INTERACTIVE PROCESSING OF GEOPHYSICAL DATA FROM ARCHAEOLOGICAL SITES

Irwin Scollar

Rheinisches Landesmuseum
Bonn, W. Germany.

In 1959, the Laboratory for Field Archaeology of the Rheinisches Landesmuseum Bonn began the development of a system for archaeological prospecting based on a differential proton magnetometer developed there. This instrument, in several different versions, was later connected to an automatic punched paper tape recording device and the whole installed permanently in a minibus. After a number of years, the automatic recording equipment was replaced with a hardwired mini computer and tape unit, and differential operation was obtained by subtracting readings coming simultaneously from two proton magnetometers. The various stages in the development of the system have been described in many publications[1]. Readings can be now made at a rate such that a 20x20 meter square is covered at 1 meter intervals (441 values including the edges, the line start commands and all other control values) in 20 minutes or less by one man in the bus and one transporting the sonde. Apart from entering the square number at the start, all operations are fully automatic and nothing need be written down by the operator. He merely observes the readings and occasionally repeats one if he feels that it is necessary. Position control of the sonde is simply by means of a set of cloth measuring tapes lying on the ground. Two-way communication between bus and sonde is by loudspeaker and the operator tells the sonde carrier the number of the next point to be measured (which appears on his control panel automatically). Very large sites can be surveyed in relatively short times, weather permitting. It is usual to obtain 40,000 or more readings for most sites, and one site to be discussed has provided nearly half a million. Evidently a data processing system is required if best use is to be made of the information.

It has been explained elsewhere that a data processing scheme is not only important because of the amount of information, but also it permits us to use digital filtering and non-linear treatment.[2] This lets us separate information of archaeological significance from that due to random variation in soil magnetism, surface iron etc. The mathematics of the treatment technique have been dealt with extensively in several papers written for specialists, and the interested reader should refer to them.[3]

After data treatment, we are faced with a very large array of numbers whose meaning must be made plain to non-mathematically minded archaeologists like those of our monuments protection branch at the Rheinisches Landesmuseum. The information must be accurately incorporated in large scale maps (1:1000) and turned over to the planning authorities in order to block planning permission on scheduled sites. Alternatively, the maps serve as the basis for an excavation if required. In the early 1960's a pseudo-half-tone presentation was evolved which we christened a "dot density" plot. It has found wide favor. In this technique a number of dots equal to a desired numerical value is displaced a bit relative to the coordinate point of the value using a random number generator giving linear random values in the x and y directions over a small interval. In the beginning plotting was done by hand, but this was replaced in 1963 by a large flat-bed EAI analog machine, tape driven. Later, a Calcomp 584 drum machine was used, and since 1971 a Calcomp 1670 microfilm plotter has been available. Plots made using the dot density technique appear to make very good use of the eye's ability to detect patterns in random arrays of points.

The program architecture for data treatment of this kind was presented[4] in summary form in a now rather outdated publication. The scheme described there, represented the state of our technique in early 1966, using an IBM 1410-7090. In late 1971 this splendid old machine was regretfully retired and replaced by an IBM 370/165. In 1973 the 165 was upped to a 168. A library including some 60 applications and utility programs written in assembler and Fortran II had to be completely written anew. Advantage was taken of the new hardware in doing this.

The configuration of the current machine includes the following. A 3 megabyte memory with half microsecond cycle time and cache store using semiconductor, a $1.6x10^9$ byte disk store, 3 fixed head disks for system, library and time sharing swapping, 8-1600/800 bpi automatic changeover tape units and a host of minor peripherals, including 42 local terminals and up to 128 external users, plus several attached computers make this one of the largest complexes available in western Europe. It is owned by the University of Bonn and managed by the Gesellschaft fur Mathematik und Datenverarbeitung, with whom the author had a consultancy up to the beginning of 1974. The 370/168 is a virtual memory machine, allowing the user up to $16x10^6$ bytes of virtual storage, but at the time of writing the center operates under OS/MVT since the VS2 operating system is not considered sufficiently debugged. At the moment, two time sharing partitions of 120K & 200K bytes, plus about 150K for the telecommunications access program serve the users interested in employing this method. The rest of the memory is occupied by batch users and the very large core resident system. The central processor, which is extremely rapid, deals with these tasks according to a complex priority algorithm which is supposed to give maximum throughput. The time sharing partitions, being fixed in size, allow at most 180K bytes of effective memory for the user's program and data, 20K being lost in overhead of various kinds. Response times are reasonably good as long as not more than about 30 users are logged on the system, but deteriorates rapidly when the number rises above this. This is largely due to jams in the queues waiting for access to the disks. The program package for the treatment of magnetic data which has grown a great deal since the last publication, was completely rewritten to take advantage of the time sharing scheme.

