 2314 or 3330 disk system. It does not operate with other things in the computer.

Another concept I would like to introduce is called the base language. It has the syntax and powerful language capability but essentially no vocabulary except the little words. For example, REL English has such words as what, and, of, and things of this sort. One might augment REL English with a second base language for statistical use and it would have such words as correlation and things like this built in, but it would not have any data and would not have any reference words like man, person, uncle, John. The user then builds on top of the base language his own vocabulary or data base. The building upon may also include building in primitive algorithms on top of the syntax so that he has powerful algorithms under him, idiosyncratic of his particular area of discourse. Another base language typical for us would be the animated film language which we developed in conjunction with our computer artist, John Whitney. With this you can sit at the display and communicate with the graphics display in a language tailor-built for the computer artist. Once you get the composition of moving forms that you want, you wheel in the camera and it automatically animates it for you. Another base language that one of my graduate students has

just completed for his thesis is a numerical analysis language, where one can state typical differential equations and matrices and things of this sort in the way the mathematician normally states them.

The final concept to talk about is the underlying system. We expect and are implementing another language of a different sort called the language writers' language, which is an REL language. It is essentially similar for example, to the work of Standish, Cheetam, and Wichright at Harvard, in the sense that it is a programmers' extensible language. These other languages do not usually accommodate the declaration of new data structures. Our language will be one where new data structures syntax can be declared. Its purpose is to write REL languages.

Now to our problems and progress. We have the parsing algorithm down, the English is coming along extremely well. Its a very powerful English capability. We have a very fine grammar so that it is typically a normal English construction. Other language developments are coming along well. The big problem was the disk access. This is a totally paged system. We cannot use the virtual memory capabilities that you usually associate with paging system for the simple fact we have to keep extremely close control at the algorithm level of page loading. So that is the kind of problem we are dealing with and that is where we are optimizing a great deal down at the language processing and operating system level, and I would say it's probably the main area where we have concentrated our effort in the last year. As well, extensions in all aspects of the systems such as the REL English, I might say for those of you interested, we have a report that gives you a general view of the language and some of the protocols from the summer 70 system, and I have several copies, and would be glad to make it available to you.

We have a BATCH version of our system running in our own BATCH processor at Cal Tech which is an 371/55, so we are in good shape to debug our routines at all levels of our system. We have the design of our interactive system complete. The language processor has been working for about a year. Much of the English is working. The animated film language is working. Many of the interactive underlying pieces are now being debugged. By October 1 we expect to be in good shape in the sense that we should have several data bases of considerable size in there, sufficiently cleanly debugged so we can have a major

demonstration at that time. That will be a major day for us when we have a demonstrable system. Now the next date for us quite frankly is October 1, 1973, and then we hope to have a completely documented prototype system. So we have a year to polish the system, to extend the system, to get a lot of wide-ranging language developments accomplished, to do a great many timing studies. IBM is very much interested in the system and is cooperating with us and giving us a lot of help.

We are committed to Skutter and by November 1 we expect to have his data base back in there and be in operation so he can get back into operation studying the Tonganese. Whitney expects to be back as the artist in residence in the Fall for purposes of carrying on his art work. One of our undergraduates is working with the California Institute of the Arts in conjunction with their composers, in particular Tenney who came from Bell Labs where he worked with Max Mathews on Music V. We are implementing that, putting syntax on that for music composition which we hope to then tie in with computer art which is a fantastically interesting development. In any case, this is where we are shooting for the prototype documented system.

The major users know nothing about computing but want to use the computer. So we hope that it will be an on-line modeling and data base analysis system, if you like, a question and answering type system. This kind of system, the so called relational data system, differs from other relational data systems which are being developed now, for example, Bill Woods at Harvard and Terry Winograd at MIT, Kellogg at SDC, Steve Kohls and Bert Raphael at SRI. We differ in the first place in that we are aiming at an early operational date and we are aiming at getting an operationally balanced system, with good turn around times and large data bases. The second major difference is, of course, extensibility which we feel to be very powerful and very important to us.

We also expect to use this system in a man-machine communications laboratory. This represents another important class of users. We want to put a highly socialized organization like a management team or a military command control team into a laboratory using the computer in three ways: 1) As a simulated environment with actual military data and use the computer and an elaborate simulation to provide that environment to an actual military team in the laboratory; 2) To communicate both with the system and with each other in the REL English language, that is the protocol; 3) As a device that the team itself uses, so it

will be their device in managing and dealing with the task environment. This will give us a capability of really looking at what happens in information processing within a social organization, and we expect to make it an elaborate tool for that purpose. What are our validity and distribution problems as producers? We have not billed ourselves as someone who will give an operational system and have not billed ourselves as someone responsible for maintaining the system. We want it to be transportable and we have several groups interested in it (colleges, etc.), and we want to make it available to them. However, we do not intend then to be responsible for maintaining it. This obviously means that we have to attend to validity and debugging problems. We are not experts. We have much to learn from you people who are working particularly on that problem, and we will be following very closely in what you say here and in your on-going work. We are very much interested in the kinds of packages you produce. We need statistical packages that are well checked and highly optimized outside. In that respect there are a number of projects we are following very closely: the Argonne-Texas-Stanford group and ORISIS, as a source of some of the packages we can use. Our data structures and our paging are highly idiosyncratic and particular to our system. This poses real difficulties in distribution and really makes difficulties in using packaged programs. On the other hand, at the same time, we definitely do not want to get into the business of finding optimal algorithms for correlation, matrix inversion, so we are much more attentive to what you are doing in algorithm development, verification, validation, and documentation than we are in actually picking up bodies of code. At that level we are very hopeful that we can make use of the work others have done in getting really good, tight algorithms that we can then build up as chunks that can be put together and formated in terms of these high level languages.

Questions.

What role will IBM play?

They are very much interested in the whole philosophy of the system. Industry cannot afford to go to highly idiosyncratic languages. We feel our system will allow particular people to use a particular system.

What is implied by "on top of OS 360"?

All of our access goes on top of OS 360 but will run on any system

that has OS 360. REL is a task on top of OS 360. We use the same queueing procedures that are already there as this is a multi-task environment.

Should you allow vague questions and vague answers? Shouldn't the user know something about the machine?

No need to, a particular social scientist defines his terms, his data and his vocabulary. He can find out more about a question or test an answer by rephrasing his questions. The content of the language is up to the user. It would allow the social scientist a chance to deal with large amounts of data (demographic changes, polygamy, etc.) that he could not deal with any other way. This gives the social scientist a capability that he does not have on other available systems. The computer is available to these people in a meaningful way. We are programmed on Assembly Language 360, and constrained to an OS 360 environment, but the system will fit on a 32K mini-computer if you can get peripheral storage costs down. This system will operate in other environments.

What about diagnostics?

This system does not print out a lot of diagnostics. If a person uses this system day in and day out he will know where to look for errors.

Reference:

1. Dostert, Bozena H., REL - An information system for a dynamic environment. REL Report No. 3. December 1971. California Institute of Technology, Pasadena, California 91109.

Documentations to Assist Users of OSIRIS I - A Social Science Software Package,
Mr. Gregory Marks, University of Michigan, Ann Arbor, Michigan

OSIRIS is a package of computer programs developed through a cooperative effort by many groups at the University of Michigan, especially at the Institute for Social Research to serve the archival and teaching functions of the Inter-University Consortium for Political Research (ICPR).

The Consortium was created in 1962 as a new development in academic organization. It is committed to the several objectives of an inter-disciplinary, inter-university research and training facility for the social sciences. These objectives include:

1) creating an archive of multi-purpose data that will serve a variety of research and training needs;
2) developing computer-based systems of data management and statistical analysis designed to maximize the utility for the individual scholar of the data available from the archives;
3) developing training programs shaped to enable scholars to make optimal use of the archival data;
4) providing a staff of professional and technical personnel to achieve these primary objectives;
5) supporting cooperative efforts to expand the total set of resources through the development of other major data archives and centers for training and research.

The Inter-university Consortium for Political Research is a partnership between the Center for Political Studies of the University of Michigan and some 150 universities, colleges, and non-profit research organizations in the United States and abroad. Activities are based on joint decisions made by the Center for Political Studies staff of the Consortium and the Council elected by representatives of the member institutions.

OSIRIS* consists of some fifty computer programs organized in a way which may be traced back to something like BMD. However, there are

*A summary of OSIRIS II (the version now maintained and distributed) and related activities is shown in Figures 1-5 which are copies of slides provided by Gregory Marks. Other supplementary material provided by him is in the Appendix. Two ICPR publications of interest are listed in the references.

considerable data management structures added to it and self-destruct files so that the user no longer worries about format statements or definitions of data, and so forth. The programs are chained together so that one can move standard data sets from program to program, relying on the capabilities of OS 360. We are generalizing this so that we have a monitor which obviates most of the dependence on JCL. The fifty programs are divided up into the various areas. Broadly speaking our goal is to enable the user to move data from his punched cards or from his tape and manipulate it, correct it, display it, and do statistical analysis. You need not be locked into it, languages and routines are provided for getting out of OSIRIS. It's predominantly in FORTRAN, using thirty to forty small to medium size assembly language routines which do manipulating without FORTRAN. We have some PL/I data management routimes. It requires about 100,000 bytes in core, though one of our two programs exceeds that. In general, one of our goals has been to keep the size down to it can run during prime time.

There are conversions now in progress. This is a comment about the difficulties of running on other machines even though OSIRIS is predominantly FORTRAN. The cost to convert to any of the 6000 series or 1100 series runs $20,000-$30,000 in computer time and staff. There is a lot of code in this system. About one-half the cost is in the translation of the assembly language I/O routines and to get use of someone else's system.

The charge for the system is $300 for members, $950 for non-members and $1900 for non-academic non-members. We charge 1/2 price each year after that for updating and so forth.

Let me speak now about documentation and dissemination of code. Essentially what got us into this was we found that a system that worked admirably in Ann Arobr where we had the programming staff, the consultants, and the users, those people you could talk to and get help, was not adequate for users many miles away. We set about trying to develop a more complete set of manuals. We are aiming at quite a variety of things. We have manuals for all levels of users. We set up a small program manual (one page for each program) for the advanced user who did not want to carry around pounds of manuals. What the program does, and so forth are explained. We decided to keep statistical formulas separate. One of the things we have done in this series of documents is try to avoid the flavor of a statistical cookbook. We have simple job printouts as we

found many people do not read manuals, but they look at the printout. We've tried to make error handling relevant not only to OSIRIS but also to I/O tape errors and IBM. We are trying to keep the user of this new system from getting lost. Outside of that we are also supplying manuals that describe subroutines; how OSIRIS is put together, and how to use the subroutines. These are tools for someone who wants to write his own programs. We ship a tape manual when we ship OSIRIS so the user may install it himself, using this as a guide. We also ship a set of test data and results which may be used for validating the implementation on another machine.

We have supplied the software to various groups in Europe and South America. There is not too much you can do about foreign languages. We have had the manuals translated into French, Dutch, Spanish, etc., however, we have no way to check delivery of manuals to see if they say what they are supposed to say, and we don't exactly know what to do about this problem.

When a new tape is prepared for distribution it is necessary to do two things. Members of our staff go to some strange machine and generally we take along some not too experienced member of the staff and ask that person to do the implementation. That is a concrete checkout of all materials on the tape. That's internal, and generally it takes about three tries, then we go to five test sites, friends who we know will try immediately to get the system running. As soon as those five are complete we know that we're ready for release to the world at large. This is a procedure we set up because several years ago, when we were first getting into this, we shipped some pretty flaky systems, and trying to recover from user impressions of bad packages as big as this is very difficult. We are, of course, now learning all the little tricks along the way. We find now that as long as one is running a reasonably standard operating system, there are no problems. In general that is the kind of installation time that is needed for the system.

Now, some comments on what our staff at Ann Arbor has to keep track of. At first, our records were sloppy. Now we are very careful. Our procedures for shipping data and so forth are fairly elaborate, in record keeping. We have to have people on the staff who are able to pick up the phone, and talk to someone who ships printout. We need people who are able to drop other things. We also have to have people who are able to pick up the phone and converse with a user about a problem. We have also found that it is important to check back

with users a number of times. About two weeks after we ship a tape we start checking to see if they have installed OSIRIS and if it is running. We found a couple of years ago that installations were given to one or two people to do and they ran into some problems, either because they found the solution interesting or because they were embarrassed, and they could not get it up immediately, and they did not tell anybody about it. They might work 3-5 months trying to get OSIRIS running. If they had called us we could have solved their problem in a matter of hours or a day or two. So we found that one way to protect our own reputation was to check back. We also periodically go back about once a year to find out what kind of software they are using, how much use OSIRIS is getting and so forth. The use of OSIRIS as we check back is fairly heavily in the research area. We have deliberately not tried to simplify it for educational purposes.

We like to think of ourselves in some ways as enabling the researcher to move from one institution to another. To do that we have data in a standard format. We found that this works particularly well. When we first got into this we were distributing programs that were adequate for researchers who worked with survey data, but we were not getting much in the way of accuracy. Numerical accuracy is now much improved. When there is a user set-up error or a data error OSIRIS makes it plain and clear that this is wrong. We know some other software packages for social scientists are designed to make the user feel better. My impression is that not too many people are that hurt by that "not give them success" approach. We simply let them bomb. One of the things you may detect through this whole tone, one of the things I'm concerned about is not simply getting the software up and then dropping it in their laps. I'm really concerned with how they react to it. There are various levels of difficulty in the various software packages for social scientists, one of which right now is data text. My understanding is that it is distributed without source code, which means one will have trouble modifying the system or adding to it. We have a pretty high degree of collaboration with the people at SPSS. We have provided them with subroutines for their system, which enables them to access OSIRIS data sets. At some point there will be a capability for OSIRIS to read SPSS data sets.

We do surveys of software use and we find a terrific scatteration of use. Predictions are almost impossible. When you find people with lots of use, you find that most of what they are using is home-brewed. In the

160 consortium schools, we find that in half of them the software package that gets the most use is some local package. You can't tell how many of the algorithms come from somplace else, but it's obvious that they are not going to be well-known packages. Another problem which concerns us is how much do you let the user modify the software. With OSIRIS, as soon as the user modifies an existing program he gets in trouble and he calls us. What precisely do we do? We have lost control of what the code looks like. There may be very useful changes. This happened to SPSS.

Charges for software is a fairly delicate area. The National Program Library Service (for social science software at the University of Wisconsin) initially tried to follow the rules of the game, i.e., that software was free or available for the cost of the tape. That just did not work. We originally tried to set out providing software free and found out it did not work. It costs a lot to distribute software and do a decent job of it. We charge a very marginal cost. The consortium budgets a great deal on that basic $300 that we charge a consortium member. We have also found the charges useful in another way. A piece of software like OSIRIS is elaborate and whoever is going to install it had better have some commitment to it. The cost increases this commitment. Another issue here is our attitude toward a commercial user of our software. The question of profits has become particularly touchy in the area of merging with other systems. Xerox Data Systems wanted OSIRIS but they would make a product they would charge for out of it. Our sense was that there should not be a profit made.

Finally, one thing that is of considerable importance to us in terms of long range planning is the role of the computer network. Exactly how does the network change what we should be doing with respect to software distribution. With respect to the consortium, what does it do to change the data distribution. We are going to be one of the first active users of MERIT. MERIT is a network running over telephone lines, between Michigan, Wayne State University, and Michigan State University. We are going to use the network in the context of the 1970 consus data. We expect that there is probably as broad a community of people at Wayne State and Michigan State interested in the 1970 census as anything that we might put into the context of an algorithm. We find so far that the biggest problem is not getting a technically useable network but getting three computer centers to figure out how they trade expenses.

Questions.

Is it more the competence of the user or the back-up he has on his campus that insures success?

In terms of getting the system installed, my impression is that that is pretty heavily the competence of the user. The best situation is if the user has counsellors that he can turn to. There are so many naive people. The most important thing about good help is this allows for failure (failure to train). One has to provide documentation, counsellors, etc. We tried a little bit of this. The consortium has deliberately set this up. There is wide variation. It is not so much the caliber of the computing center as what they are interested in. Where we find people having trouble is at a computing center at a university that is predominantly hard science and engineering oriented. The computing center resources do not go for social science machine time. Or if the computing center is really restrained in resources. OSIRIS up to the present version is pretty awkward. If you want to put the tapes in a standard library, and if it is a fair amount of distance and the social scientist isn't there all the time, it is useless.

Can OSIRIS be self-supporting?

Our impression is that it would survive, but we run into problems with University subscribers for instance, where they would like to continue but have the money reallocated elsewhere. But once they have OSIRIS, it's not like we're offering them a new commodity. We've seen this happen in the consortium in recent years, universities wondering where they're going to get the money.

References:

1. An Introduction to Computing and OSIRIS II, (July 1971). Center for Political Studies Computer Support Group, University of Michigan, Ann Arbor, Michigan.

2. A Guide to Resources and Services of the Inter-University Consortium for Political Research (1971-1972). University of Michigan, Ann Arbor, Michigan.

