AD-A080 359 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 20/4
THE GRAPHICAL DISPLAY OF MULTI-DIMENSIONAL AERODYNAMIC FLOW FIE--ETC(U)
DEC 79 E P AMBURN
UNCLASSIFIED AFIT/GCS/MA/79D-1 NL

1 OF 2
AD
A080359

THE GRAPHICAL DISPLAY OF MULTI-DIMENSIONAL AERODYNAMIC FLOW FIELD DATA

(9) Master's THESIS,

AFIT/GCS/MA/79D-1

Elton P. Ambur
Capt USA

UNITED STATES AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY
Wright-Patterson Air Force Base, Ohio

AFIT/GCS/MA/79D-1

(6) THE GRAPHICAL DISPLAY OF MULTI-DIMENSIONAL AERODYNAMIC FLOW FIELD DATA

(9) Master's THESIS,

(14) AFIT/GCS/MA/79D-1

(7) Elton P. Hamburn, Capt USAF

(11) Dec 79 (12) 112

1980

Approved for public release; distribution unlimited.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/GCS/MA/79D-1	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) The Graphical Display of Multi-Dimensional Aerodynamic Data		5. TYPE OF REPORT & PERIOD COVERED MS Thesis 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Elton P. Amburn Captain USAF		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Institute of Technology (AFIT-EN) Wright-Patterson AFB, Ohio 45433		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Flight Dynamics Laboratory (AFFDL-FXM) Wright-Patterson AFB, Ohio 45433		12. REPORT DATE December 1979 13. NUMBER OF PAGES 109
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Approved for public release; IAW AFR 190-17

JOSEPH B. HIPPS, Major, USAF
Director of Information

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer Graphics, Computer Generated Movies, Automatic Parsing, Table-Driven Parsers

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The Air Force Flight Dynamics Laboratory has solved aerodynamic problems using sophisticated numerical techniques. Large volumes of data are created during the numerical solution of a problem, and the purpose of this project was to build a computer graphics software system to display this data. The system is controlled by using a command language which is interpreted by a table-driven parser. The primary output of the system is computer generated movies of the aerodynamic data.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

AFIT/GCS/MA/79D-1

THE GRAPHICAL DISPLAY OF MULTI-DIMENSIONAL AERODYNAMIC FLOW FIELD DATA

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air Training Command

in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

by

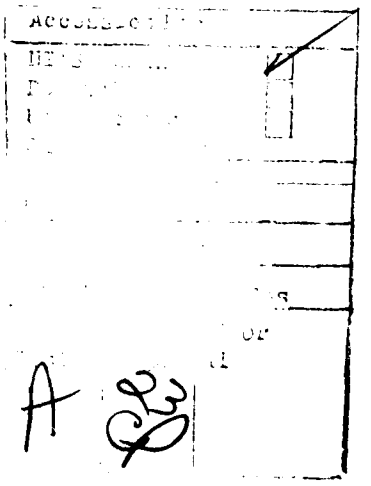
Elton P. Amburn, B.A.

Capt USAF

Graduate Computer Systems

December 1979

Approved for public release; distribution unlimited.



Preface

I have been interested in computer graphics for several years, and I particularly enjoyed creating computer generated movies, a fairly recent addition to the computer graphics repertoire. There were several steps involved in the creation of film during this project, and coordinating and implementing the different steps in this process were at times tedious. However, the results were very gratifying.

During the design and development of this computer graphics system, I received help from many different sources. The Air Force Flight Dynamics Laboratory (AFFDL) sponsored this project and several people there deserve my thanks. Dr. Will Hankey and Dr. Joe Shang were the AFFDL scientists that sponsored this project, and they served as my points of contact within the laboratory. These gentlemen kindly answered many questions, provided direction, guidance, and test data. Major Bill Golbitz, also a scientist at the AFFDL, was also very helpful during this project; he provided some realistic three-dimensional test data at a critical time. Another AFFDL member that was a great help throughout this project was Major Mike Wirth. He expressed an interest in this project from its inception and was active throughout. His computer expertise and scientific background combined to make him a valued consultant at several times during the design and implementation of this system. The sponsorship and support of the AFFDL were greatly appreciated.

In addition to thanking AFFDL personnel I also want to thank Capt Roie Black, my thesis advisor. He worked as an AFFDL scientist

before he became an AFIT instructor, and that experience as well as his understanding of computer science allowed him to have an exceptionally thorough understanding of this project.

Finally, there are two others that I wish to thank for their help during my time at AFIT. These two encouraged me, helped me, and stuck by me the entire time I was at AFIT. The first is my wife Patty. She helped me keep going through many hard times and good times. I love you, Patty. The second one I wish to thank is the most important one of all, Jesus Christ. He is my Savior, Lord, and soon coming King, and I owe Him everything!