There is a considerable advantage in using an interactive system for the processing of complicated data. If properly written, the program can be halted at suitable places and intermediate results examined almost immediately. If necessary, control values can be altered and calculation restarted, or the entire process repeated as desired. The new programs have all been written in Fortran IV so as to be independent of changes in machine configuration and manufacturer. With a large fast machine the somewhat more inefficient modules that result are still quite rapid enough. The programs contain break points at suitable places for complete interaction if desired. They may also be run as batch jobs with the needed control data supplied as a separate data set. The programs are arranged for use as an overlay structure if required, so that storage requirements are reduced at the cost of very small increase in computation time. In batch this increase is not noticed at all, and in time sharing it is hardly perceptible. Much use is made of the disks for intermediate storage so that everything can fit nicely into the time sharing partition.

The requirements for complex disk allocation, overlay structured pre-compiled load modules, format free input of control data at break points and other features are rather complex. Fortunately, IBM has provided an extremely flexible time sharing command language. Most of our effort was shifted from the Fortran problem to the programming in the TSO Command Language. Readers who have used 360/370 type installations with anything but catalogued procedures certainly remember the difficulty which they had when they encountered the Job Control Language (JCL) for the first time. Many of these difficulties are happily smoothed over in the TSO Command Language which, although executing nearly all of the functions of JCL for the time sharing user, appears in a much more rational and easy to use form. The command language allows construction of command procedures which can modify themselves according to the progress of the work and which can receive parameters from the terminal or from external data sets. Whole sets of complex programs can be executed with a single request word, and the skilled user can make up his own command language which is oriented toward his problem.

Standard Fortran IV requires strict observation of format for data entry. This is inconvenient when entering control information from a display terminal. Extended Fortran IV frees the user from this necessity, and all input-output subroutines have been compiled using this language. If the programs are to operate on a machine which does not allow an extended compiler, then format bound standard input-output routines can be selected. These are usually used if work is done in batch with entry from punched cards. The Linkage editor, required to hook up this complex with the many subroutines, the main programs and with the various computer and private libraries, is quite slow when using time sharing. Therefore it proved better to create fully compiled load modules and store them permanently on disk files. These are called up by the command language as required and executed immediately since they are in machine language. A rather wasteful use of the disks perhaps, but very much more comfortable. All the needed temporary and permanent disk storage is previously allocated in a command procedure executed at the beginning of a TSO session. The overall scheme is shown in Figures 1 & 2. Unfortunately the mix of Fortran compilers, libraries etc. require record formats, types and block lengths which are not supported by the standard commands at our disposal (variable length spanned blocked records of small size) and a certain amount of ingenuity was needed to bypass these restrictions.

Field paper tapes input to the machine via an off-line conversion unit (MDS) which change over to magnetic tape. This tape is run as a batch job onto a catalogued disk. The disk file serves as input for a program which checks the raw data for formal errors of simpler kinds and produces card images on disk as output, together with hard copy at the terminal or at one of the line printers. The data is edited with the standard editing program in time sharing, and a checking program run via a command procedure. This program performs higher order checks, sees to it that the measured squares are really in the right place and orientation, if necessary corrects for errors in the placement of the fixed