APPENDIX

Gregory A. Marks
Inter-university Consortium
for Political Research

DOCUMENTATION FOR OSIRIS II

The documentation being completed under our current project will be divided into six separate manuals as follows:

1. A Beginner's Introduction to OSIRIS II
2. A Summary of Program Setups
3. OSIRIS II, A Full Documentation
4. Statistical Formulas
5. Sample Jobs and Printout
6. Error Handling

A Beginner's Introduction to OSIRIS II is meant to provide the inexperienced user with a non-technical description of computers and OSIRIS II. The materials will include:

1. definitions of basic computer and software terminology
2. descriptions of computers, programs, and I/O devices
3. use of a keypunch
4. introduction to basic characteristics of OSIRIS II
5. use of three basic OSIRIS II programs

A Summary of Program Setups is intended for people already familiar with OSIRIS II, who would need only a quick reference to prepare their computer jobs. For each program in OSIRIS II, the volume will contain:

1. order of program control cards
2. content of program control cards
3. logical I/O unit references (e.g., ddnames needed for execution)

OSIRIS II, A Full Documentation provides a thorough description of the system's characteristics and programs. Each program will be described in the following categories:

1. general description
2. uses and functional relationships to other programs
3. extended explanation of program options and features
4. detailed control card descriptions
5. restrictions on use
6. input and output data
7. executing the program
8. references

Statistical Formulas will contain the formulas used by each analysis program to compute statistics which are output. For each formula a reference will be given.

Sample Jobs and Printout will contain a sample printout for each program. Where appropriate, the printout will be annotated with explanations of abbreviations or references to the program parameters responsible for the printout.

Error Handling will contain for each program a complete listing of error comments produced by the program, as well as a selected set of common system and I/O error comments. Where necessary, fuller explanations of the error and corrective actions for the user will be provided.

March 20, 1972

OSIRIS Distribution
Summary for all Versions

The following pages provide a convenient single list of all known OSIRIS installations. OSIRIS is a product of years of development by many people within the Institute for Social Research at the University of Michigan, as well as many friends and colleagues elsewhere. The two left-most columns on the next page indicate the users who have one of the two current versions. OSIRIS II is distributed by the Inter-university Consortium for Political Research, which has placed special emphasis on the usability of the package at non-Michigan installations. OSIRIS/6 contains a smaller number of programs, but several of them are more sophisticated than the OSIRIS II counterparts. OSIRIS/6 is distributed by the Survey Research Center at the Institute. A new version, merging the best features of these two current systems, is now being planned.

In the second two columns are found the installations with slightly older versions of OSIRIS. In some instances we are uncertain whether these are presently functional. This problem is more evident with the last three columns, labelled "original", where information on status is very shaky. Periodic efforts to get user feedback are undertaken for OSIRIS II, but the older installations do not provide much response.

Academic or Government	Current: OSIRIS II Level 2	Current: OSIRIS/5 or 6	Recent: OSIRIS II Level 1	Recent: OSIRIS/4	Original: OSIRIS/3	Original: OSIRIS/2	Original: OSIRIS/1
Alberta, University of, Canada	X						
American University							X-Joint
Amsterdam, University of, The Netherlands	X						
Asociacion Columbia de Facultades de Medicine, Columbia		X					
Atelier Parisien d'Urbanism, France		X					
Auburn University at Montgomery	X						
Australian National University, Australia	X						
Ball State University	X						
Bergen, University of, Norway	X						
Boston College						X-CSF	
Boston University						X-CSF	
Bowling Green State University	X						
British Columbia, University of Canada	X						
Brown University	X						
California, University of, Berkeley						X-Joint[1]	
California, University of, Los Angeles	X						
California, University of, San Francisco						X-CSF	
California, University of, Santa Barbara	X	X					
Carleton University, Canada	X[2]						
Case Western Reserve University			X[3]				
Catholic University, Brazil	X						
CELADE, Chile		X					
Centro Italiano Studie Ricerche, Italy					X-CSF		
Centre d'Etudes Sociologiques, France					X-ICPR		
Chicago, University of	X						
Cincinnati, University of	X						
City University of New York (Hunter College)	X						
COLSISTEMAS, Columbia		X					
Columbia University	X						
Connecticut, University of	X						
Cornell University	X						
Danish National Institute, Denmark		X					
Datum, W. Germany	X						
Department of Health Education and Welfare						X-CSF	
Essex, University of, England	X						

1. CDC 6000's
2. XDS Sigma's
3. Univac 1100's
4. ICL
5. PDP-10
6. Siemens

All others are IBM 360/370 computers

Academic or Government	Current: OSIRIS II Level 2	Current: OSIRIS/5 or 6	Recent: OSIRIS II Level 1	Recent: OSIRIS/4	Original: OSIRIS/3	Original: OSIRIS/2	Original: OSIRIS/1
Florida, University of			X				
Fordham University		X					
Gent, University of, Belgium						X-CSF	
Georgetown University						X-ICPR	
Georgia, University of	X_1						
Goteborgs Stads Servicekontor, Sweden						X-CSF	
Gothenburg University of, Sweden	X	X					
Hacettepe University, Turkey				X			
Harvard University	X_2						
Hawaii, University of					X-ICPR		
Hebrew University, Israel			X				
Howard University	X						
Idaho State University	X						
Illinois, University of at Chicago Circle	X						
Illinois, University of at Urbana	X						
Indiana University	X_1						
Iowa, University of	X						
Johns Hopkins University	X						
Kentucky, University of	X						
Kyota University, Japan			X				
Louisiana State University	X						
Loyola University (Chicago)	X						
McGill University, Canada	X						
McMaster University, Canada			X_1				
Mannheim University of, W. Germany	X_6						
Memphis State University			X				
Miami University (Ohio)	X						
Michigan, University of	X						
Milano, University of, Italy	X						
Minas Gerais, University of, Brazil	X						
Minnesota, University of	X_1						
Mississippi, University of	X						
Missouri, University of	X						
National Academy of Sciences		X					
National Institute of Mental Health					X		
New Hampshire, University of	X						
New Mexico, University of				X			
New York University	X_1						
North Carolina, University of			X				
North Texas State University	X						
Northern Illinois University	X						
Northwestern University			X_1				

Academic or Government	Current: OSIRIS II Level 2	Current: OSIRIS/5 or 6	Recent: OSIRIS II Level 1	Recent: OSIRIS/4	Original: OSIRIS/3	Original: OSIRIS/2	Original: OSIRIS/1
Ohio State University	X						
Oklahoma, University of	X						
Oregon, University of						X-CSF	
Pennsylvania State University							X
Pittsburgh, University of			X_5				
The Population Council, Columbia		X					
Princeton University					X		
Purdue University			X_1				
Queens University, Canada						X-ICPR	
Rochester, University of	X						
Southern California University of	X						
State University of New York at Binghamton	X						
State University of New York at Stony Brook	X			X			
Strathclyde, University of, Scotland			X				
Temple University						X-ICPR$_1$	
Texas Technological University	X						
Tubingen, University of, W. Germany				X			
University College London England					X-CSF		
Vanderbilt University					X-ICPR$_1$		
Vermont, University of					X-CSF		
Washington University (St. Louis)	X						
Washington State University					X-ICPR		
Wayne State University	X						
Western Kentucky University	X						
Western Ontario, University of, Canada				X_5			
Windsor, University of, Canada	X						
Wisconsin, University of at Madison	X_3						
Wisconsin, University of at Milwaukee	X_3						
York University, Canada	X_3						

Business	Current: OSIRIS II Level 2	Current: OSIRIS/5 or 6	Recent: OSIRIS II Level 1	Recent: OSIRIS/4	Original: OSIRIS/3	Original: OSIRIS/2	Original: OSIRIS/1
AB Volvo				X			
Bendix Corporation			X				
Datatab						X-CSF	
Department of Motor Vehicles, California					X-CSF		
Doubleday and Company		X					
Gallup Institute, Norway		X					
Genesco						X-CSF	
General Motors Proving Grounds				X			
Goodbody and Company						X-CSF	
Health Insurance and Resources (a hospital in Canada)					X-CSF		
HumRRo		X					
Merril Lynch		X					
Oxtoby-Smith Incorporated					X-CSF		
The Prudential Insurance Company of America						X-CSF	
Shell Oil Company						X-CSF	
Southern California Gas Company		X					
Spiegel, Incorporated				X			
State Farm Insurance Company		X					
Transaction Technology Incorporated		X					
Column Totals:	61	17	12	8	12	16	2

35 foreign Universities and government installations

3 foreign businesses

38 total foreign

71 U.S. Universities and government installations

16 U.S. businesses

87 total U.S.

125 GRAND TOTAL

OSIRIS II

50 PROGRAMS, CHAIN VIA JCL (MONITOR)

- "CARD" FILE MANAGEMENT
- CREATION AND MANAGEMENT OF STANDARD SELF-DESCRIBED OSIRIS DATASETS
- DATA DISPLAY
- STATISTICAL ANALYSIS

MOSTLY FORTRAN, SOME ASSEMBLER FOR I/O, ETC., SOME PL/1

100K BYTES IBM 360/370 (85 ICPR MEMBERS HAVE THIS CAPACITY)

CONVERSIONS IN PROGRESS: CDC 6000, UNIVAC 1100, XDS SIGMA, PDP-10

$300 ICPR MEMBERS
$950 NON-MEMBERS (ACADEMIC)
$1900 NON-MEMBERS (NON-ACADEMIC)

YEARLY UPDATES, NEWSLETTER AT 1/2 CHARGE

Figure 2

OSIRIS II Documentation Project

We found that what is workable for local users is not enough for the user of the distributed package.

6 Manuals:

1. Beginner's Introduction to OSIRIS II
2. Summary of Program Setups
3. OSIRIS II, A Full Documentation
4. Statistical Formulas
5. Sample Jobs and Printout
6. Error Handling

Materials outside of project:

- Subroutine Manual
- Implementation Manual
- Internal Structure
- Test Job Stream

Considerations:

- Help many different types and levels of users
- Plan for updates
- Allow local variations
- Foreign languages

OSIRIS II Distribution Process

System on tape, plus additional manuals, output sample for test job

Files on tape:

- Instructions & JCL for implementation
- Documentation
- Cataloged Procedures
- Source Code
- Load Modules
- Test Job Stream

System installed by users in 1-2 days, 1-2 hours (360/50) computer time

Ann Arbor staff:

- keeps records on who has each version, updates, status of problems
- counsels users by phone or mail, a mixture of substantive and technical problems
- able to tackle problems immediately, drop other work
- checks back with users to see if they have problems

Fig 3

Figure 4

Some Overall System Goals

Enable people to move themselves or their data from institution to institution without changing to a new software system

Bug-free and accurate programs and documentation

Software screams if there is an error, does not cover and run

The package is open-ended and easy to augment

Ties To Other Work

SPSS

- OSIRIS dataset interface

National Program Library/Central Program Inventory Service

Figure 5

Some Broader Concerns in Software Sharing

- SURVEYS SHOW A SCATTER OF SOFTWARE IN USE, OFTEN HOME-BREW
- USER MODIFICATION OF SOFTWARE; PREDICTABILITY VS. USEFUL CHANGE
- USER REDIFFUSION OF SOFTWARE WITH LOSS OF DIRECT LINES OF HELP AND INFORMATION
- CHARGES FOR SOFTWARE
- COMMERCIAL USE
- PRESSURE FOR MACHINE-INDEPENDENT SOFTWARE VS. EFFICIENCY, INTERACTION
- COMPUTER NETWORKS

Conference on Statistical Computing - Dr. W.J. Dixon, University of California at Los Angeles, Los Angeles, California

There seems to be a question of exportability and how much of what things have gone where. As a statistician I was interested in the frequency which universities use statistical programs. Hugh Cline from the Russell Sage Foundation did a survey on the availability of program packages to social science departments in the country, he also documented how many places use their own little system and how many have consultants using their system. The results are summarized in figure 1.

We felt that it was time for a new generation of statistical programs, far different from most of the systems that exist, because the use of these programs themselves have started to change the statistical field so fast. We asked NSF for some support for a conference to look at two frontiers: where statistics was going and what kind of software needed to be developed. The first conference was on the use of the computer in teaching statistics. David Andrews, a statistician from Toronto and Bell Labs, Alsop from Stanford, Nemeny from Virginia State, Forsythe from UCLA, Postelnicu from the Romanian Academy of Sciences, Hartigan from Yale, Janet Elashoff from Stanford, and Norwick from Iowa were some of the people who attended this conference. One of the main products of that conference was the conclusion that there should be data sets made available which would be effective in trying to introduce the computer in statistics. In the report on this conference we are not only going to publish the individual papers, but also statistical data sets, some from textbooks and some that various people have collected. One of the things which I presented there is the concept of annotated output. We have several consultants go over the output, and their comments are put on as an overlay over the output. This is very helpful as a teaching aid to get students to understand what they should be looking at. One of our goals is to have these things annotated even more intensely. The output applies to a specific test case. The more general a program, the less help you can give the individual on each output that he gets from it. The general feeling at this conference was that if you have one annotated output per program with many different programs, it in essence becomes a computerized textbook. Most of the people who attended this conference were the kind of statisticians who called themselves data analysts.

however, that it is better to use a full-time programmer and a full-time research worker rather than graduate students, because graduate students have other interests, and they are, afterall, in school to be graduate students.

Questions.

What are your thoughts on portability?

It is programmed in FORTRAN IV. We are working on a 360/65.

Are you aiming at portability beyond that particular compiler?

We certainly hope to make it portable and have thought of using ANSI FORTRAN.

Are you planning on testing the program at other centers?

Although we have not yet made the arrangements, we are planning to do this.

I think it is unreasonable to expect that a project with two men on it could produce good, portable, completely debugged software.

Sometimes concern with portability can subvert your whole project. Our project will not have wide portability. I know the people who will use the software, the people who are interested in this kind of thing. And it will be somewhat portable no matter what kind of FORTRAN we use. We don't expect it to have as wide a capability as say OSIRIS.

Variance Component Estimations - Dr. Shayle Searle, Cornell University, Ithaca, New York

My program is miniscule compared to the others discussed here. I will give you a brief, nontechnical outline whereabouts in statistics the particular small aspect aligns itself with what I'm concerned with and briefly comment on the people I have hired and what I hope to do. We are all familiar with what analysis of variance is. It is concerned with what I would call nice data or balanced data. There are hundreds of books that tell us about analysis of data and there are programs that say it is easy to compute, and there is teaching of it in basic methods courses in statistics. Now for the same kind of data the procedures for various component estimation are also well known. They have been written about in books, but in no sense in the detail that analysis of variance has been written about. Unfortunately the teaching of variance components is often overlooked, usually because of a lack of time. Now data comes in two forms: nice data and messy data. Messy data is where we have rows and columns and where we don't have the same number of observations in every cell, where the number of observations can differ enormously, and where there can be a large number of cells with no observations at all. Now in this context analysis of data is pretty well known and written about. But it is certainly more difficult to understand, but not as well written about as analysis of balanced data. Variance components for messy data are also more difficult to compute and the subject is usually only taught to the more advanced student. My project is concerned with variance component estimation for messy data. It is a subject in which I have had a long standing interest as a statistician. It would be nothing to have thousands of rows and thousands of columns, and a million cells. But you may only have 70% of them with any data in them at all. My object is to provide the people who use these statistics with some programs to do the calculations. Then my hope would be to make these programs available not only to the people who have data, but I would also like to use them myself for research.

When I do have a set of programs, I want to use them for simulation of various calculation procedures to make a comparative study. There is a capability now to produce a package to do the whole lot. Some programs are already available and we have some graduate students working on others. I have found,

At the second conference we wanted individuals who are at the forefront of research in this field, and we had a team from Bell Labs who had been working on meteorlogical data, John Tuckey; H.L. Lucas and some of them who worked at North Carolina State on a Mathematical Differential model for various types of data to test the models. Hardigan with his cluster analysis on qualitative data, Ray Mickey from UCLA who has been working on a very heavy data reduction problem, on long term drug effects as they are monitored by an EEG; Jim Dickey from Yale on missing value, gross error and so forth. We also had Alston Householder there to listen to numerical procedures. These problems had in common heavy use of the computer. It was clear from this conference that a new field called data analysis is emerging in which the statistician is a direct colleague of the researcher. The computer in every case was used in a heavy sequential fashion and also used interactively. This was true of the educational groups as well. The statisticians were not working alone as far as they themselves are concerned. In no case did we find a statistician working on a frontier problem working alone as a statistician. There were several statisticians working together in their own area of expertise. We also found that the sophisticated data analyst is very little interested in producing a super program package. They are using great ingenuity in using various subroutines and packages which already exist. At this level he has the capability to reformulate his problems to make use of the existing packages. Even though they are more interested in using these techniques rather than developing new programs, they are nevertheless developing new statistical procedures. We found that the researchers were able to shift very easily from one computing facility to another. That in general meant that they had learned something about operating systems, and were able to use the system effectively. Graphical displays were rated even higher than fast turnaround.

Another purpose of this conference was to give guidance to the next generation of our BMD program. We are about convinced that least squares cannot stand alone, at least in the statistical field, and you can guess this is going to be a major overhaul. Of course, all the people at the conferences were using these packages and they had to keep reminding us that these least squares are biased. Already in use are some programs on graphical use of non-linear discrimination. There is a heavy attack on the assessment of accuracy as a data dependent monitoring activity. Some of the programs have flags which indicate if things are getting into dangerous areas, but there has not really been a large scale attack on this problem.

We are concerned with instructing students in the use of statistical routines. The elementary student wishes to work with interesting as well as simple problems. To this end we are constructing problems with large data files and annotated output. We have a data analysis problem which has a very large file connected with it, approximately a million words. Even those with smaller data files are still quite large - some of the smaller ones in Biology that we have to work with would be something like 20 variables on 200 cases, that would be about as small as anybody would ever work with. The annotated output shows how some analytical person followed a problem through a sequence of program usages. We have used this technique extensively in the graduate courses, and we find the students moving much more rapidly to the computer.

We find that the various aberrations of our subroutines differ from many of the others. It is continual up-grading process, in that when some new development comes out it is picked up by others. I might make an additional comment on algorithms compared to programs. The time is here when algorithms can be refereed, and certified and published in much the same way that mathematical and statistical research can be. I think programs and packages are not yet at that stage. There is no definition for a good program. It is highly dependent on the data, the kind of individual who is going to use it, what its purposes are and so on.

Questions.

Do you think your distribution procedures will change?
When you add more data that complicates the distribution. We have a system much like Thompson described in which our data management is laid over the top of OS 13600. It certainly makes it easier to transport. We have tried to think of a way to transport graphics. We have many different graphical approaches to statistical analysis but we do not see yet any compatability of operating systems which have graphics in them that you can look at. This is where systems development is holding back the statistical field. The statisticians are very graphically minded.

Availability to Social Science Depts.

BMD (UCLA)	277	
SSP (IBM)	179	
OSIRIS (U of Mich)	59	(63)
SPSS (Nat. Op. Res)	58	
Data Text (Harvard)	36	
P-Stat (Princeton	33	
TSAR (N. C.)	27	
MSA	21	

5-15 locations

TROLL GPSS ICES MPS

STATPAC TSP UMST MIDAS

SRC (Berk)

plus 150 others.

From Hugh Cline Russell Sage Foundation

fig. 1
DIXM

Software for Statistical Analysis - Tom Aird, Purdue University, Lafayette, Indiana

On this project I am mainly interested in the numerical analysis aspects. The goal of the project is to develop quality software for statistical analysis, combining the state of the art in numerical analysis with the latest statistical methodology. We are attacking three areas: linear regression, non-linear regression and multivariate analysis. In linear regression you have a numerical problem of solving linear least-squares problems, in non-linear regression you have the non-linear problems and in multivariate problems you have the eigenvalue problems. Purdue University has a Control Data system 6400 and we maintain a library of statistical programs. The bulk of them are binary programs. Some of the programs were developed locally, some of them we don't know where they came from. There is an overlap in these programs. I would guess we have at least five or six linear regression programs. All this overlap causes difficulties for the user, just making the initial decision on which package to use. By and large these programs are of low quality numerically. By that I mean as long as the input is set up correctly the problem is within the scope that the routine can handle, but as soon as the user puts in a problem that is ill-conditioned or misses a parameter - meaningless results are often produced. In linear progression problems for example, a lot of people are unaware of the problem until the sum of the squares turns out to be negative.

The system that we have been working with is SPSS. The reason we don't use OSIRIS is that it will not run on the CDC 6400. The SPSS system runs both on the IBM 360 and the CDC's. We have been running SPSS since September 1971. We find it quite satisfactory. Users have overwhelmingly been in favor of the system when they can use it. It doesn't have the complete set of programs like the BMD series or like OSIRIS. The way we're heading is to expand SPSS or combine OSIRIS and SPSS so that we have a unified system. The advantages for SPSS that I see are first the deck set-up is independent. Once the user reads the SPSS Manual and understands it, he can run his data on the 6000 series or the 360 without any change in set-up, because the SPSS system scans its own cards and determines what to do, independently of the machine. The only thing the user would have to change would be his basic control card. A second advantage is the options that statistics card reading provides. This makes it easy for the user

to select options and also makes it easy for many options as one would have if everything were combined into one program. Thirdly, the data management part of this system is isolated from the data analysis part. SPSS is designed in an overlay structure so that each program is essentially a new overlay. So the system can be expanded without any degradation to the existing programs. The overlays are typically only called once so there is no disk access problem.

For the above reasons we decided to develop quality statistical software within the SPSS system. The first area specifically is linear regression. We propose a three stage program, three different methods for solving the problem. One is just the standard that's built into SPSS right now. It uses a sort of Gauss pivoting technique for solving the normal equations. The second approach is to use the normal equations where the standard deviations are not divided or subtracted out. The third technique involves Bjork's algorithm which was published in 1968. This algorithm is the only algorithm I've run across that will actually produce an accurate solution to any problem. As part of our work we have converted Bjork's algorithm to a FORTRAN subroutine and tested it extensively and found with our test cases that it works very well. We ran into problems with ill-conditioned data that was also rank efficient. The iterative refinement did not always do its job in that case. (Question: Is there a difference in computer time between the simplest method and the more accurate and complex one? Response: Very definitely. That's sort of the reason for having three different techniques. More important than the time involved is the fact that Bjork's algorithm has to have all the data in core at one time. The difference of a few seconds is not that important, but as the size of the problem goes up, Bjork's algorithm becomes worse and worse.) (Question: You said the algorithm didn't work satisfactorily when you had a test case with repeated columns. What do you consider satisfactory performance? And what did the Program do? Response: The Program did not converge. There is no correct solution.) (Comment: But there are many best solutions. Response: Yes, but we look for the one that is closest to the norm. It has two options, one is the basic solution - the other way is to project the best least squares solution.) The user will be able to select any one of the three techniques also we would like to be somewhat automatic, and base a decision on the size of the program and timing studies to select a technique automatically. In case of trouble we would have the basic algorithm switch to one of the other algorithms.

Now that there are three ways to solve the problem, the next thing to do in error analysis is compute a condition number. The next step is to do what we would call a Monte Carlo study where the user indicates to the system in some way the relative precision of data he is inputting. Then one can solve the problem again with random errors in it. This will give the user an indication of the difference between solutions. The third step is to compute rigorous error bounds. Professor Lynch and I have worked on error bounds for linear systems of equations [1]. The user is allowed to specify intervals for the data and rigorous error bounds can be computed. Actually, this is a generalization. You can't of course do error bounds on linear equations and apply them to linear least squares problems without some difficulty because of the special nature of the least squares problems. I think the error bound is sometimes going to be too big, sometimes by a factor of two although that is not too bad when you are doing error analysis. Currently then in this linear regression package we are debugging the three algorithms we have already put into the SPSS system. The error bounds routine has been debugged, the linear system routine is written and being debugged for linear least squares systems. We are using user type data sets to test these algorithms. We have several others we use to test the constraints.

Part of our project is to investigate different techniques for selecting subsets of variables, rather than one at a time, many at a time. There is a program, for example, that does this. It does not have to compute all possible regressions. There is another author who has a program that does all possible regressions. I have heard rumors that all possible regression is faster than the other one up to a certain point, although I don't know where that point is. We have just recently got the program running and we ran it on a ten variable problem, and if found the best subset for 10, 9, 8 all the way down to one. The running time was three seconds. I don't know how many users will be working with problems with more than ten variables. I don't think we could hope to pick the best subset of say 100 variables. The computing time would be astronomical These procedures seem to work up to 25 or 30, and maybe 40. We are talking about the best subset in the sense of least squares. The way we plan to build this in, we would also have the step-wise entry of variables so you could say, include all of these five. Quite often the user has a particular variable or variables that he wants included in the model. The statistics end of it plans to investigate the cost criterion. Quite often in agriculture you have variables

that are easy to measure and cheap, you have others that are very expensive.

Singular value analysis can be used in place of regression analysis, although our feeling there was users want to do the regression analysis and although singular value analysis has its merits, converting to it would be an educational process, probably out of the scope of what we are trying to do here. Non-linear regression is a difficult problem, I don't think there is any straightforward answer. There are hundreds of algorithms that have been published and each one claims to be the best. We are investigating several local algorithms, trying to develop a POLY algorithm approach to this so one can switch from one to another if it fails. We are also investigating techniques for generating starting points.

Questions.

On local accuracy checks, I don't know any way you can rigorously compute error bounds when a user has written a subroutine to find a function. In the linear cases it is a theoretical computation, but in the non-linear cases I think one can only use some common sense checks that should be built into the program that sort of evaluate the function in a neighborhood. Possibly there are other things.

References.

1. Aird, T.J. and Lynch, Robert E.; Accurate Error Bounds for Approximate Solutions of Linear Algebraic Systems; Purdue University Computing Center Report, February 1972.

Research Center for the Development of Computer Technology in Economics and Management Science - Mr. Mark Eisner, National Bureau of Economic Research, Cambridge, Massachusetts.

The National Bureau of Economic Research Computer Research Center for Economic and Management Science is a discipline oriented research center whose goals are to produce new software tools for economics and management science. Basically a lot of quantitative theory has been developed, both in economics and management science, and the theory has gone far ahead of a lot of the computing technology. We have three goals. The first is to provide software that people can use. The second is to permit people to use the software to test the theories. The third is to try to stimulate research in both economics and management science by introducing new tools to people and making these tools available in a comfortable environment.

The center is a spin off of a project that was at MIT for five years called the TROLL Project in which an on-line research environment for econometrics was created. This is a system with a goal of providing a complete laboratory environment for the econometrician so he could sit down and have all the tools he needs to do econometric research at his fingertips, in an interactive environment. That program is now running on an IBM 360/67. We have had a lot of experience with interactive computers and we have tried to use this in an innovative way.

In the present effort we have created a stand-alone application operating system for an IBM 360. We are trying to produce a machine independent operating system so that we can interface easily with the real basic operating system on a computer system, and thus carry our own operating system with us. Our programs are designed to run only on IBM equipment and we have no hopes of ever transferring them to other types of equipment. We made a definite decision on the question of how we were going to disseminate this research: not to sacrifice systems capabilities for portability. We want to provide sophisticated environments for people to do research, and to do that we cannot be restricted to simple FORTRAN shipped around. In conjunction with that decision we gambled in the creation of computer networks and we see that as a primary means of disseminating our software: not the only means but a primary one. There is a good reason for taking that gamble, as it solves many of the problems I have heard discussed here. You get much more active feedback and you have very active

testing. Users (researchers) are involved in the building of the software, they are involved in the testing of it and the same research community gets involved in the use of the software. You get a much more active interface between the research community and the programming system itself. Another aspect of the work that goes on at the center is demonstrated by the fact that we did create an operating system. In general, software that is produced for social science, economics, or management science has a great deal of emphasis on the algorithm and the production of results and not too much emphasis on the support environment. However, I would say of the 100,000 lines that we coded, 80,000 lines are concerned with support activity, data management, a good file system with good interaction, and graphics support on many different devices. Actually the control system that was developed has only two major applications, one is a single equation regression capability - that includes linear and non-linear regression, and the other is simultaneous equations simulation of very large systems. This is what the standard econometrician is doing these days. Those were the two main activities in the program, and yet there are 80,000 lines of code - high level language code - supporting that activity.

We have a large thrust in data analysis, the kind of thing Bill Dixon was talking about using interactive graphics, and we are concentrating on cheap graphic devices, storage tube devices, so that people can work and have the graphic capability. With the new technology coming out on graphic terminals, it's very cheap to run these storage devices. Basically that activity is to introduce people in econometrics and management science to things like data analysis to get them closer to the data and the main thrust there is to change some of the research activities of the econometrician. Another large effort is in mathematical programming. Another area is econometrics. We are now launching an effort to put in full information maximum likelihood, full system regression techniques, where you are estimating not just the parameters, but a full system of equations. We also have an effort involved in using spectral analysis and spectral techniques in regression analysis. There are a number of small projects of this nature going on.

A lot of our technology depends on virtual memory and we are waiting for IBM to announce its virtual memory capability on the 370, which seems to be a sure thing, but they are holding off. So our efforts in OS since it is so variable right now are going quite slowly.

We have an automated system for documenting the code we produce. We have a much smaller maintenance problem than a lot of people who are generating systems that they expect to exist in thousands of locations. But even then we want to have well documented code, so we have developed an automated system which insures consistency of documentation. This means that every programmer uses a program to document his code so we know that we get the same kind of documentation throughout. And certain vital information that is machine readable is entered on every procedure in every program. As far as user documentation and support is concerned we have basically four types of documentation. One is a primer, which is a 30 page document where you can learn how to get on and do simple regressions, for example. Another is a basic tutorial guide, a 250 page document which describes concepts and teaches people how to use the system in terms of the concepts that are in the system. There is an advanced tutorial guide for the sophisticated user. Finally, there is a very simple reference manual which is command and function oriented. It is simply like a list of specifications. The user can get explanations and examples of each command. The primer is sort of a teaching tool. It tells someone how to log in, etc. It is a very simple thing. The reference manual is a very rigorous information guide. We also have a great involvement in interactive computing. We have on-line tutorials as well as these basic reference manuals, and they include simple prompting when the user does not know what the computer expects of him next. Our next level includes command formatting of information so that the user can, without the reference manual, get a good explanation of the command format - what the command expects. Finally, there is a third version which is essentially an on-line reference manual, so the user can get the same kind of reference documentation on line as off line. We also maintain a small consulting staff to help people through problems, mostly through telephone communication. We do not have the problem of worrying about installing the system. The system is currently running at MIT and at Yale on 67's - 360/67's. We are waiting for a network so that lots of people can use it. People in Cambridge certainly have used it, and there are some users coming on board from the Washington area.

Questions.

What about the expense of running computer graphics through a network?

The kind of grpahics that we are talking about is storage tube type of graphics

there is not a dynamic picture. It is a static picture and lines are drawn on a screen and are very low cost - not much more than running a regular terminal. My experience with graphics in economics and management science is that people are satisfied with a lot less sophisticated graphics than one supposes or that they may later on want as they get used to graphics terminals. We can find adequate picture drawing capability at 300 baud, if people are willing to sit and watch the picture.

What are you going to do now that MIT is giving up the project?

We have some funding for computer time, and we are talking to a number of places, including MIT, about providing computer resources for the center.

Appendix

M. Zisman

The NBER's Computer Research Center for Economics and Management Science

by

Edwin Kuh

March 1972

Introduction

The social sciences, particularly economics and management science, have been hampered by their disorganized use of computer capabilities. Massive strides in computer hardware and operating systems were not accompanied by equivalent advances in applications software, though here and there fine programs were created and used. Until recently, computer usage in the social sciences was still in the "do it yourself" phase, where relatively isolated groups of individuals wrote programs to suit their particular needs. This has resulted in the proliferation of small, restricted programs that were too inflexible to adapt to the needs of other users or even to the changing needs of the authors themselves. At the same time, methods in computer science that could greatly increase the scope and availability of computer programs were all but ignored by economists and management scientists.

In response to this situation, five years ago the TROLL research project at MIT began to bridge the gap between computer-systems potential and econometric practice. The product of that research, TROLL/1, is a large interactive system that is just now becoming available to practicing economists. This system has sophisticated data-handling and graphics capabilities, as well as a broad set of estimation and simulation techniques. It, more than anything else, has demonstrated the feasibility and power of systems programming as applied to quantitative economic research.

Given the enormous impact of the computer on the economics profession and the concerns of the NBER with applied economics and its methodology, it was natural for the NBER to propose to the National Science Foundation that a Computer Research Center be established, to apply large-scale

systems programming to the more intractable research needs in quantitative economics and management science. At the Center's inception, John Meyer remarked that:

> ...the Bureau has also undertaken the establishment of a center to explore in depth the computer's role in economic research and management. *The creation of this Computer Research Center in Cambridge, Massachusetts during 1970-71 stands in all probability as the single most important event to have occurred within the Bureau during the last decade.* The new center's research program, as it has emerged from plans formulated over the past summer, will reinforce what is best in the Bureau's empirical tradition. One of the Center's first priorities will be to apply or, more accurately, adapt to economics the tools of a newly emerging field in statistics called data analysis. As its name implies, data analysis is concerned with how the intuitive feel or understanding of data can be systematically acquired and applied to empirical research problems. Thus, much of the early research to be conducted at the Center will focus on improving visual displays in the manipulation of data. The objective will be to bring the economic researcher back into closer contact with his data, something that the computer revolution has unfortunately too often diluted in economic research. In essence, data analysis greatly improves the analytic content of data investigation and manipulation. The nomenclature is different and the techniques often more mathematical, but the concepts would be familiar and congenial to Wesley Claire Mitchell.
>
> In short, the new Computer Center is concerned with some of the most fundamental and also practical aspects of modern empirical research in economics. Since such research is the basic occupation of the Bureau, the new Center's importance, even centrality, to the Bureau's whole program can hardly be overestimated.[1]

The National Science Foundation grant began February 1, 1971, and continues for two years. However, the National Science Board has given

---

[1] J.R. Meyer, "President's Report for the Board of Directors, September 27, 1971," *NBER Newsletter,* February 28, 1972.

approval for a five-year funding period so that continued support from NSF can be anticipated with confidence. The core programming staff transferred to the Center's new quarters from the TROLL project in August 1971, when actual work at the Center commenced. The transition from work on TROLL to new activities at the Center has been gradual and is now largely completed. Since TROLL was a prototype of these new activities in a number of significant respects, the completion of TROLL has provided useful insights into organizational and support activities the Center should pursue. A fundamental difference between the TROLL Project and the Center is that the latter has a much greater research, as distinct from computer-systems, thrust. While considerable new algorithmic research has been incorporated into TROLL, the intent was to embody econometric techniques into an interactive computer environment. Thus, most of the resources of the TROLL project went into system programming. The Center, on the other hand, because it can capitalize on the TROLL operating system, is able to put its major efforts into research and applications programming without neglecting serious system programming work.

The Programming Effort at the Center

One unique feature of the Center is the combination of sophisticated researchers and a highly skilled programming staff. The programming staff is composed of a group of committed professionals who represent many years of experience in the creation of large application-oriented computer programs. In order to provide an effective computer environment, the programming staff is active in all aspects of software development. This development can be divided into three broad categories: the basic operating system, an Application Control Language, and various application

programs themselves. The first two categories provide the basic support which allows new computer methods and programs to be created and executed in an efficient and well organized process. The last and by far the largest activity is the implementation of methods and techniques themselves.