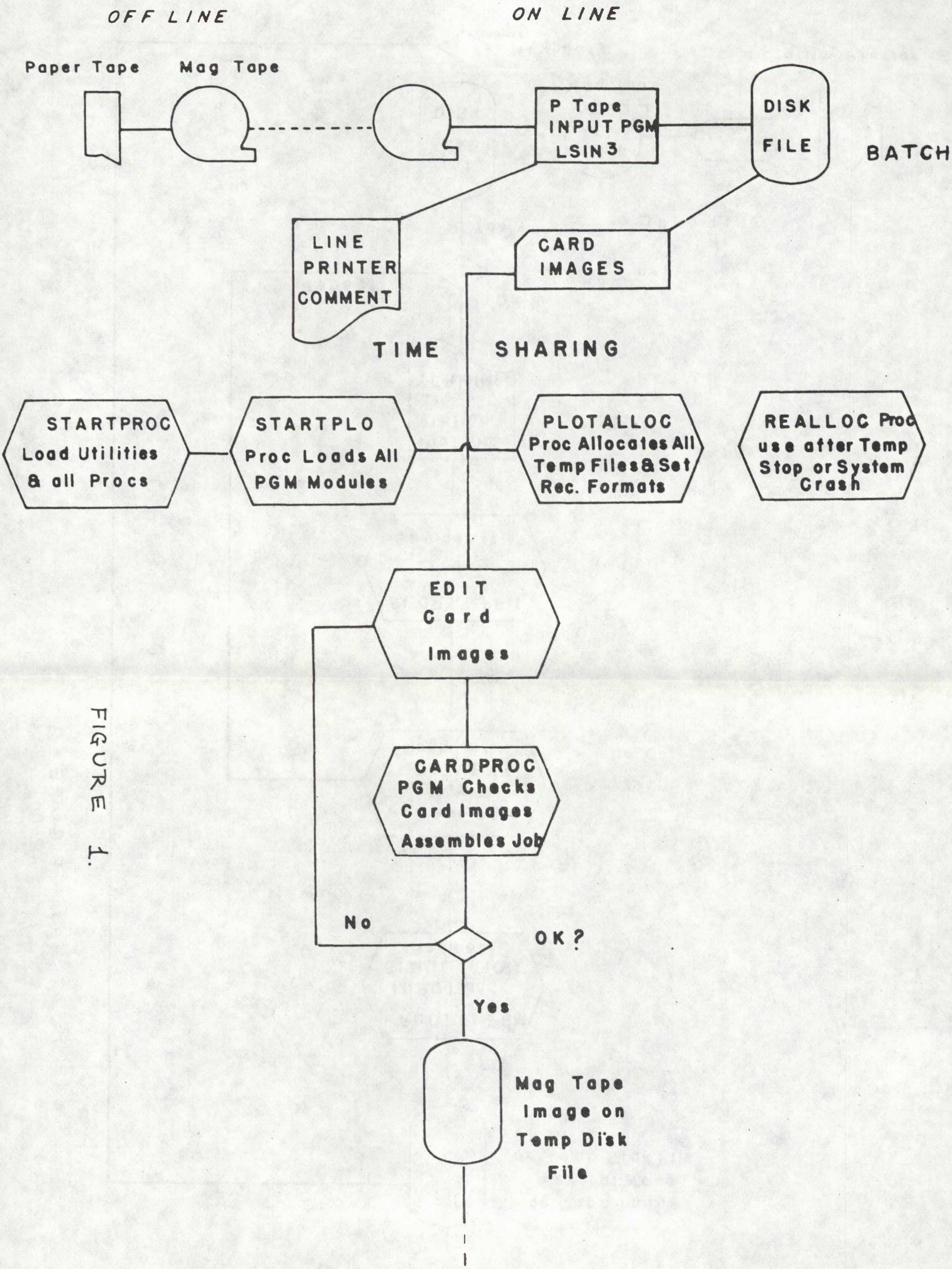
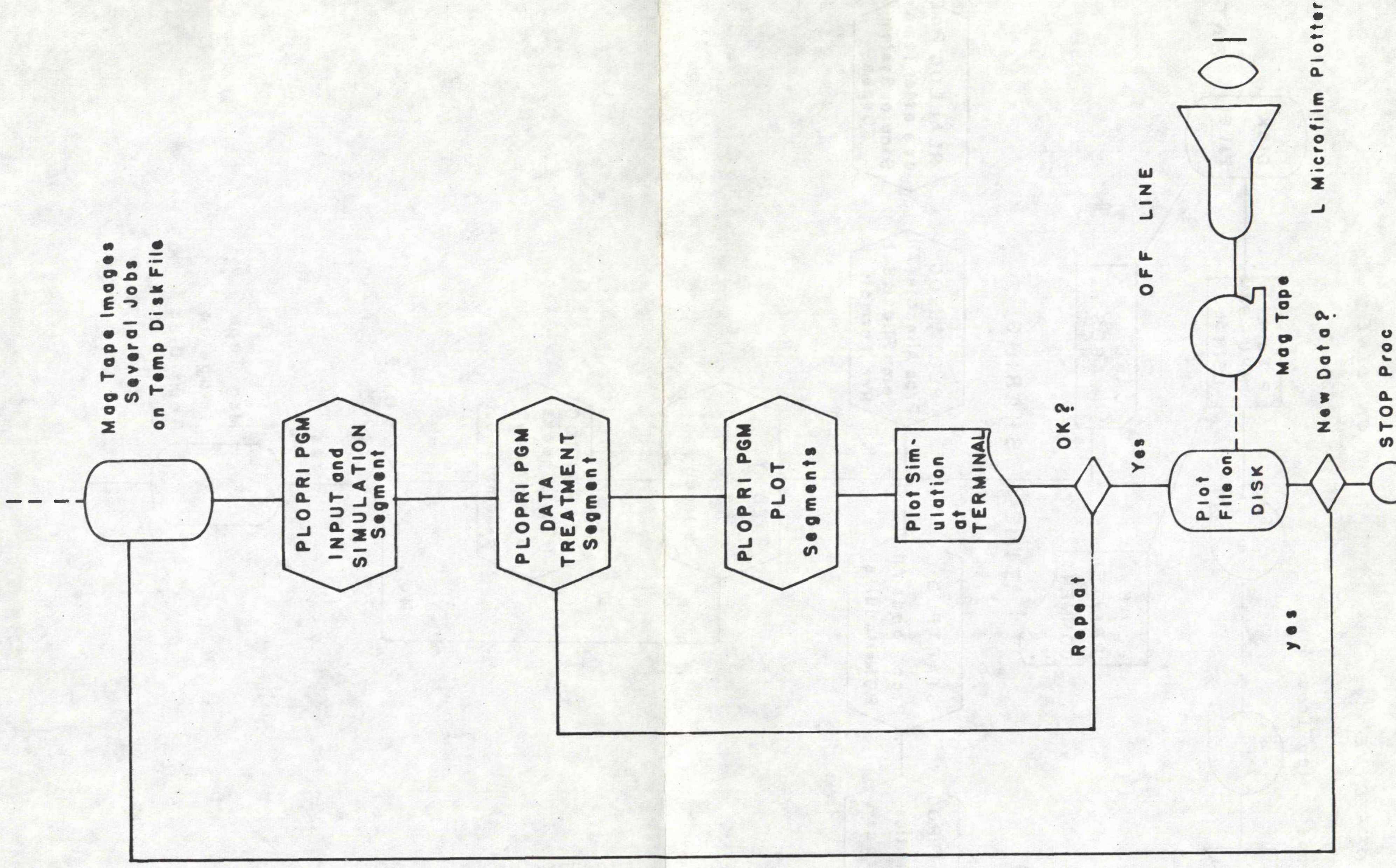