The system programming group is currently implementing a specialized operating system (COS) to take advantage of the discipline-oriented nature of the applications being developed. This operating system provides the functions of input/output control, interrupt handling, and file maintenance as well as interfacing with the computer's operating system. Currently, COS is designed to run under the IBM Control Program for the 360/67. In the near future, COS will also interface with new releases of IBM's standard OS operating system, allowing programs produced at the Center to run on appropriately configured IBM 360 and 370 computers.

The Center has also embarked on some innovative system programming to provide capabilities for a large number of simultaneous users. This programming involves modifications to the IBM Control Program, increasing the efficiency of the 360/67 to a degree not previously attained. A further aspect of this system programming activity is the development and implementation of procedures to maintain, debug, and test the software developed by the Center.

A second major system programming effort will be made to create an Application Control Language. This is a specialized language which greatly facilitates the addition of new applications tasks into the system as a whole. This language will provide essential features to the applications programmers, e.g., the ability to define a command interface, the types of files they wish to create, and special communication or "common" regions. It will also allow modules to be called from within other programs and will have a macro facility to enable programmers and sophisticated users to create their own tasks by combining existing tasks or modules.

In the applications area, the programming staff exhibits a special expertise. All of the programmers are knowledgeable system programmers, capable of applying the most sophisticated programming techniques to the application software they are producing. The staff is growing slowly since we wish to maintain our high standards. Another feature of the applications programming effort is the encouragement given to dialogue between researcher and programmer. This vital communication is given the highest priority and is a direct responsibility of the Technical Director of the Center.

The programming effort at the Center is based on the new technology of virtual memories and interactive systems. We feel that the programs and methods developed here will provide researchers in the field with capabilities that will be valuable for many years to come. However, the programming effort itself must be viewed as a research project in the better utilization of computer resources. We think this effort will result in significant accomplishments in the area of software development.

Mathematical Programming

Present knowledge of mathematical programming is so sophisticated that future progress requires an intimate relationship between the development of mathematical techniques and their implementation in efficient computer systems. The Center has begun assembling an experienced staff of mathematical programming systems scientists, maintaining this staff full-time and providing the research support (working environment, supporting personnel, and availability of large blocks of computing time on a large-scale machine) necessary to produce advanced systems.

Since January 1972, William Orchard-Hays has been at the Center. Over the past twenty years he has been responsible for the design and implementation of several large-scale mathematical programming systems for IBM, Honeywell

and Control Data Corporation. Orchard-Hays has also worked for the Rand Corporation and several management consulting firms. His knowledge of systems programming and experience with practical problems provide the Center an excellent start in this area. Professor Jeremy Shapiro of MIT, an authority on integer programming, will be on leave at the Center in 1972-73, working with Orchard-Hays. The months from January to June will be spent designing the new mathematical programming system; development will begin during the summer. William Northrup will assist in the development of integer programming algorithms. Several more mathematical programming experts and computer programmers will join the Center to help implement this large, complicated effort.

A number of mathematical programming systems have been developed in the past, some of considerable power. It has become clear, however, that the power of mathematical programming systems is not always usable in the context of a total project, either because of system conflicts or because of financial and policy problems. Furthermore, none of the present large mathematical programming systems can be used in a truly interactive sense, that is, from a remote console. It seems desirable to create a mathematical programming system which can be used in interactive or batch mode, or in both modes, and which can be made readily available to research and academic groups. Such a system should have the power of existing systems with respect to problem size, algorithmic repertoire, etc., but would be much more flexible to use, more standardized in conventions, and more adaptable to research in such iterative schemes as mixed integer and nonlinear programming.

Data Analysis

Data analysis reorients empirical research methods by integrating statistical theory more closely with the realities of applied problems. An emphasis on an initial visual and parametric exploration of the data series

is one central aspect of data analysis. This initial exploration helps clarify distribution characteristics, e.g., outliers, skewness and multi-modality. The Center has adapted the capabilities of SNAP-IEDA, a data analysis program prepared at Princeton by Professor David Hoaglin under the direction of Professor John Tukey. This interactive adaptation is being used by Professor Hoaglin and Professor Roy Welsch in courses that they are currently teaching at Harvard and MIT, respectively. The program contains various graphical displays of sample information, a number of functions (including seven probability distributions and Monte Carlo sampling from each), and various data transformations.

The Tukey approach can be beneficially modified in two directions. First, data analysis lends itself naturally to interactive computer programming and we intend to move strongly in that direction. It will be particularly useful to have interactive graphic capabilities, enabling an investigator to delete or add points, sketch lines or curves summarizing data configurations, etc. The second aspect open to improvement is the emphasis that the Tukey approach places on hand processing of a data set. It should be a relatively simple matter to expedite the data analysis process by computerizing a number of steps.

Beginning July 1, 1972, Professor Paul Holland of the Harvard University Statistics Department will join the Center on a full-time basis. He will have major responsibility for the subsequent development of research in data analysis. Hoaglin, Welsch, and Kuh will have a continuing and active research interest in this and other aspects of data analysis.

Since real-world data often violates convenient assumptions about the error processes used in the mathematically-more-tractable statistical models, the development of robust estimators is another integral part of data analysis. Under Tukey's leadership, a group of statisticians at Princeton University

has made an excellent analysis of robust estimators of the mean where the frequency distribution assumes a variety of non-normal forms. Econometricians could benefit greatly if these tools were readily available. Some work is now going on in the area of robust estimation of regression models and the Center will evaluate alternative approaches in promoting research and subsequent dissemination.

One problem that arises frequently in applied econometric work is the stability of the underlying regression regime. Regression coefficients often change from one period to the next. When the break-points in the regime must be estimated along with the different sets of regression coefficients, the problem becomes both more realistic and difficult. Several approaches have been proposed, including work by Richard Quandt and James Durbin. Professor David Belsley of Boston College will work on this and other data analysis problems at the Center during the coming summer.

Cluster analysis is a promising nonparametric way to study multivariate data. Cluster analysis can be thought of as a first cousin to principal components or factory analysis. There are a number of significantly different algorithms for clustering. The Center plans to study these alternatives and determine to what extent these methods can illuminate problems in economics and management science.

Estimation of Equation Systems

Simultaneous equation estimation theory has produced literally dozens of estimators, all having similar asymptotic properties. Existing comparative studies of alternative estimators provide little guidance on which ones are best. From among the plethora of estimators, combinations of two major principles provide four major options: least squares in its limited or full information variant (two- and three-stage least squares, respectively); and

maximum likelihood, either in its limited or full information formulation (designated as LIML and FIML). Most practicing econometricians prefer two-stage least squares on grounds of computational simplicity, since full information methods (even three-stage least squares) require sophisticated computer programming. FIML is far and away the most complicated estimation procedure since it is nonlinear in the coefficients to be estimated even when the structural equations are linear. Thus, few full information estimation programs are available. Where they have been implemented, the choices among them are unclear.

Under the joint direction of Professor Gregory Chow of Princeton University and J. Phillip Cooper of the University of Chicago, the Center has started a small-scale project to create programs for a half-dozen full information estimators. Mr. Cooper has spent three months at the Center working on this and related subjects. The Center can undertake this project at moderate cost because particular modules in TROLL, as well as the programming skills to extend them, are already available. Further, the concurrent work on mathematical programming is beneficial since both tasks require the optimization of functions that are nonlinear or discontinuous in the parameters of large simultaneous systems of equations. Repeated, accurate inversion of large matrices is also of central importance to both.

Spectral Analysis

Spectral analysis can be viewed as one sort of data analytic device that should be available in a complete set of estimation tools. Professor Robert Engle of MIT has designed a set of modules for this analytic process. Because of the existing TROLL system and the simplicity of the modules themselves, these have proven to be easily implemented. Two extensions of the basic program will be completed by summer, both of which utilize spectral

analysis in the more familiar regression context. Of greatest novelty and potential importance is the use of spectral analysis to decompose a time series into components of different frequency. A fraction of the original series, purged of certain components, can then be used as most germane to the subject at hand. For instance, Engle with Duncan Foley, also of MIT, are studying aggregate investment behavior where only the low-frequency component of stock prices is a strategic variable. Engle has developed the estimator and its sampling properties. A second regression application is generalized least squares, where the spectrum from regression-equation residuals (based only on exogenous right-hand-side variables) provides the basis for generalized least squares for any linear autoregressive or moving average process, a generality missing in most standard first- or second-order autoregressive transformations.

The Support Staff

The Support Staff at the Center is responsible for system dissemination and assurance. In this context, dissemination means recruiting and training users of the Center's systems and providing appropriate documentation. Also implicit in the dissemination activity is the gathering and channeling of feedback from users to system designers and programmers. The system-assurance function involves coordinating the release of system revisions with appropriate testing and documentation updating.

The focus of our initial efforts in these areas has been TROLL. TROLL's operating system is a prototype of the Center Operating System (COS) that is now being developed; the relationship is especially close in the case of user-visible support facilities such as the shared file system, generalized graphics facilities, and on-line user prompting and documentation. Further, TROLL's current application subsystems - regression, simulation, and a basic data

analysis/transformation facility - will be transported into the COS environment for use in parallel with the new research subsystems developed at the Center. In general then, TROLL has given the Support Staff a good preview of the patterns of system-user-Center interaction which will be encountered over the life of the Center.

Our priorities and progress to date have been shaped by three major problems of an essentailly start-up nature. First, there was a large backlog of TROLL user documentation to be written. Second, although two Support Staff members were incumbents of the old MIT TROLL project, other people had to be recruited and trained to complement their efforts. Third, systems and procedures for maintenance-related problems (e.g., technical documentation, bug reporting, system testing) had to be developed.

In the user-documentation area, a researcher's overview of TROLL, a brief primer, and a lengthy tutorial guide to the system's capabilities were written. A system reference manual for TROLL is being published in installments. Several levels of on-line documentation (i.e., documentation available to the user while working with the system) are being created, including both concise command-format summaries and fuller narrative explanations of each command.

Rapid completion of an initial version of the Center's data-analysis package allowed us to experiment with a collaborative approach to user documentation of new software. In this approach, Support Staff members served as editorial middle-men between programmers who wrote initial low-level user specifications, and researchers who, having designed the new routines, were able to draft explanations of their application to specified problems.

In our dissemination work, we have experimented with a variety of training, consultation, and promotional methods. Several Support Staff members conducted a series of six half-day introductory seminars on TROLL at various institutions throughout New England. These were attended by researchers from about twenty

analysis/transformation facility - will be transported into the COS environment for use in parallel with the new research subsystems developed at the Center. In general then, TROLL has given the Support Staff a good preview of the patterns of system-user-Center interaction which will be encountered over the life of the Center.

Our priorities and progress to date have been shaped by three major problems of an essentailly start-up nature. First, there was a large backlog of TROLL user documentation to be written. Second, although two Support Staff members were incumbents of the old MIT TROLL project, other people had to be recruited and trained to complement their efforts. Third, systems and procedures for maintenance-related problems (e.g., technical documentation, bug reporting, system testing) had to be developed.

In the user-documentation area, a researcher's overview of TROLL, a brief primer, and a lengthy tutorial guide to the system's capabilities were written. A system reference manual for TROLL is being published in installments. Several levels of on-line documentation (i.e., documentation available to the user while working with the system) are being created, including both concise command-format summaries and fuller narrative explanations of each command.

Rapid completion of an initial version of the Center's data-analysis package allowed us to experiment with a collaborative approach to user documentation of new software. In this approach, Support Staff members served as editorial middle-men between programmers who wrote initial low-level user specifications, and researchers who, having designed the new routines, were able to draft explanations of their application to specified problems.

In our dissemination work, we have experimented with a variety of training, consultation, and promotional methods. Several Support Staff members conducted a series of six half-day introductory seminars on TROLL at various institutions throughout New England. These were attended by researchers from about twenty

of the smaller New England colleges and universities. Intensive workshops, from two days to two weeks in length, have been presented, or are being planned, for major user communities, including MIT, New Haven (Yale/NBER), New York (NBER), and Washington. These more or less formal educational efforts are complemented by direct, on-going consultation with individual users of TROLL. In support of the dissemination effort, articles about COS/TROLL and the Center were placed in the Bureau's *Annuals of Economic and Social Measurement*, in MIT's *Technology Review*, and in *Computerworld*.

In the system-assurance effort, procedures for keeping track of software changes were developed and implemented. Procedures for testing software changes were also developed. Other Support Staff activities have included the establishment of a reference library to serve the Center's research projects, and the preparation of standards for technical documentation of software.

National Center for Computer Based Language Research - Dr. Sally Yeates Sedelow, Dr. Walter A. Sedelow, University of Kansas, Lawrence, Kansas

At the outset I want to make the disclaimer that some of the other participants have made and that is I want to make it clear that the focus of this conference, validation and distribution of software, is somewhat beyond the present focus of the study in which Walter (Sedelow) and I are engaged. Our efforts thus far are concentrated upon identifying and possible applications areas and users of a possible national center or network for computer based language analysis. Also we have been concerned with identifying some of the gross technological prospects and problems entailed by the development of such a center or network. We will obviously be concerned with software validation and distribution, but we are not yet at the point of having given detailed attention to these matters. So although I hope my remarks won't be so scattered as to be desultory, I am not in a position to provide you with a tightly structured commentary on the implications of the conclusions (yet to be made) of our study for software validation and distribution.

What we are looking into then is a question of the desirability and feasibility of a national center of undetermined size or network of undetermined size for computer based language research. For the purposes of this investigation, language is very broadly defined. It really comprises any symbol system, including natural languages, formal languages, programming languages, phonetics, and notational systems of music, possibly even the language of the central nervous system. Why define it so broadly to begin with? Well, there are several reasons for thinking in this very general way. One is the notion that a given notational system or language, and sometimes I think it is useful to think of a language as a notational system and vice versa, can serve as a heuristic for problems in another discipline. Another kind of interest of ours is a basic motivation for this undertaking. It is an interest in how one notational system or language maps onto another. What do you lose when you move from one language to another and what do you gain? What are the disparities? Very little work really has been done on this topic. So far as natural languages are concerned, I will talk more about natural languages than others because that is my area, linguists have tended to concentrate on syntactic categories and comparative linguistics trying to see if syntactic categories which are appropriate for one natural language might be appropriate for another. Very

little work has been done on semantics. No coherent effort has been made to see how natural languages map onto each other so far as semantics is concerned. So that gives you some sense of some of the motivation and general background for our interest in this kind of an effort.

Now perhaps it would be useful to mention some of the kinds of users, and I am going to define them as to goals, of such a center or network. We have talked to a lot of people, we have talked I suppose to now 150-160 people about uses they would like to make of this kind of facility. I am just going to give some examples of areas and people who might have an interest in this kind of effort. A number of people are now concerned with cognition in some form and the aspect of their work that I find interesting is their effort to deal with semantics structures. People who work in this area now include: Don Norman, University of California, San Diego, who is a psychologist, Aaron Ciccorrel, also there who is a sociologist, Ray Dongradi who is also at San Diego who is an anthropoligist. And they are interested in modelling cognition for various reasons, but in general I suppose the sociologist and the anthropologist want to try to get at the way people structure reality, structure their worlds. You can argue that the language which people use will tell you something about the way they perceive reality. And it is in this kind of context that comparative semantics has extremely important implications, I think, for society today. Ableson at Yale is another leading psychologist who is interested in this kind of effort, some people at Stanford, John McCarthy now I think is pursuing this in connection with machine translation which he seems to be reviving. I did not mean to mention machine translation, but I will in that context. And I suppose you could say that Winograd at MIT and some of the other people associated with Minsky are to some degree modelling cognition. Another kind of project is concerned with computer support of linguistic field research. Linguist Joseph Grimes of Cornell wants to set up, and is already beginning, a comparative word list data base which would store words which linguists pick up in the field when describing languages. This would mean then that a linguist working in the field would be able to very quickly see whether a particular vocabulary or subset of a vocabulary used by one culture or subculture is entirely disparate or somewhat disparate from the vocabulary used by other cultures which have been investigated by linguists. He would, by the way, like to have a terminal in New Guinea or wherever he is so that he can access his files which in this case would be at the University of Oklahoma. Obviously

a number of software problems are implied by such a method. Another kind of project which is related to linguistic field research might be to continue the effort begun at Berkeley a few years ago on Chinese dialects. Professor Wong had a rather large research group at Berkeley at one time working on Chinese dialects, and many of those researchers are now scattered and they are not able to access the data files or the programs they developed, for studying Chinese dialects. So there is an example of a research project which in a sense has disappeared because the people who were involved with it took their degrees and went off to places like the University of Virginia, and they are no longer able to work on the project except in the summer, if they can find the funding to get to Illinois in this case, or Berkeley. Another large area representing prospective users is speech. We mention the project underway at the Haskin's laboratory. They are interested in reading for the blind which means moving from the printed page to spoken output. They also are concerned with another research problem which has to do with the encoding that controls the musculature which controls articulation. Speech it seems to me is an integral part of any effort concerned with language research. Another kind of research with which I have been associated is what I call stylistic analysis. As Walter and I define it a stylistic analyzer is really a comprehensive pattern recognizer for natural language. I define style so that it includes content, I do not distinguish between words and their meanings and whatever form they may take on the page. So, a comprehensive pattern recognizer for natural language would include content analysis, that in turn includes semantics, that in turn for us has entailed research on topics presently such as the theory of prefixing.

We have chatted with Cheatham, Irons and Bob Rotens as far as research on computer software is concerned. They have interest in comparative syntactics and comparative semantics, and also have indulged in speculation about the kind of hardware which is desirable for this kind of research. We have talked with people in art, Kenneth Nolton for one, who are concerned with synthesis, the major graphics capability. That perhaps gives you some sort of picture of the range of people with whom we talked.

Questions more directly related to software validation and distribution would flow from whether the effort was a center or some sort of network. If it is a network there are obvious problems - some of which ARPA has dealt with and some not. ARPA seems to have dealt with the interfacing communications software implied by the network. It has not dealt with the problem of the load on any

given computer at any given time. So far as ARPA network is concerned the user has got to know what he wants, he must know where it is and he must also know the system he intends to use.

In response to a question Gordan Sherman raised after the linguistics conference in Washington, let me make an observation or two as to the big range of user goals which define in part the demands upon documentation and validation and the interest in sharing programs. As was obvious from that conference some people write parsers because they want to explore in depth a particular linguistic theory or model. Stanley Petrick, interested on transfunctional grammar, might have no strong interest in using the Kuno Oettinger context-free phrase structure grammar except, perhaps, for purposes of comparison. On the other hand, the PEG people, who simply wanted sentences parsed in order to be able to say something about structural correctness and complexity, would be glad to use any parser which would give them that information. For the PEG people, validation of whatever form and state of parser they use is vital. For Petrick, validation is still important, but he would probably be willing to accept a larger number of bugs than the PEG people.

The point is, that there are systems within systems - a system which may be the focus at research for one project may be a production component in someone else's research project. I would suppose that systems or programs being used as production components should not only be more extensively validated than systems which are purely experimental but they should also be insofar as it is possible and known, bugfree.

Practically speaking, this means that applications systems, which are often extremely experimental, are handled analogously to operating systems; that is, at a particular stage of development they are validated, documented and made available; then later versions are documented and released as development of the system proceeds. The problem has been that most applications programs haven't been documented nearly as well as at least vendor-supplied programming language systems (which in my view sets a pretty low level). Ideally, no program should be accepted by the project director, or whoever is responsible, until it is well documented - a center or network ought to have some agency which assures both good documentation and validation.

Questions.

Do you have people who are primarily interested in developing systems for research? How do you motivate them to document?
Those of us who do research and have graduate assistants try not to accept what they turn in as an operating program unless it is documented.
In my case it is an incremental system - it has grown over the years. So one does a piece and documents that and does another piece. If you have some sort of center or network then you would have to have some sort of agreement so that if a given researcher was going to make his system available, then availability would either mean a key or someone acquainted with the system to provide the documentation.

Would you concede one of the purposes of the center would be maintenance?
It could be. And that has frequently been suggested as a very useful task.

Would there be great emphasis on mini-computers and switching of loads between computers or would you have many people but only one large computer?

In the case that we are concerned with which is language processing, possibly very broadly defined, it is not entirely clear whether the hardware that would be most desirable for this sort of research yet exists.

Remote Access Lexicographic System - Dr. Richard Venezky, University of Wisconsin, Madison, Wisconsin

I was not present here this morning so I am not sure what the appropriate or effective invocation is to the local dieties, I'm sure I will not be held liable for what I say. The examples that I will draw upon are taken mostly from the design of lexicographic systems, that is systems used to develop natural language dictionaries. However, my concern is with processing in the entire field of the humanities. What I will try to do is characterize the kinds of software that is used most typically in the humanities, especially in contrast to the social sciences and in some of the more number conscious applications areas, then attempt to characterize the users of the applications area and then make some very important comments on how software should be validated and distributed for the humanities given this characterization of the software on the one hand and the characterization of the user on the other hand.

When I talk about systems for making dictionaries, I mean systems that operate roughly as follows, I will give you a specific example. Two of the more interesting computational systems are concerned with the Italian dictionary and the Historical Hebrew dictionary. Now both of these projects are processing files of hundreds of millions of words. They are all trying to do automatic processing of forms in terms of discovery. Essentially they all start the same way. A text is encoded by hand then to document the manuscript they make certain types of henscratches to mark it for later analysis. These texts then go into machine readable form. Then the material is put into an archive and this has all kinds of implications for data bases that are going to be manipulated and variations of these data bases across different genre like poetry, prose, western novels, etc. There is a certain amount of error correcting. Then the test from the archives is processed in a variety of ways. One of the most common is to generate a concordance. In making the concordance one can go to a great degree of sophistication, in terms of what is identified as a word, how much stripping of prefixes and suffixes, how much automatic normalization, etc. goes on. The concordances then are used for certain kinds of analysis and at some point out of them will come a dictionary file. This dictionary file will then be used to match against further input, to determine cutoff limits for words for different kinds of semantic analysis. Eventually there will be some kind of output. The Hebrew and Italian projects may be typed out on a selectric typewriter at the rate of 12 characters per second of camera ready copy.

What characterizes the software here and distinguishes it from software in other applications is first the files are very large - 150,000,000 words, and in English a word is 6.3176 characters on the average, so we're talking about fairly large files that are processed over and over again. Social Sciences applications also process large files, but more typically one is talking about a single pass through a file. I am not sure that is totally true, but most of the applications that we see at our social science research center are large files that usually involve single passes. On the other hand, with the dictionary systems there are multiple passes through the files. There are a large number of different data bases that have to be designed according to the types of data that go in and according to an operations form. A lot of work done by the CODASYL group on data base management systems are directly applicable, at least to conversing about file structures. But the majority of that work does not quite touch on the problems that are found in a large amount of dictionary maintainence. There is a large amount of string processing, there is a large amount of pattern matching, that in general, is no different than what you do with SNOBOL. It simply operates on a much more restricted, linear string of characters all chosen from a rather restricted output deck so there is a very large amount of sorting. There is a limited amount of numerical work that takes place. Finally, to give one further characterization, there is a tremendous amount of looking up of words in dictionaries, and some of the more interesting work on software in the humanities has been concerned with look-up schemes where there may be up to 100,000 entries in a list. Some of the examples of the software that you can actually see in existence are first some of the text archiving systems, for example the group in England that worked on the Scottish national dictionary have constructed a large, but rather crude and uninteresting text archiving system. Each text goes in in exactly the format that the user who prepared it decided to put it into.

So much for software characteristics, other than summarizing by saying that primarily we are concerned with operations that are from a computational standpoint very simple, but perform on very large data files involving sorting, string manipulation, and dictionary lookup for the most part. One could probably put together most things done in the humanities out of these kinds of operations as long as they are willing to say that the marginal 5 or 10% can be anything.

The users in the humanities are very much like the users in the social sciences. They vary from extremely sophisticated systems programmer types down to the larger number of persons who could probably be characterized by an inate disdain for subscripts. What they have gained about computing they have gotten from

New York Review of Books, publications of the Modern Language Association and other technical journals. In simple words, the average user has not read Knuth, Volume 1. He would not know how to use the index anyway. Nevertheless he has learned about computing somehow, he has gotten on the machine, he is an extremely capable user in some higher level language like FORTRAN, and he has amazing will-power and endurance. He has limited funding and tends to learn better inductively than deductively, meaning of course that non-procedural languages are much more amenable to his life style than procedural languages. Now by this I don't mean to say that these users are totally inexperienced in computing. They are simply not overly interested in the niceties of computing or computing machinery. They are not the kind of people who can read a normal form definition of a language and begin to use it right away. They certainly could learn how to use it but that is not their interest. Their interest is research. Their interest is getting a particular job done. Now given this kind of user characterization and some of the notions about the software that these users use, let me make some very important statements about validation and distribution.

The distribution of software in the humanities and there is a very desparate need for this especially for what sounds like very trivial systems - concordance generators, file maintainence, text archives, and so on. A system that is distributed should be composed of the following four items: a source program; a standard text deck with a listing of the output, to be used, of course, for checking out installations; and then internal and external reference specifications; user documentation and maintainence documentation. Now the source deck - the source program raises all sorts of problems about portability because we are talking about systems that have to be used across a large number of machines that have different character sizes, different internal word sizes, and therefore create all sorts of problems if you are insisting upon writing in Assembly Language or if you insist on writing in any language that takes advantage of the actual word size of a machine. The programs that have been portable generally have a very small number of primitives written in something, and then a large system that is machine independent built on these primitives. With this machine it is possible to write fairly general and portable software. There are however a number of subtle problems, for instance hash coding is a very effective table lookup scheme especially for our dictionaries, however hash codes are extremely machine dependent. You cannot just simply design a good hash code scheme to run on one machine and expect it to run as efficiently on another machine. You are almost guaranteed to get a totally different collision rate. There are two systems that are being

distributed fairly widely, one of which does use a type of hash coding although there is no information on cross machine comparison. This is the spiral text management system of the Sandia Corporation. There is another system called SAMULOS which is a bibliographic historical retrieval scheme from the National Forestry Service. It is purposely designed for CDC, IBM, and UNIVAC equipment and runs pretty much with the same efficiency on these machines. It is totally written in FORTRAN.

User documentation is a very big problem for users in the humanities. One simply cannot write a reference like an IBM FORTRAN manual and assume that for every page you are reading, you have read every page before and after. You truly need a training reference guide. In fact the term reference guide is inappropriate for the kind of documentation needed in humanities. Secondly if you want to distribute your concordance routine and dictionary lookup routine or any other routine that manipulates large files you must provide timing figures, you must give some notion of where the various steps come in the sorting operation. Otherwise at a typical installation as we see happen over and over again, someone brings his 28 boxes of card in, he has punched all of the Illiad backwards and he is going to prove something about suffixes. He reads the manual, he submits the program to run, and gets through all of whatever, and halfway through the collating sequence as he is printing out the results his time runs out, after running five straight hours on a $1000 machine. He has used up his grant, half of the computing center's budget for that month and he has been kicked off. There is no restart dump anywhere, of course. However, if he were provided with some timing charts that said for this amount of input, estimating words in the following way, you can expect the following time for this part of the project, etc, this catastrophe might have been avoided. These are very typical experiences that we have thrown back at us. The user needs information when he should partition a file, when he should not try to handle 200 billion running words in a single file. These are things people in humanities are simply not aware of, it is a simple thing to tell them in general but very difficult to find people with sophistication in management of large files. Thirdly, it is important to give output formats in the software you distribute, because you are guaranteed no matter what system you write or how fancy it gets that half the users will want to do something else you have not thought of, they want to take the intermidiate tapes with the 200,000 words of Illiad coded backwards and do something that you have not built in. It would be so much simpler to tell them, and of course in the maintainence documentation we assume you are going to do that anyway, and finally, we assume you are

telling people what configuration they need to run on, especially in terms of amount of mass storage.

As far as maintainence documentation is concerned, what is needed beyond what most industries typically require is very clear notes on how to modify the main variables in the program. For example, if you have a system that has a format for input of 400 characters and someone wants 401 characters, he should be able to read in the maintainence documentation where and what he has to change to get 401 characters. We would expect some kind of control flow diagrams, core allocation, for each subroutine techniques and purpose, and of course input/output characteristics. Functional descriptions giving algorithms for each main process that the software undertakes, a very nice cross index of the main variables used. We would hope that if you are writing in something like FORTRAN that you would have a list of every variable and every common block and every routine it uses in every common block so that when you start changing them you do not spend six nights generating your own cross reference table. And with that kind of documentation validation and distribution of software in the humanities would be very simple.

Collaborative Research for the Development of a Certified Substantive Library - Dr. Wayne Cowell, Argonne National Laboratory, Argonne, Illinois; Dr. Cleve Moler, University of Michigan, Ann Arbor, Michigan; Dr. Yasuhiko Ikebe, University of Texas, Austin, Texas

The NATS project is a collaborative effort and we will make a team presentation this afternoon. The project is directed to the question of producing highly reliable software. This morning Gordan Sherman mentioned quality tested software which is accurate, robust, portable, easily maintained, easily available, and I am sure we can think of other adjectives. The desirability of such software is undisputed. The question of how much does it cost and how do we organize ourselves to do it comes up. The NATS Project is a frontal attack on those questions, in which we select software, verify it, distribute it, support it, not only for the sake of producing some good software, but as a learning experience so that we can have a look at what was done, and how to do it in the future. I have written down several topics to cover briefly here.

What kind of software did we select? The two packages that we have directed our attention to are first of all a package of eigensystem routines called EISPACK. FORTRAN versions of some 34 which originally appeared in Algol in Wilkinson's and Reinsch's book [1]. The second class are special functions which were developed for the most part at Argonne. Now after we have been working on this for a year guided by the interest of the people who were involved and in a position to sit back and reflect a little bit, we ask, why did we select these particular packages and what characteristics are desirable for the software. Based on hindsight we wrote down these principles:

Principle A - The algorithms underlying the routines have a sound numerical basis.

Principle B - The routines are written in a widely used source language. They have undergone testing for efficiency and absence of gross errors. Basic documentation exists.

Principle C - The routines have been organized into a coherent collection which solves a class of problems.

Principle D - Computable measures of quality exist.

Principle E - The routines satisfy a basic computational need and are required for use in a number of institutions. They have potential for becoming an authoritative standard within the computing community.

The first three are basically criteria for algorithms which would be selected as candidates for certification. The fourth principle says that we are able to tell when we have acceptable subroutines. And the last one says that it is worth doing in the sense that there is a real need for the software in question.

NATS is collaborative in the sense that there are three principle institutions involved - Argonne National Laboratory, Stanford University, and the University of Texas. There are a number of cooperating test sites. We have been able to involve some first class people who are willing to spend some part of their time on the testing effort.

What sort of procedures are involved? First of all, let me mention what we mean by certification. We talk about it rather easily, but we found that as we got into the work and started thinking very carefully about it, that is a term that has to be very carefully defined. The sample documentation that I am going to distribute has this statement in it [see Appendix]. It says that this subroutine has been tested on and herewith certified for the following machines and operating systems (then follows the list). The NATS project fully supports this in the sense that detailed information on the testing procedures is available and reports of poor or incorrect performance on at least the machines and operating systems listed will gain immediate attention from the developers. There is a man's name where questions and comments should be directed. Certification says not that we are willing to pay you for the time that you waste if the routine does not work, but it says that we have very carefully tested it, that we believe in it, and we are willing to stand behind it.

The overall view of how we brought this about is in Figure 1. Input into the system are routines that satisfy the first three principles that I had up here - they are good candidates for certification. The next step is to assemble the routines into packages which are in a standard format, with documentation. There has been local testing and analysis by the developers and in this case by groups at Argonne, Texas and Stanford. Then check out a test site. There are two aspects to field testing. One is computer center checkout - the presenting of selective test cases, using drivers furnished by the developer, a fairly formal structured testing process. Then field testing is done. There is feedback to the developers, refinement of the routines. Once the software has reached the stage where it has been certified, in the sense I described, the feedback to further refinement, further assembly into packages will take place only in very unusual circumstances. That is not to say we will not improve the routines

over a period of time, but they have had to be gone into so we expect them to be quite stable. The developers - the people responsible for the program - produce driver programs, test systems and otherwise supply and help the field test site people who actually have the routines and are trying them out on their system. A list of test sites is shown in Figure 2. The primary machines are IBM 360 and CDC 6000/7000 series machines - there is one each of Honeywell, a UNIVAC 1108, an ICL 1906A, a British computer. There are six sites involved in testing of program machinery. At each of these test sites there is a representative with whom we correspond, sending the programs and information about the programs, and he in turn is expected to coordinate testing at his site and to feed information back to us.

Talking briefly about my experience, we sent edition one of the Eigensystem package to the test sites at the end of last summer. I should say that at the beginning of the summer they were used at the Engineering Conference at the University of Michigan. But they were at the rest of the test sites at the end of the summer. They were tested through the fall, and several bugs were located, only one really hard bug was located. There were a number of suggestions made for improvement. On December 14 last year there was a meeting at Argonne of the test site representatives, where the experience they had thus far was examined, compared, and they gave us their suggestions on ways to improve the package for edition two. I might note one way in which we listened to them. They were somewhat unhappy about the fact that there were machine dependent parameters in the calling sequence, and there seemed to be a consensus that we take those out. We did and the result is that we now have five versions of the Eigensystem package, whereas before we had two. We have a master version from which we generate the other four - actually we generate all five because the master version is on a different machine. We are able to do this and produce the tapes which can be read on the various test site computers using OS 360 on a 75. That took a fair amount of development and I think we solved some of the problems there which we will try to write up and share with other people trying to do the same sort of thing. I want to pass out some examples of documentation (see appendix). These are drafts but they are almost complete. One of these is the Eigensystem package and one is the exponential integral. It is a very simple thing. The second edition of the Eigensystem package went out to test sites in February. The information is still coming back to us, we expect to be able to certify after they run the second edition, we know of no errors that we need to correct. We expect to get general distribution of these from the Argonne code center in three to four weeks for the IBM version [availability has now been

announced - LDF]. We hope that at each of the field test sites the coordinator has made consulting available. The field test sites are not really paid for this effort. Many are putting their own resources into it. Let me say we underestimated the difficulties and our deadlines continually slip. I think that assembly into packages and making tapes and that sort of thing are difficult problems. I think we have solved both of those problems now, but they were hard.

A word about costs. If we take the EISPACK code as a set of five package: a source code statement in one is different from a source code statement in another - that is our assumption. We have to run on six machines because we have five certified packages. I believe if we include the AEC development and the NSF support for this certification activity that it probably costs somewhere around $15 per source code statement. I think that can be improved because a lot of that is initial development cost. I am not including the test site support they furnish themselves, I am not including the initial research.