OFF LINE
ON LINE
Paper Tape
Mag Tape
P Tape INPUT PGM LSIN3
DISK FILE
BATCH
LINE PRINTER COMMENT
CARD IMAGES
TIME SHARING
STARTPROC Load Utilities & all Procs
STARTPLO Proc Loads All PGM Modules
PLOTALLOC Proc Allocates All Temp Files & Set Rec. Formats
REALLOC Proc use after Temp Stop or System Crash
EDIT Card Images
CARDPROC PGM Checks Card Images Assembles Job
OK?
No
Yes
Mag Tape Image on Temp Disk File

FIGURE 1.



Mag Tape Images
Several Jobs
on Temp Disk File
PLOPRI PGM
INPUT and
SIMULATION
Segment
PLOPRI PGM
DATA
TREATMENT
Segment
PLOPRI PGM
PLOT
Segments
Plot Sim-
ulation
at
TERMINAL
OK ?
Yes
Repeat
Plot
File on
DISK
OFF LINE
Mag Tape
L Microfilm Plotter
New Data ?
yes
STOP Proc

FIGURE 2.

reference sonde (by fitting a plane or other surface to each square of data and then subtracting) and assembles a sequential data set with all the data in one big matrix. If it aborts at any point, then one returns to the edit program and corrects the errors detected.

The main data treatment and plotting program also contains the checking and paper tape programs as subroutines in slightly altered form. If required, it can be executed in batch running either from mag tape, cards, paper tape, card images on disk, tape images on disk or almost any combination of these. In theory one could run in TSO or batch from field paper tape right through to plot, but in practice things would certainly break down somewhere along the line. Usually the main program is run from the output of the checking program in time sharing. The various options of the input phase are all separate overlay segments with branches as required. The filtering segment is the largest part of the module, and finally there are two output segments, of which more below. The entire package occupies the equivalent of 16,000 punched cards in Fortran, but fortunately never needs to be used all at once in that form.

Following the input phase, filtering is carried out preceded and followed by non-linear treatment. The result remains in storage at the end of the operation for the next segment to operate upon. When we speak of something remaining in storage in the context of the time sharing "environment" this has to be understood in terms of the appearance of things to the user. In fact the data and the program of the user are in main memory for but a very short time, and afterward they are swapped out against those of another user onto one of the fixed head disks. The user is not directly aware of the process and he need not concern himself with it. However, when very many users are on the system, the continuous swapping in and out of actively computing programs (users who are merely thinking about what they are going to do receive very little time indeed) reduces the available number of machine cycles to a given user and computation slows down. If other users are merely doing a lot of input-output then a program with high computation content can have as much as a quarter of the central processor time with correspondingly quick results. The time sharing users stand in three imaginary circular queues and the priority allotted to each user is a function of the amount of straight computing and input-output operation done the last time round the queue. Since input-output is favored over computation, high computation programs are penalized. However, over-frequent access to the disks also increase response times greatly, since the request to use these devices waits in a queue or queues to which all machine users including all core resident batch jobs have access. Considering the complexity of it all, it is rather surprising that the scheme works. Considerable juggling of program structure, block size, record length, type of intermediate disk data, and terminal use is needed for fastest response time. The system is very opaque to the user and the right mix had to be found by trial and error. We are not entirely happy with the result, and there remains much to do in this area provided the operating system is not changed in the near future!

After digital filtering, which usually takes about 30-40 seconds of CPU time, and anything from 120 to 1500 seconds of 'real' time, the program requests the user to tell it what to do next. A simulation of the dot density plot on a typewriter terminal or on a display is available. This is usually used first. If the results are satisfactory, then the data can be passed to the segment which contains the plot and dot density programs. Contour or perspective presentation is also available, but seldom used for real data.

The actual commands required to drive the microfilm plotter are accumulated along with those from other users on a disk, and when this is full enough, a complete microfilm is exposed automatically, removed manually and developed, and turned over to the users, usually the next day.

The control parameters for the interactive program are usually stored as a separate data set. This is referred to during execution which then becomes a 'no hands' operation in time sharing. It is like running in batch with a set of control cards. If only a few control values are to be changed, this is the preferable method, since all that is needed is an editing step prior to main computation.

A typical result provided by the new program package is shown in Figure 3. Here we see some of the magnetic anomalies measured at the Colonia Ulpia Trajana, near Xanten on the lower Rhine. The image is enlarged directly from the microfilm, and a 200x200 meter area has been treated in one go. The small crosses indicate 20 meter intervals. Three such images have been put together and reduced in scale, as shown in Figure 4. About 110,000 magnetic field readings were required. These were obtained by the automatic recording magnetometers in about six weeks working time allowing for bad weather in the autumn of 1971. The image shows an area of about 550 meters long and of irregular width varying between 100 and 200 meters, as dictated by houses and roads in the area. A late 4th century pair of ditches cut through the middle empire town and surround part of the Forum visible at the lower right. The older street boundaries and structures are visible at the left, and a disturbance of unknown date has partly destroyed the 4th century ditches in the upper center. Results of this quality are perhaps possible with other evaluation techniques. They are hardly likely.

<u>References</u>

1 I. Scollar, Magnetic methods of archaeological prospecting-advances in instrumentation and evaluation techniques. Phil. Trans. Roy. Soc. London A 269 109-119 1970

I. Scollar, A wide range, differential proton magnetometer with direct readout in gammas. Prospezione Archaologiche, 5, 1970, 59-66.