Yasuhiko Ikebe

At Texas we are also supported by the National Science Foundation. We have a 6600 and a 6400. Our project began in January of last year. By the end of the last year we had certified and checked all group two EISPACKS. I am planning to use the package in teaching.

Cleve Moler

One thing that has concerned us all along is the question of user confidence, we want to avoid the SHARE image. So we have been very careful and have taken a long time before we released these programs. We have started with very good material. We found it helpful to make use initially of the WATFIV compiler, at Stanford to do a syntactic check. We had been running the programs in FORTRAN at Argonne, and before we sent them out to users we ran them on WATFIV at Stanford. That produced a lot of complaints from WATFIV and corresponding changes have been made in the FORTRAN codes. That particular test helped out the test sites. In matrix testing Wayne mentioned the measure of quality. We have a very simple measure of quality. All we have to do is try to solve $Ax=\lambda x$ and then compute the residuals. If they are small that means the routine was was correctly implemented. We have the Wilkinson inverse error analysis to back us up. We can prove theorems about the round-off error and what it must be, assuming the routine is correctly implemented. We are really not testing the

algorithm, we do not have to run large matrices and compare the computer results with the true results. We know what is going to happen there because of inverse error analysis. The only thing we have to test is whether the code is correct. I have been concerned in several different ways. I have used the codes in class. We used them at the Engineering Summer Conference at Michigan. They will be used again this year, Wilkinson will be here as one of the lecturers. Then we have been concerned with testing of real users. It is difficult to find real users, but we have some. I would estimate between Stanford and Michigan we have somewhere between 50 and 100 real users. We do not get much feedback for the users. They come from Engineering. One of our best success stories is a graduate student in Engineering at Michigan who had an eigenvalue problem with a complex matrix of about 10 x 10. He was solving it by forming a 20 x 20 real matrix, calling the SSP routine. We have in this package complex routines. Using those reduced his run time from 2.3 seconds to .13 seconds, cut his storage requirements in half and increased his accuracy. Deep down in the bowels of SPSS, in the factor analysis program, used to be the same SSP eigensystem routine. At Stanford last year we took that out. It reduced job time from 11 minutes to 2 minutes. Through this conference we have made contact with statistical people and took a look at some of the same things there.

Let me summarize. First of all we have a set of validated routines, ready to go. We developed a testing methodology. We developed a relation with users. We can improve the efficiency and accuracy of people's computations and we have extended their capabilities. We have got our foot in the door to work cooperatively and educate each other as to our needs.

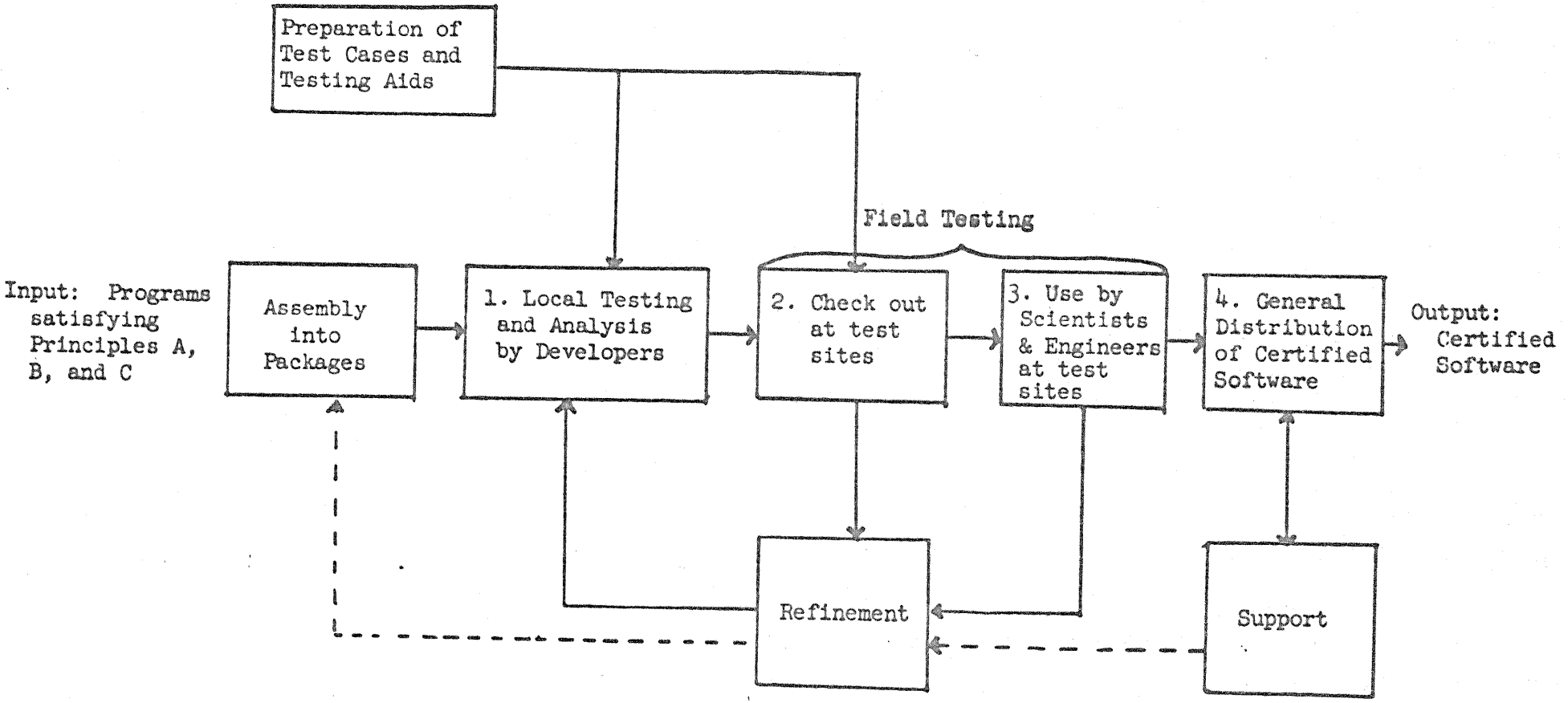
W.J. Cody, Applied Mathematics Division, Argonne National Laboratory, Argonne, Illinois

The major emphasis of this project has been on the matrix codes and I think rightfully so. We have frequently stopped all work on special functions to work on matrix material. They were originally included in the project for a very special reason. Matrix codes are in a sense transportable. That is, we have essentially 34 master programs which, with minor changes in the program, will run on a variety of systems. In the case of the special function codes the portability is the last thing we want. These codes are highly tuned to individual computers, and operating systems. To give you an idea, I was just talking to the gentleman from NCAR during the coffee break and I was under the impression that the 6000 series special function routines would perform on the 7600 as well. It turns out that the 7600 at NCAR does not have the option of rounding. This code

has been optimized for rounding on the 6000 series. This means we either accept inferior performance from using these codes on the 7600 or we support yet another version. At the moment, there are 14 special functions involved, there are three computers for which we are programming: the IBM 360, the CDC 6000 series and the UNIVAC 1108. For each separate subroutine we must go through a process of evaluation on the machine, comparisons against precise machine arguments and corresponding precise function values. At the moment these function value pairs are supplied by magnetic tape. The magnetic tape is generated on our CDC 3600 at Argonne, it formats the data for each of the three machines. We have concentrated lately on the generation of such tapes. The status: each version of these routines requires not only the source deck but a magnetic tape and at least two test drivers. One is the test driver that produced the magnetic tape which allows for detailed testing. The second is a less ambitious but equally demanding demonstration program which will supply the routine with a limited set of arguments and function values. We send you the source deck, the demonstration program, and the differences we have obtained. You can run them and compare the differences you get on your machine to what we have obtained in our testing. If there is much difference we want to know about it. The CDC 6000 series routines have been fine tuned at the University of Texas, the UNIVAC routines are being fine tuned at the University of Wisconsin. We have had requests to include more machines, but we are not quite sure what the scope of this might be.

References:

1. Wilkinson, J.H. and Reinsch, C.H., Linear Algebra, Handbook for Automatic Computation, Vol. 2. Springer-Verlag (1971).



Over-all View of Collaborative Software Testing

Fig 1.

appendix

DRAFT

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

-E2F220- ELTRAN

A FORTRAN IV SUBROUTINE TO ACCUMULATE THE TRANSFORMATIONS IN THE REDUCTION OF A REAL GENERAL MATRIX BY ELMHES.

APRIL, 1972

1. PURPOSE.

THE FORTRAN IV SUBROUTINE ELTRAN ACCUMULATES THE STABILIZED ELEMENTARY SIMILARITY TRANSFORMATIONS USED IN THE REDUCTION OF A REAL GENERAL MATRIX TO UPPER HESSENBERG FORM BY ELMHES (E2F273).

2. USAGE.

A. CALLING SEQUENCE.

THE SUBROUTINE STATEMENT IS

SUBROUTINE ELTRAN(NM, N, LOW, IGH, A, INT, Z)

THE PARAMETERS ARE DISCUSSED BELOW AND WORKING PRECISIONS FOR THE VARIOUS MACHINES ARE GIVEN IN SECTION 7.

NM — IS AN INTEGER INPUT VARIABLE SET EQUAL TO THE ROW DIMENSION OF THE TWO-DIMENSIONAL ARRAYS A AND Z AS SPECIFIED IN THE DIMENSION STATEMENTS FOR A AND Z IN THE CALLING PROGRAM.

N — IS AN INTEGER INPUT VARIABLE SET EQUAL TO THE ORDER OF THE MATRIX A.
N MUST BE NOT GREATER THAN NM.

LOW, IGH — ARE INTEGER INPUT VARIABLES INDICATING THE BOUNDARY INDICES FOR THE BALANCED MATRIX. SEE SECTION 3 OF E2F269 FOR THE DETAILS. IF THE MATRIX IS NOT BALANCED, SET LOW TO 1 AND IGH TO N.

A — IS A WORKING PRECISION REAL INPUT TWO-DIMENSIONAL VARIABLE WITH ROW DIMENSION NM AND COLUMN DIMENSION AT LEAST IGH. ITS LOWER TRIANGLE BELOW THE SUBDIAGONAL CONTAINS THE MULTIPLIERS WHICH WERE USED IN THE REDUCTION TO THE HESSENBERG FORM. THE REMAINING UPPER PART OF A IS ARBITRARY. SEE SECTION 3 OF E2F273 FOR THE DETAILS.

INT IS AN INTEGER INPUT ØNE-DIMENSIONAL VARIABLE OF DIMENSION AT LEAST IGH. INT IDENTIFIES THE ROWS AND COLUMNS INTERCHANGED DURING THE REDUCTION BY ELMHES. SEE SECTION 3 OF E2F273 FOR THE DETAILS.

Z IS A WORKING PRECISION REAL OUTPUT TWO-DIMENSIONAL VARIABLE WITH ROW DIMENSION NM AND COLUMN DIMENSION AT LEAST N. IT CONTAINS THE TRANSFORMATION MATRIX PRODUCED IN THE REDUCTION BY ELMHES TO THE UPPER HESSENBERG FORM.

B. ERROR CONDITIONS AND RETURNS.

NONE.

C. APPLICABILITY AND RESTRICTIONS.

IF ALL THE EIGENVALUES AND EIGENVECTORS OF THE ORIGINAL MATRIX ARE DESIRED, THIS SUBROUTINE FOLLOWS ELMHES (E2F273) AND SHOULD BE FOLLOWED BY HQR2 (E2F287). OTHERWISE, THIS SUBROUTINE WILL NOT ORDINARILY BE USED.

3. DISCUSSION OF METHOD AND ALGORITHM.

SUPPOSE THAT THE MATRIX C (SAY) HAS BEEN REDUCED TO THE UPPER HESSENBERG FORM F STORED IN A BY THE SIMILARITY TRANSFORMATION $F = G^{-1} C G$ WHERE G IS A PRODUCT OF THE PERMUTATION AND ELEMENTARY MATRICES ENCODED IN INT AND IN A LOWER TRIANGLE OF A UNDER F RESPECTIVELY. THEN, ELTRAN ACCUMULATES G INTO THE ARRAY Z.

THIS SUBROUTINE IS A TRANSLATION OF THE ALGOL PROCEDURE ELMTRANS WRITTEN AND DISCUSSED IN DETAIL BY PETERS AND WILKINSON (1).

4. REFERENCES.

1) PETERS, G. AND WILKINSON, J.H., EIGENVECTORS OF REAL AND COMPLEX MATRICES BY LR AND QR TRIANGULARIZATIONS, NUM. MATH. 16, 181-204 (1970). (REPRINTED IN HANDBOOK FOR AUTOMATIC COMPUTATION, VOLUME II, LINEAR ALGEBRA, J. H. WILKINSON - C. REINSCH, CONTRIBUTION II/15, 372-395, SPRINGER-VERLAG, 1971.)

2) BOYLE, J.M., GARBOW, B.S., KLEMA, V.C., AND SMITH, B.T., A USER'S GUIDE FOR THE EIGENSYSTEM PACKAGE EISPACK, ARGONNE NATIONAL LABORATORY REPORT (IN PREPARATION).

5. CHECKOUT.

A. TEST CASES.

ELTRAN HAS PERFORMED SUCCESSFULLY ON A LARGE COLLECTION OF TEST MATRICES AND HAS BEEN TESTED UNDER THE AUSPICES OF THE NATS PROJECT ON THE MACHINES LISTED IN SECTION 7.

B. ACCURACY

ELTRAN INTRODUCES NO ROUNDING ERRORS SINCE IT ONLY TRANSFERS THE MULTIPLIERS, USED IN THE REDUCTION PROCESS, INTO VECTOR MATRIX Z.

C. DEMONSTRATION TEST.

THE DEMONSTRATION DECK, WHICH MAY BE ORDERED FROM THE ARGONNE CODE CENTER, TESTS ELTRAN AS PART OF THE PATHS FOR REAL GENERAL MATRICES USING SUBROUTINES FROM THE EISPACK PACKAGE (2). THE MEASURE OF THE CORRECT PERFORMANCE OF THIS PATH IS BASED ON THE NORM OF THE RESIDUAL MATRIX AZ-ZW, WHERE W IS THE DIAGONAL MATRIX OF EIGENVALUES. AN EVALUATION OF PERFORMANCE OF THE PATH, BASED ON THIS MEASURE, IS DISPLAYED IN THE OUTPUT FROM THE TEST RUNS.

6. ACKNOWLEDGMENTS.

EISPACK WAS THE RESULT OF THE COMBINED EFFORTS OF MANY PEOPLE. BUILDING ON WORK CITED IN THE REFERENCES, J. M. BOYLE, W. J. CODY, B. S. GARBOW, V. C. KLEMA, AND B. T. SMITH IMPLEMENTED THE ROUTINES AT ARGONNE NATIONAL LABORATORY (SUPPORTED BY THE USAEC). C. B. MOLER OF THE UNIVERSITY OF MICHIGAN ALSO CONTRIBUTED TO THE IMPLEMENTATION.

CERTIFICATION WAS ACCOMPLISHED UNDER THE AUSPICES OF THE NSF-SUPPORTED NATS PROJECT WITH PRINCIPAL INVESTIGATORS W. R. COWELL (ARGONNE), W. J. CODY (ARGONNE), C. B. MOLER (STANFORD UNIVERSITY), AND Y. IKEBE (THE UNIVERSITY OF TEXAS AT AUSTIN). THE ROUTINES WERE THOROUGHLY EXERCISED BY 16 TEST SITES USING VARIOUS MACHINE CONFIGURATIONS, REFINED AT ARGONNE, AND FINALLY RETESTED FOR CERTIFICATION.

7. STATEMENT OF CERTIFICATION.

THIS SUBROUTINE HAS BEEN TESTED ON AND IS HEREWITH CERTIFIED FOR THE FOLLOWING MACHINES, OPERATING SYSTEMS, AND WORKING PRECISIONS.

MACHINE	OPERATING SYSTEM	WORKING PRECISION
IBM 360, MODELS 65, 67, 75	OS RELEASES 19, 20	LONG
UNIVAC 1108	EXEC 8	SINGLE

TO BE FILLED IN LATER AS RESULTS FROM TEST SITES BECOME AVAILABLE.

THE NATS PROJECT FULLY SUPPORTS THIS CERTIFIED ROUTINE IN THE SENSE THAT DETAILED INFORMATION ON THE TESTING PROCEDURES IS AVAILABLE AND REPORTS OF POOR OR INCORRECT PERFORMANCE ON AT LEAST THE MACHINES AND OPERATING SYSTEMS LISTED ABOVE WILL GAIN IMMEDIATE ATTENTION FROM THE DEVELOPERS. QUESTIONS AND COMMENTS SHOULD BE DIRECTED TO

B. S. GARBOW
ARGONNE NATIONAL LABORATORY
APPLIED MATHEMATICS DIVISION
9700 S. CASS AVE.
ARGONNE, ILLINOIS 60439

WRITEUP	4	PAGES.
SOURCE DECK	80	CARDS.
DEMONSTRATION DECK	###	CARDS.

Principle A - The algorithms underlying the routines have a sound numerical basis.

Principle B - The routines are written in a widely used source language. They have undergone testing for efficiency and absence of gross errors. Basic documentation exists.

Principle C - The routines have been organized into a coherent collection which solves a class of problems.

Principle D - Computable measures of quality exist.

Principle E - The routines satisfy a basic computational need and are required for use in a number of institutions. They have potential for becoming an authoritative standard within the computing community.

Appendix

DRAFT

NATS PROJECT

SPECIAL FUNCTION SUBROUTINE PACKAGE

DEI -F1EI-

A FORTRAN FUNCTION SUBROUTINE TO CALCULATE EXPONENTIAL INTEGRALS E-SUB-I(X), EXP(-X)*E-SUB-I(X), OR E-SUB-1(X), FOR A GIVEN ARGUMENT X.

FEBRUARY, 1972

1. PURPOSE.

THIS SUBROUTINE COMPUTES APPROXIMATE VALUES OF THE EXPONENTIAL INTEGRAL

$$EI(X) = \text{INTEGRAL (FROM } -\text{INFINITY TO X) } (EXP(T)/T)\ DT, \quad X .GT. 0.$$

OR

$$EI(X) = -\text{INTEGRAL (FROM } -X \text{ TO INFINITY) } (EXP(-T)/T)\ DT, \quad X .LT. 0.$$

WHERE THE FIRST INTEGRAL IS A PRINCIPAL VALUE INTEGRAL. ALTERNATE ENTRY POINTS RETURN VALUES OF

$$EXP(-X) * EI(X),$$

OR

$$E1(X) = -EI(-X), \quad X .GT. 0.$$

2. USAGE.

A. CALLING SEQUENCE.

THE CALLING STATEMENTS ARE

```
Y = DEI(X) ,

Y = DEXPEI(X) ,
```

AND

```
Y = DPEONE(X)
```

WHERE ALL QUANTITIES ARE AT WORKING PRECISION, AND WHERE THE ENTRY POINTS CORRESPOND TO THE FUNCTIONS EI(X), EXP(-X)*EI(X), AND E1(X) RESPECTIVELY.

IN ADDITION, ERROR MONITORING AND ERROR STATUS CHECKS MAY BE MADE USING THE STATEMENT

```
XJ = DEIMON(XK)
```

WHERE ALL QUANTITIES ARE AT WORKING PRECISION AND XK HAS AN INTEGER VALUE (SEE SECTION 2B FOR DISCUSSION OF THESE PARAMETERS).

B. ERROR CONDITIONS AND RETURNS.

THE FOLLOWING TABLE INDICATES THE TYPES OF ERROR THAT MAY BE ENCOUNTERED IN THIS ROUTINE AND THE FUNCTION VALUE RETURNED IN EACH CASE. (XINF IS THE LARGEST MACHINE

NUMBER. XMIN IS THE SMALLEST LEGITIMATE ARGUMENT FOR DEI, AND XMAX IS THE SMALLEST ARGUMENT WHICH WILL CAUSE OVERFLOW IN DEI. SEE SECTION 2C FOR APPROXIMATE VALUES OF THESE PARAMETERS.)

ERROR NO.	ERROR	ARGUMENT RANGE	FUNCTION VALUES FOR DEI	DEXPEI	DPEONE
1	UNDERFLOW	X .LT. XMIN	Ø	-	Ø
		X .GT. -XMIN	-	-	Ø
2	OVERFLOW	X .GE. XMAX	XINF	-	-
3	ILLEGAL X	X = Ø	-XINF	-XINF	XINF
4	ILLEGAL X	X .LT. Ø	-	-	USES ABS(X)

THE DEFAULT ERROR PROCEDURE IS TO ONLY PRINT AN ERROR MESSAGE FOR THE FIRST ERROR ENCOUNTERED. THE FREQUENCY OF EACH TYPE OF ERROR IS ACCUMULATED, AND EXECUTION IS CONTINUED. THE FOLLOWING TABLE DETAILS WAYS IN WHICH THESE ERROR PROCEDURES MAY BE ALTERED OR THE ERROR STATUS INTERROGATED USING THE FORTRAN STATEMENT

```
XJ = DEIMON(XK)
```

WHERE XJ AND XK ARE WORKING PRECISION FLOATING POINT INTEGERS. EC-N DENOTES THE ERROR COUNT FOR ERROR NUMBER N IN THE ABOVE TABLE, AND -- DENOTES NO CHANGE FROM THE PREVIOUS SETTING.

XK	XJ	ERROR COUNTS	ACTION UPON FINDING ERROR PRINT MSSG.	STOP**
.LT. -1	XK	--	--	--
-1	XK	--	YES	YES
Ø	XK	RESET	1 ONLY	NO
1	XK	--	YES	NO
2	XK	--	NO	NO
3	*	--	--	--
4	EC-1	--	--	--
5	EC-2	--	--	--
6	EC-3	--	--	--
7	EC-4	--	--	--
.GT. 7	XK	--	--	--

* THE RETURN FOR XK=3 IS THE ERROR NUMBER N (FROM THE PREVIOUS TABLE) FOR THE LAST ENTRY TO THE ROUTINE. N=Ø INDICATES NO ERROR CONDITION WAS ENCOUNTERED.

** INCLUDES A TRACEBACK IN THE IBM AND UNIVAC VERSIONS.

C. APPLICABILITY AND RESTRICTIONS.

Ø.Ø IS AN ILLEGAL ARGUMENT FOR ANY FUNCTION ENTRY TO THIS ROUTINE. DEI WILL COMPUTE FUNCTION VALUES FOR ALL NON-ZERO ARGUMENTS BETWEEN XMIN AND XMAX. DPEONE WILL COMPUTE FUNCTION VALUES FOR ALL POSITIVE ARGUMENTS LESS THAN ABS(XMIN) AND WILL TREAT NEGATIVE ARGUMENTS AS DESCRIBED IN SECTION 2B. THERE ARE NO RESTRICTIONS ON THE ARGUMENTS FOR DEXPEI (EXCEPT FOR X=Ø.Ø).

THE FOLLOWING TABLE LISTS APPROXIMATE VALUES OF CERTAIN MACHINE-DEPENDENT PARAMETERS FOR THOSE COMPUTERS FOR WHICH VERSIONS OF DEI ARE AVAILABLE.

MACHINE	WORKING PRECISION	APPROXIMATE VALUE FOR XMIN	XMAX
IBM/360	LONG	-175.0	182.6
CDC 6000 SERIES	SINGLE	-669.3	748.2
UNIVAC 1108	DOUBLE	-703.2	715.6

3. DISCUSSION OF METHOD AND ALGORITHM.
THIS ROUTINE USES RATIONAL CHEBYSHEV APPROXIMATIONS GIVEN IN REFERENCES (1) AND (2) TO EVALUATE THE VARIOUS EXPONENTIAL INTEGRALS. SUGGESTIONS GIVEN IN THE REFERENCES FOR IMPROVING THE NUMERICAL CONDITIONING OF THE COMPUTATIONS HAVE BEEN FOLLOWED. IN ADDITION, OTHER ADJUSTMENTS HAVE BEEN MADE TO MINIMIZE THE EFFECTS OF COMPUTER ARCHITECTURE, E.G., ROUNDING.

4. REFERENCES.
1) CODY, W.J. AND THACHER, HENRY C. JR., RATIONAL CHEBYSHEV APPROXIMATIONS FOR THE EXPONENTIAL INTEGRAL E1(X), MATH. COMP. 22, 641-649 (1968).

2) ----- , RATIONAL CHEBYSHEV APPROXIMATIONS FOR THE EXPONENTIAL INTEGRAL EI(X), MATH. COMP. 23, 289-303 (1969).

3) CODY, W.J., PERFORMANCE TESTING OF FUNCTION SUBROUTINES, PROCEEDINGS 1969 SJCC, AFIPS PRESS, MONTVALE, N.J., 1969, 759-763.

5. PROGRAM MATERIALS AVAILABLE.

MACHINE	WRITEUP	SOURCE DECK	TEST DECK
IBM/360	- 5 PP.	416 CARDS	160 CARDS
CDC 6000 SERIES	- 5 PP.	425 CARDS	160 CARDS
UNIVAC 1108	- 5 PP.	XXX CARDS	YYY CARDS

6. CHECKOUT.
A. TEST CASES.
THE FOLLOWING TABLES SUMMARIZE THE RESULTS OF RANDOM ARGUMENT ACCURACY TESTS USING TECHNIQUES SIMILAR TO THOSE DESCRIBED IN REFERENCE (3). MASTER FUNCTION VALUES WERE GENERATED IN 25S ARITHMETIC ON A CDC 3600 COMPUTER AND TRANSMITTED, TOGETHER WITH THE CORRESPONDING ARGUMENTS, TO THE TARGET MACHINE IN BINARY OR HEXADECIMAL, AS APPROPRIATE, VIA MAGNETIC TAPE.

IBM/360 (56 BIT FRACTION)

ARGUMENT RANGE	FREQUENCY OF BIT ERRORS NO. OF BITS IN ERROR 0	1	2	3	OTHER	MAX. REL. ERROR	RMS REL. ERROR
(-50.0, -4.0)	254	890	397	350	109	0.468E-15	0.141E-15
(-4.0, -1.0)	53	414	896	241	396	0.631E-15	0.247E-15
(-1.0, 0.0)	275	754	758	50	163	0.444E-15	0.138E-15
(0.0, 6.0)	563	668	421	297	51	0.107E-14	0.112E-15
(6.0, 12.0)	439	687	587	256	31	0.397E-15	0.114E-15
(12.0, 24.0)	527	685	458	256	74	0.539E-15	0.109E-15
(24.0, 100.)	664	655	474	188	19	0.306E-15	0.881E-16

CDC 6000 SERIES (48 BIT FRACTION)

ARGUMENT RANGE	FREQUENCY OF BIT ERRORS NO. OF BITS IN ERROR					MAX. REL. ERROR	RMS REL. ERROR
	0	1	2	3	OTHER		
(-50.0, -4.0)	845	989	166	0	0	1.430E-14	4.612E-15
(-4.0, -1.0)	734	943	316	7	0	2.032E-14	5.612E-15
(-1.0, 0.0)	957	888	155	0	0	1.414E-14	4.349E-15
(0.0, 6.0)	804	840	299	55	2	5.432E-14	7.207E-15
(6.0, 12.0)	626	781	427	154	12	4.733E-14	9.359E-15
(12.0, 24.0)	662	1060	278	0	0	1.575E-14	5.269E-15
(24.0, 100.)	1143	818	39	0	0	1.373E-14	3.541E-15

B. DEMONSTRATION PROGRAM.
THE DEMONSTRATION PROGRAM FOR THIS SUBROUTINE COMPARES COMPUTED VALUES OF DEI(X), DPEONE(X) AND DEXPEI(X) AGAINST STANDARD VALUES FOR VARIOUS ARGUMENTS. DEI IS ALSO INVOKED FOR 8 ARGUMENTS IN THE VICINITY OF XMIN AND 8 MORE IN THE VICINITY OF XMAX, AND ALL OPTIONS FOR THE ENTRY DEIMON ARE USED.

7. ACKNOWLEDGEMENTS.
THE VARIOUS VERSIONS OF THIS SUBROUTINE RESULT FROM THE COMBINED EFFORTS OF MANY PEOPLE. K. PACIOREK AND W. J. CODY PREPARED THE ORIGINAL FORTRAN VERSIONS AT ARGONNE NATIONAL LABORATORY UNDER AEC SPONSORSHIP. THE PROGRAMS WERE MODIFIED WITH ADVICE AND COMPUTATIONAL ASSISTANCE FROM Y. IKEBE, U. OF TEXAS, IN THE CASE OF THE CDC 6000 SERIES VERSION, AND FROM W. WALLACE, U. OF WISCONSIN, IN THE CASE OF THE UNIVAC 1108 VERSION. THESE PROGRAMS WERE THEN CHECKED BY TEST SITES WITH VARIOUS MACHINE CONFIGURATIONS. ALL OF THIS LATTER WORK WAS UNDER NSF SPONSORSHIP.

8. STATEMENT OF CERTIFICATION.
THIS SUBROUTINE HAS BEEN TESTED ON AND IS HEREWITH CERTIFIED FOR THE FOLLOWING MACHINES AND OPERATING SYSTEMS-

IBM/360 MODELS 50, 75 OS RELEASES 19 AND 20
CDC 6000 SERIES SCOPE XXX ,
UNIVAC 1108 EXEC-8

THE NATS PROJECT FULLY SUPPORTS THIS CERTIFIED ROUTINE IN THE SENSE THAT DETAILED INFORMATION ON THE TESTING PROCEDURES IS AVAILABLE AND REPORTS OF POOR OR INCORRECT PERFORMANCE ON AT LEAST THE MACHINES AND OPERATING SYSTEMS LISTED ABOVE WILL GAIN IMMEDIATE ATTENTION FROM THE DEVELOPERS. QUESTIONS AND COMMENTS SHOULD BE DIRECTED TO

B. S. GARBOW
ARGONNE NATIONAL LABORATORY
APPLIED MATHEMATICS DIVISION
9700 S. CASS AVENUE
ARGONNE, ILLINOIS 60439

Computer Based Technology Transfer in Civil Engineering - Dr. Steven Fenves, Carnegie-Mellon University, Pittsburgh, Pennsylvania; Dr. Melvin L. Baron, Paul Weidlinger Consulting Engineer, New York, New York; Dr. Robert Schiffman, University of Colorado, Boulder, Colorado.

I can give you some background and sum up our ideas and plans. Our plans are changing by the minute. As you probably know Civil Engineering is a very diversified profession, in terms of scope of activity encompassed: transportation; housing; environmental problems; etc. It has a terrible distribution in size. The largest consulting engineering firm does 3% of the consulting work and the largest contractor does 1% of the construction. The result of this is that support that many of the other professions and industries have enjoyed has never existed in Civil Engineering. The diversity in affiliation between consultants, in-house organizations, etc. is complicated. The result is that even though Civil Engineering was one of the first large heavy user groups of application programs and computers, we have not moved very much lately. Software started out on very small machines, locally developed. In the 60's the profession went through the idea of large scale systems, probably not as sophisticated as some of those discussed here today, but many of you have heard of ICES. The idea was to incorporate all of the separate procedures and large data bases into some form of integrated system and wrap up that work at that stage. So far that has not worked out too well. The systems have become extremely large. Even though these systems were designed to be expandable, this just has not come about. No more than three organizations out of the original organization at MIT, have actively added subsystems. As a result, the profession is in sad shape. There is duplication of effort and information lag. It is fair to say that for anything the Civil Engineer may want to do on the computer there are probably ten different programs written under ten different systems in ten different languages, in ten sets of assumptions that exist somewhere. Furthermore, it is fair to say that somewhere there is a doctoral dissertation encompassing a computer program and algorithm for an improved way of doing that. Yet the guys who have to do the job have no access to any of the systems, and they certainly do not have access to the new work coming out of research. There is quite a bit of concern. In many other industries and professions the information gap is narrowing, in Civil Engineering, because of the existence of computers the information gap is widening. The new methodologies are embodied in computer programs and are so complex and difficult to transport from the original environment in which they

were developed into the working environment of the practicing engineer. The engineer, in fact, is slipping behind. So the profession has been going through a great amount of soul searching on how resources may be pooled, how information may be transported, how quality of applications software should be improved, made portable, etc. The latest on this collective decision making process was a conference held in Boulder on Engineering Software Coordination [1].

Our project was just approved on March 15. It is concerned with a technical feasibility study of what a national software center for civil engineering software might do. Another way of looking at it, it is an exercise to apply software engineering. That term is probably sufficiently descriptive. No one has yet defined software engineering and we are attempting to do applied software engineering. There is a parallel study to look at some of the professional and economic implications of a national software center also funded by NSF. In essence, we are working on an assumption which was suggested earlier today by Dr. Thompson. Namely, that a center of this type could be viable if it does an active job - it cannot be a passive repository of programs collected or submitted. It has to do an active processing on the source material that comes in. And in the process it must improve the products coming in to be available on line to its potential users. By that mechanism it lets the originator off the hook. Several people alluded to this problem. When we are talking about an industry distributed across the country with users using software in a variety of circumstances and a variety of machine configurations, it is not feasible to put down the phone number of the originator on the form. He is not the person who can be held responsible for the implemented operating version. He can be held responsible for the original idea or the first prototype algorithm, but his package - the distributed version - must be someone else's responsibility.

In essence we are going to look at two approaches, the preventive problem, that is, the development of guidelines, procedures of the design of software so as to be more easily distributed and more readily transportable. You cannot bank on the idea that everything will be redeveloped. We also have to do quite a bit of work on the post mortem approach. That is in some fashion collect the best ideas put forward by the profession, and then perform dissection, enhancement, etc. to obtain the product that the center may eventually distribute. This dissection is very important, because the kind of code that we encounter in the profession is a nightmarish mixture of engineering assumptions which are not unique, design assumptions, behavior assumptions of the particular design organization, and a strong component of numerical algorithms. This is the

software available to the engineer when he thinks about it. If all he has learned is reduction for solving simultaneous equations, that is what he codes in his program, and that portion cannot be easily separated. There is a tremendous component of environment dependent considerations, source programming language, operation system related considerations and then even worse than that, specific data structure implementations. They are all embodied in this one program. Our objective is to start with case studies, take representative programs, and again dissect and separate out these various components that I have mentioned. We intend to do this on a representative number of problem types and then through our procedures and software to generalize on this process. As you can see from your list there are three organizations involved reflecting the distribution of the entire profession. Dr. Schiffman represents the Computing Center here at the University of Colorado; he will look at the environment dependent aspects, the systems programming aspects, the portability and interface aspects. Dr. Baron represents a consulting engineer firm in New York City; their group is going to supply the engineering design oriented aspects, what sort of things can the practicing engineer at the end of the line tolerate. His firm is not a realistic test case. They are one of the most outstanding firms in the country, but I presume his people talk to more average civil engineers as well, and we can get feedback from other people. I represent the civil engineering department at Carnegie-Mellon University, and my primary goal in this project is going to be to help Joe Traub and his people start looking at the conceptual tools and techniques that can be applied. Here is an illustration. The civil engineering programs have become very complex and comprehensive. There are all kinds of options that the user can specify. What happens is that all of these options are exercised at execution time. The program reads a parameter and determines which subroutines it has to follow. This penalty for being able to do things you do not want to do on a particular run is very often an absolute penalty. The general purpose system will not fit on your machine whereas you know very well that the subsets you intend to exercise in your design should and will fit. So one of the techniques that excites us quite a bit is the concept of macro generation at time of execution whereby the parameters that are known to be fixed in advance can be specified in some form of a precompiler which will extract out of the master program those features that the user wishes to exercise and then extract an executable source program with just those features. We have done a small amount of experimenting with this approach where the logic of the program is described in the form of decision tables.