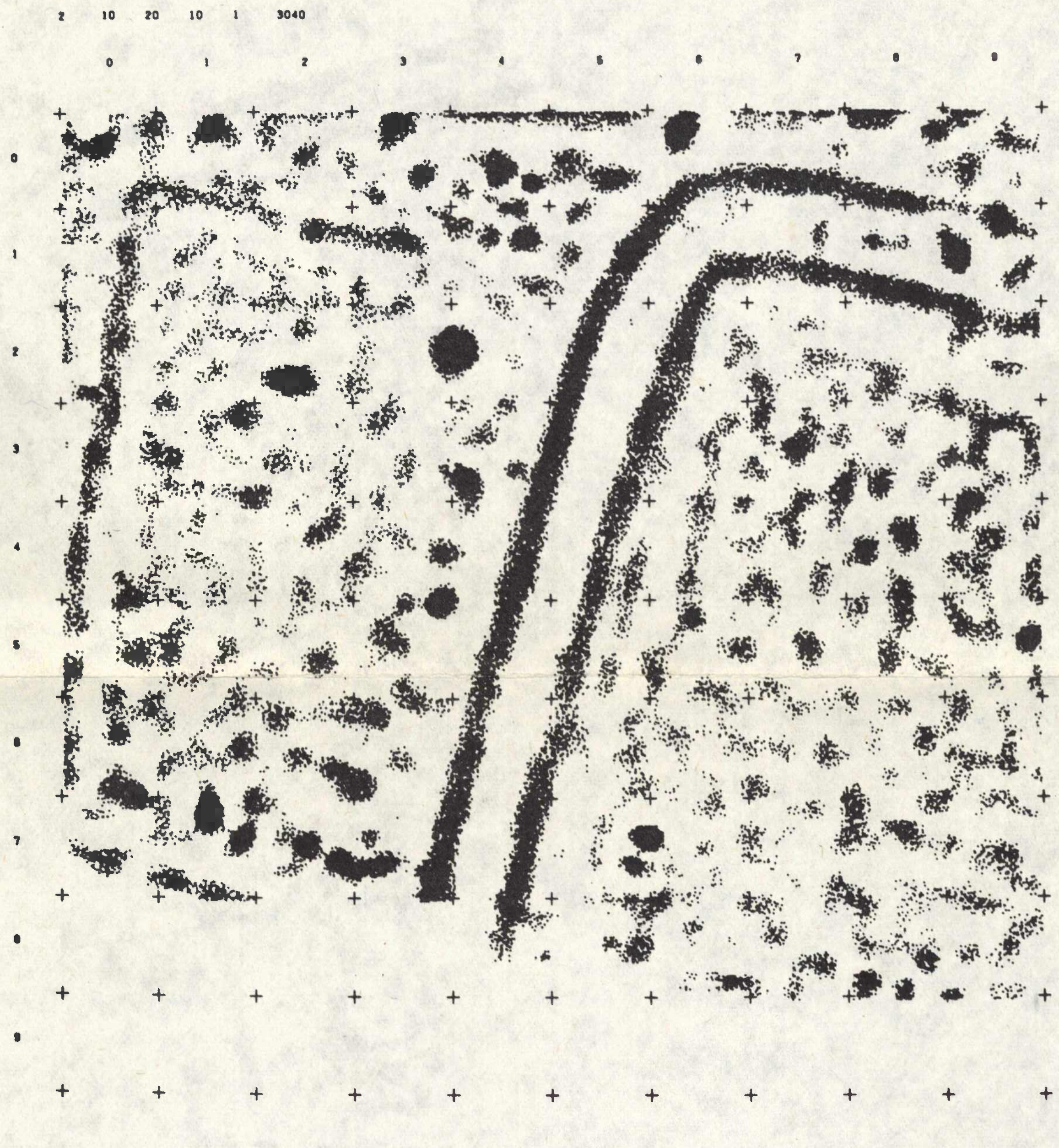
2 I. Scollar & F. Kruckeberg, Computer treatment of magnetic measurements from archaeological sites. Archeometry, 9, 1966 61-71

I. Scollar, Some techniques for the evaluation of archaeological magnetometer surveys, World Archaeology 1, 1969, 77-89.

Xanten, Kreis Moers, Colonia Ulpia Trajana, Eastern Half.

FIGURE 4.

FIGURE 3.



Microfilm Enlarged, 200x200 magnetic readings,treated,
Colonia Ulpia Trajana, Xanten Kr. Moers

3 D.I. Black & I. Scollar, Spatial Filtering in the Wave vector domain. Geophysics, 34, 1969, 916-923.
I. Scollar, Fourier transform methods for the evaluation of magnetic maps. Prospezione Archaeologiche 5, 1970, 9-41.
D. Gubbins, I. Scollar, P. Wisskirchen, Two dimensional digital filtering with Haar and Walsh tranforms. Annales de Geophysique, 27, 1971 85-104.

4 I. Scollar, A program package for the interpretation of magnetometer data. Prospezione Archaeologiche 3, 1968, 9-18.

Acknowlegement

The author is indebted to the Gesellschaft für Mathematik und Datenverarbeitung for the unlimited free computer time placed at his disposal. Without the help of Herr H. Rieger of that organization to guide the author through the thickets of the IBM operating system, this work would never have reached a successful conclusion. For many fruitful discussions with him and his colleagues, the author is most grateful.