Dr. Melvin L. Baron:

I would like to say one thing about a problem that is particularly prevalent as far as the engineering profession is concerned. It is one that is changing. The normal engineering firm, at least for a while, somewhat resists computers. We found, in our own firm, some of the older engineers just do not know what a computer does and they are somewhat afraid of it. It is not until they find out what it does and that it could make them more productive that they are more enthusiastic. There is a problem with people using programs they know nothing about. We put together a program for the Navy; it was a set of design programs in engineering mechanics which remained in the time sharing mode and were used by a series of very unsophisticated users at about 20 different Navy labs. In the documentation we found it was very necessary to provide a whole spectrum of different levels of documentation, ranging from a paragraph or so description to maybe several pages of detailed description and several not insignificant test cases that the man could run and check and find out if he knew what he was doing. We found it was necessary to have continuous feedback, between the user and the systems developer, and this was accomplished by the means of comments whereby the user indicated to the developer what some of the pitfalls were. There was continuous revision of this thing until the amount of comments got smaller and smaller and smaller. In engineering there are very few experienced users, although I would imagine that in five years this situation will change.

Dr. Steven Fenves:

We have had considerable discussion on the question of what motivation the originator has in doing whatever extra work is necessary in order to get his algorithm or procedure incorporated in a system for distribution. There have been several ideas, one is to locate the software center somewhere in Hawaii or some place like this, and let the originator of the idea go out to the center for a month or so on vacation while he works with the center staff. Another more practical idea would be to guarantee to the submitter that after he has submitted his raw product, and after this has been dissected, enhanced, recombined, etc. the center will be in a position to return to him his program running 5-10 times faster than the program that he started out with. I do believe that this is possible.

We have not done the type of dissecting that I am talking about now with this philosophy in mind. All three of us individually have done a great deal of incidental dissecting necessary to move from one environment to another.

The most difficult problem for the center if such a center comes about, is going to be the initial selection. As I said, bits and pieces of just about anything that anybody wants to do exist, probably in lousy implementations and probably in highly localized environments. How does one apply professional judgment at that level to select either this idea or a collection of best ideas. I think the profession is ready to do this on some basis, to ask the consultants what are the best components of a program to be maintained in the system.

Reference

1. Schiffman, Robert L., Report on the Special Workship on Engineering Software Coordination. Report No. 72-2, Computing Center, University of Colorado (March 1972).

William Hetzel - Computing Center, University of North Carolina, Chapel Hill, North Carolina

I am the associate director of the UNC Computer Center. We have been interested in errors in software and methods for testing software for about five years. The first three years were simply a matter of keeping data on library programs. These have about 450 source statements changes a month on the total library. Changes per program are about 9 per month. I think my predecessor collected it without much thought. Before we applied to NSF we started to work with the data and see what the situation was. We discovered a program that had been executed on the UNC campus several thousand times with what we are calling a serious type error in it. The results that the user was getting were wrong by a small amount, small enough that it was not really noticable. But we know of two published results that were wrong based on that. This sent us off on testing programs, forgetting the methodology, but testing it as thoroughly as we could. In this general class that I'm talking about, large applications packages, we feel that at least half of the errors that are plaguing the users are not of the numerical analysis type but are a combination of improper logic and programmer mistakes, so we are emphasizing that aspect in looking at a way to come up with a better methodology. We chose an SPSS program and perhaps in a way that in retrospect is rather stupid, but we advertise incentive awards for anyone who can find anything wrong with it, and we ran a total of 1200 executions - 1200 different data sets were run. The SPSS people had stated that the program had probably been run about 10000 times before, and we found 7 errors, 6 of which are of the hard error type. My conclusion from this quite frankly is rather pessimistic.

We started out with the idea that someone knows how to do testing better so we sent out a survey, the survey was very simple, to programming development shops all over, and we simply said - do you know anything about how to test these things; if so, check yes and we will get in touch with you. We also said - if you are interested in test methods and want to be a part of this project, put your name and address here. We got a 61% return and only five people said they had anything to contribute, so I have a mailing list of 610 people. We subsequently visited four of those five, and did find some interesting techniques going on, but basically that was a waste of time. We have a

facility to do an automated search of data files, two of the files that looked like good ones to search were a NASA file and a CDC file. We searched both of those files and found only about three references to testing. The third thing we did was to make a series of trips, I visited a number of installations to see how they were doing it now and if they had something to offer. We went to SDC, RAND, Computer Sciences Corporation, NASA, CDC, Honeywell, Digital Equipment Corporation, Argonne, and a number of universities. A total of 25 trips to 43 places. I sat down and had talks with all the major manufacturers, many of these were well planned meetings where the manufacturer sent out a notice about the meeting. Several of the meetings required non-disclosure agreements. It turned out in general that these visits were very worthwhile. We put together a bibliography, an annotated one, with about 80 of what I consider relevant things about what has been published on testing methods, with a brief one sentence description of what is in the article. So these visits and literature searches give us a general feel for what was going on. All of this activity was part of the first activity of the NSF grant to familiarize ourselves with the state of the art. That concludes with a symposium to be held on June 21-23 in Chapel Hill, on computer program testing. We chose to have SIGPLAN sponsor it. Dr. Cowell is the program chairman. Many of the concerns some of you may have for high quality software may be addressed at that meeting. The general sessions will cover testing concepts and design of languages and programs to help testing large systems like operating systems and testing mathematical software. This is where we stand at the moment.

The next thing that I have started working on are the proceedings of the symposium which are going to be published as a Prentice-Hall book entitled Program Test Methods. This is essentially an edited proceedings. It should be available in September. We have started now in the second phase of what the grant was about. The first phase was to study the state of the art and make it known. I feel pleased because it was not in the original concept of the grant to have a large symposium or to produce a book. The second phase of the grant is to build a system, and the primary thrust of this is to be able to evaluate some test rules. Specifically we want to be able to quantitatively know whether a test is useful or not. We are planning to build a system which includes the test tools and ideas that have come up in this year that we have been collecting information. We would like to make a system that is easy to plug these tools into and easy to get data out on how easy they are. We hope to start the

implementation of the system in the fall, and will be experimenting with students to measure its effectiveness. The end result will hopefully be a methodology that we could recommend to developers of the packages that would reduce the probability of errors.

I want to make a couple of comments about the system. These are quite preliminary, there is quite a bit of modification going on as the design is slowly taking shape. I really do not want to sum it up until the conference is complete. I am hoping that I will learn more between now and then. First of all, we are not concerned with efficiency. That is evident by the way we tested SPSS regression. I am not talking about those kinds of techniques as such, but in general we are not going to be concerned with having a system that operates efficiently, we want one that will be able to measure the effectiveness of test rules. Similarly it is not a system that is expected to be transportable. We know it has to be flexible enough to define new test ideas and it has to be flexible enough for these ideas to be measured and evaluated. One of its goals is to be able to teach programming courses. I know of nowhere in the country where a programming course is taught with a real emphasis on testing. I think this is one of the fundamental problems in this area. For our work we need a static analyzer and a dynamic analyzer. The dynamic analyzer may be contributed to us by IBM. The static analyzer, I do not know where it is going to come from yet. One of the tools that we plan to first experiment with is a test data generator, by that I mean one that will generate test data that will travel down the various paths. This is the way to construct the thing - and plug it into the system to analyze it and have test data generated through the various paths. Unfortunately a lot of the test tools are not clear. The tools that are helpful still involve a lot of user interaction. Simple ones, such as printing out a graph don't require it, but some of the others that dynamically analyze a program do.

Planning an Approach to Testing and Dissemination of Computer Programs for Research and Development - Dr. Lloyd D. Fosdick, University of Colorado, Boulder, Colorado; Dr. Wayne Cowell, Argonne National Laboratory, Argonne, Illinois

Our project began early last fall. We are concerned with the problem of producing and distributing high quality mathematical software. Our thoughts have concentrated on a center which would serve as a focus of activities in this area. The need for such a center seems obvious to many of us, but providing a really convincing case which would lead to the actual funding of such a center is difficult. This is largely due to the fact that the cost of using bad software is hard to determine.

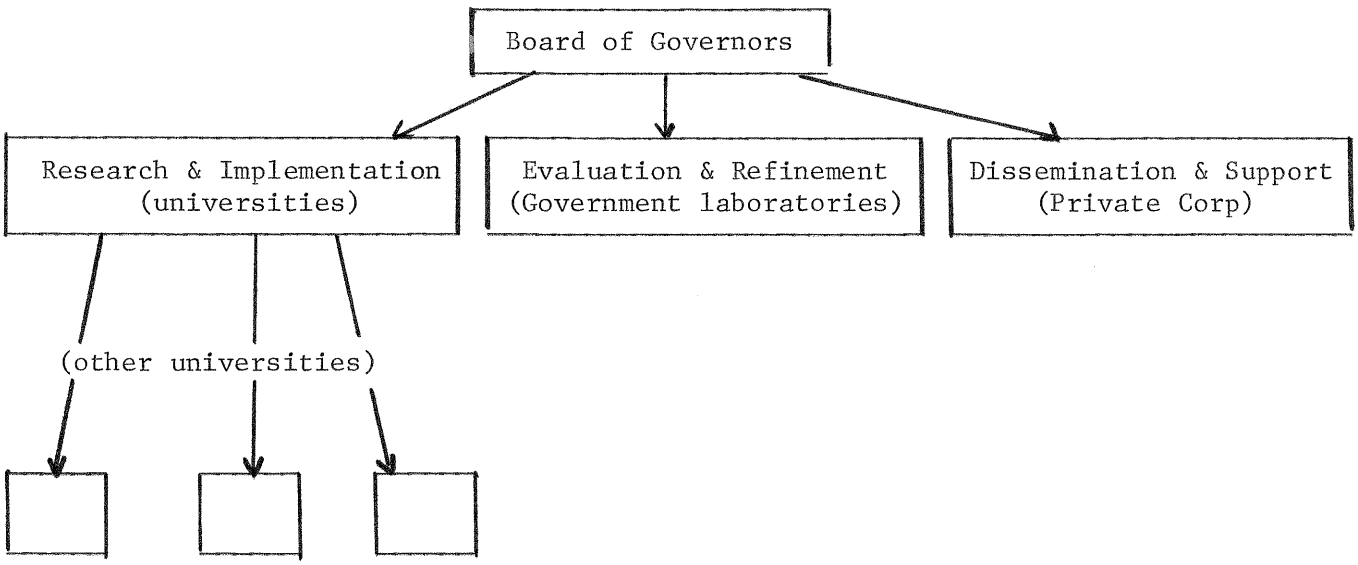
We are trying to obtain ideas and suggestions from outstanding members of the computing community who have interests in this area. Accordingly, we have visited with people at the National Bureau of Standards, the Jet Propulsion Laboratory, International Mathematics and Statistics Library, Carnegie-Mellon University, University of Michigan, University of Kentucky, and others. These discussions will culminate in a three day meeting involving about twenty-five people at the end of August. This meeting, to be held near Grand Lake Colorado, will attempt to tie together the various ideas and suggestions which have been made and draft a recommendation for some type of collaborative effort aimed at testing and disseminating high quality mathematical software for research and development.

To guide our discussions we have a list of five topics: definition of software quality; educational programs; research in testing and portability; sources of programs for testing and dissemination; coordination and funding. Each of these topics has been discussed during our visits, and the results of these discussions will be presented at the Grand Lake meeting.

Since the area of research in testing is a particular interest to me I would like to mention briefly some work we have been doing at Colorado; a number of ideas in this area are presented in a report being distributed to you (this report has since been published [CACM, Vol. 15, No. 7] - LDF). We are concerned with the problem of verifying that all statements in a FORTRAN program have been executed in the course of testing it. In order to verify this we have constructed a program, BRNANL, which accepts a FORTRAN program as data, identifies each basic block in the program, and inserts a special subroutine

call in each basic block. Thus output from BRNANL consists of a modified FORTRAN program wherein a subroutine call has been placed in each basic block. With this modified program it is then possible to ascertain which blocks have been executed during execution of the program. I should remark that a basic block is a sequence of consecutive statements which must be executed consecutively and which can be entered only at the first statement of the block; thus if a block is entered it is known that every statement in it must be executed. We are using this program experimentally to aid in testing ACM algorithms. Other work in this area includes identification of control paths in FORTRAN programs, and automatic generation of test data to force execution of selected control paths. We are interested in studying the control path structure of programs to see if any patterns emerge that might aid us in characterizing the programs for various purposes.

I would like to conclude with a brief comment on a proposed structure for a software center. A diagram is shown below:



We identify three main activities for the center: research and implementation; evaluation and refinement; dissemination and support. Each of these activities is identified with a different group, university, Government laboratory, private corporation; however, we anticipate considerable overlap. The attention of this center would be on mathematical software for the scientific community. I want to emphasize that this structure is just one of several under consideration. We hope to explore potential arrangements further in the coming months and to discuss them at the Grand Lake meeting in August.

Comments.

Dr. Wayne Cowell felt that the Board of Governors would be experts in the field, and this Board of Governors would change, therefore it would have to be innovative because of the change.

Another participant felt that everything depends on the fact that the center will save people money, and this software center will. There is a small bit of funding for critically evaluating data and examining literature. Any new things would have to be sent to the center.

Dr. Cowell pointed out that they are open to alternatives in this study, and this structure is just proposed for purposes of discussion.

It was felt that the funding of money in many small grants, i.e. mini-grants, paying 1/2 the cost of software that researchers in the university would want. If there was an ensured market it would seem that computations and decisions being made by groups of users would determine what was good and what was bad, and would also help determine if this would be profitable. There would also be public reports on quality of the software. This would make everything possible - it would be similar to a consumer report. It could then be phased out after a few years. This is a discrete situation rather than a long life situation. A university would have to put money for software into the project. This idea seemed to indicate that there would be a good market for quality software.

It was generally felt that universities might not want to hold off for a few years to wait for higher quality software. You can test the software and see where you spend your time, but you need to define the scientific community and the market for the software first. In this type of operation, you need feedback mechanics from users - revisions, expansions, etc., and also to keep the center up-to-date. You cannot allow programs to evolve in a random way.

Attendees at NSF Conference, March 30-31, 1972

1.	T.J. Aird	Purdue University, Lafayette, Indiana
2.	Charles A. Babendreier	National Science Foundation
3.	Melvin R. Baron	Paul Weidlinger Consulting Engineers, New York
4.	E.L. Battiste	IMSL, Houston, Texas
5.	Alan K. Cline	National Center for Atmospheric Research
6.	W.J. Cody	Argonne National Laboratory
7.	Wayne R. Cowell	Argonne National Laboratory
8.	W.J. Dixon	University of California, Los Angeles
9.	Mark Eisner	National Bureau of Economic Research
10.	Richard Fairley	University of Colorado, Boulder, Colorado
11.	Steven J. Fenves	Carnegie-Mellon University, Pittsburgh, PA
12.	Lloyd D. Fosdick	University of Colorado, Boulder, Colorado
13.	John M. Gary	National Center for Atmospheric Research
14.	William Hetzel	University of North Carolina, Chapel Hill
15.	Yasuhiko Ikebe	The University of Texas at Austin
16.	Seymour Jeffery	National Bureau of Standards
17.	C.L. Lawson	Jet Propulsion Laboratory, Pasadena, California
18.	M. Stuart Lynn	Rice University, Houston, Texas
19.	Gregory A. Marks	University of Michigan, Ann Arbor, Michigan
20.	N.C. Metropolis	Los Alamos Scientific Laboratory
21.	Cleve B. Moler	University of Michigan, Ann Arbor, Michigan
22.	Leon Osterweil	University of Colorado, Boulder, Colorado
23.	John R. Pasta	National Science Foundation
24.	John R. Rice	Purdue University, Lafayette, Indiana

25.	Gerald Ruderman	National Bureau of Economic Research
26.	R.L. Schiffman	University of Colorado, Boulder, Colorado
27.	Walter L. Sadowski	National Bureau of Standards
28.	Shayle Searle	Cornell University, Ithaca, New York
29.	Sally Sedelow	University of Kansas, Lawrence, Kansas
30.	Walter Sedelow	University of Kansas, Lawrence, Kansas
31.	Gordon R. Sherman	National Science Foundation
32.	Roberta Smith	University of Colorado, Boulder, Colorado
33.	Roland Sweet	University of Colorado, Boulder, Colorado
34.	Frederick Thompson	California Institute of Technology, Pasadena CA
35.	Richard Venezky	University of Wisconsin, Madison, Wisconsin