CONTENTS

Abstract

Aerodynamic scientists, engineers, and researchers have been working for a long time in the analysis of fluid flow, and they have often used water tunnels and wind tunnels to visualize a fluid (water, air, etc.) flowing over an object. While tunnel experimentation is one of the major traditional techniques used to evaluate and visualize fluid flow fields, recently there have been some advances in both numerical methods and computers designed for scientific work that have provided new possibilities. The Air Force Flight Dynamics Laboratory has been using the best scientific computers and new numerical techniques to evaluate the Navier-Stokes equations; these equations mathematically describe a flow field. Scientists at this laboratory have become increasingly involved with numerical solutions of aerodynamic problems. Many of these solutions require several hours of computer time and generate large amounts of raw data. However, the volume of data was often so large that it was difficult to completely understand the solution, and it was impossible to visualize the fluid flow that had been mathematically described by examining large stacks of computer printouts.

The subject of this thesis was the display of numerically generated aerodynamic flow field data using computer graphics, and to achieve this goal a computer system named CINEMA was developed. By examining the techniques used in tunnel experimentation it was possible to design the graphics output of CINEMA such that it would be possible to compare experimental results with theoretical results. Some of the aerodynamic problems solved by the laboratory were time-

dependent problems. The best way CINEMA could display the time-dependent solutions was with movies. Therefore, the resulting system accepts the numerically calculated flow field and uses computer graphics to generate movies that allow the scientists to visualize their theoretical results. Although CINEMA is tailored for aerodynamic flow field problems, it is sufficiently flexible to be used in other areas.

I INTRODUCTION

For many years scientists and engineers doing research work in aerodynamics have used water tunnels and wind tunnels to visualize fluid flow. Some very sophisticated techniques have been developed for use with these tunnels; however, these experimental techniques are expensive. The tunnels themselves are costly to build and operate, and the scale models used in tunnels are difficult to construct. There is also a lot of preparation involved in performing a tunnel experiment, and as with any experimental technique there are some limitations on the data collected. Almost all of the data from tunnel experiments is two dimensional (2-D); there is very little three dimensional (3-D) tunnel data. Most tunnel experimentation provides information about the flow field on the surface of the object although some optical techniques do provide shock wave visualization for high speed flows. Tunnel experiments produce valuable data and they are certainly one of the major techniques used to analyze aerodynamic flows. Recently a new capability has been developed, the numerical solution of aerodynamic problems. The Navier-Stokes equations mathematically describe fluid motion but they are a rather complex set of equations. Twenty or thirty years ago these equations could not be solved using existing numerical methods on the best computers. However, both numerical methods and computers have been improved and now these equations can be solved numerically. This fact is starting a whole new era in aerodynamic research.

The Computational Aerodynamics Group (CAG) of the Air Force

Flight Dynamics Laboratory (AFFDL) has been using finite-difference techniques to solve the Navier-Stokes equations. They use some of the best scientific computers available to generate solutions to various problems. Some of their problems are 2-D while others are 3-D; some are time-dependent (i.e., the solutions change in time) while others are time-independent. Often these numerically generated solutions represent large amounts of raw data and it is difficult to comprehend the solution as a whole. It is particularly hard to understand the solution to a time-dependent problem since numerically solving the equations only provides tables of the numerical values of aerodynamic variables such as air density, temperature, and velocity. Examining tables of changing values is not an easy way to understand what is happening to the flow field. The CAG continues to model and solve more difficult problems (i.e., more complex geometries and 3-D, time-dependent problems). All this adds to the complexity of the problem and increases the volume of data. The ability to display all this data as a unit would greatly increase its usefulness.

The purpose of this thesis was to design and build a computer system that could display aerodynamic data. That computer system, named CINEMA, is a computer graphics software system. It executes on the Aeronautical Systems Division (ASD) Control Data Corporation (CDC) computers and generates a file of graphics data. This graphics data can then be displayed using several pieces of graphics hardware. CINEMA reads and processes the user's data; its actions are directed through the use of a command language. It is designed to produce computer generated graphics output ranging from high quality

plots to movies of aerodynamic flow field data. Figure 1 shows the organization of the CINEMA system.

The majority of the work in developing the CINEMA system was in building the software. There are three groups of routines in the system: (1) command language processing routines, (2) application routines, and (3) graphics routines. A command language and companion language interpreter were developed specifically for CINEMA to provide the overall system control functions. The application routines are called by the language routines; they read and process flow field data and prepare it for display. After the data is ready to be displayed the graphics routines are called to create the computer graphics output.

The command language and language processing routines were built using a software package from Lawrence Livermore Laboratories (LLL) (11). With this package a language can be defined and described to the system and then a table-driven language interpreter can be built. Through the interpreter, commands are read, error-checked, and executed one at a time. When a legal command is received the action requested by that command is often accomplished by calling one of the application routines.

The application routines use the raw flow field data to create graphical output. These routines are capable of using aerodynamic flow field data and generating a particle plot (place a particle in the flow and observe its motion) in 2-D or 3-D or generating a 2-D contour plot of data values. After the data has been prepared for plotting, the application routines call graphics routines to actually display the results. The graphics routines used are all part of the

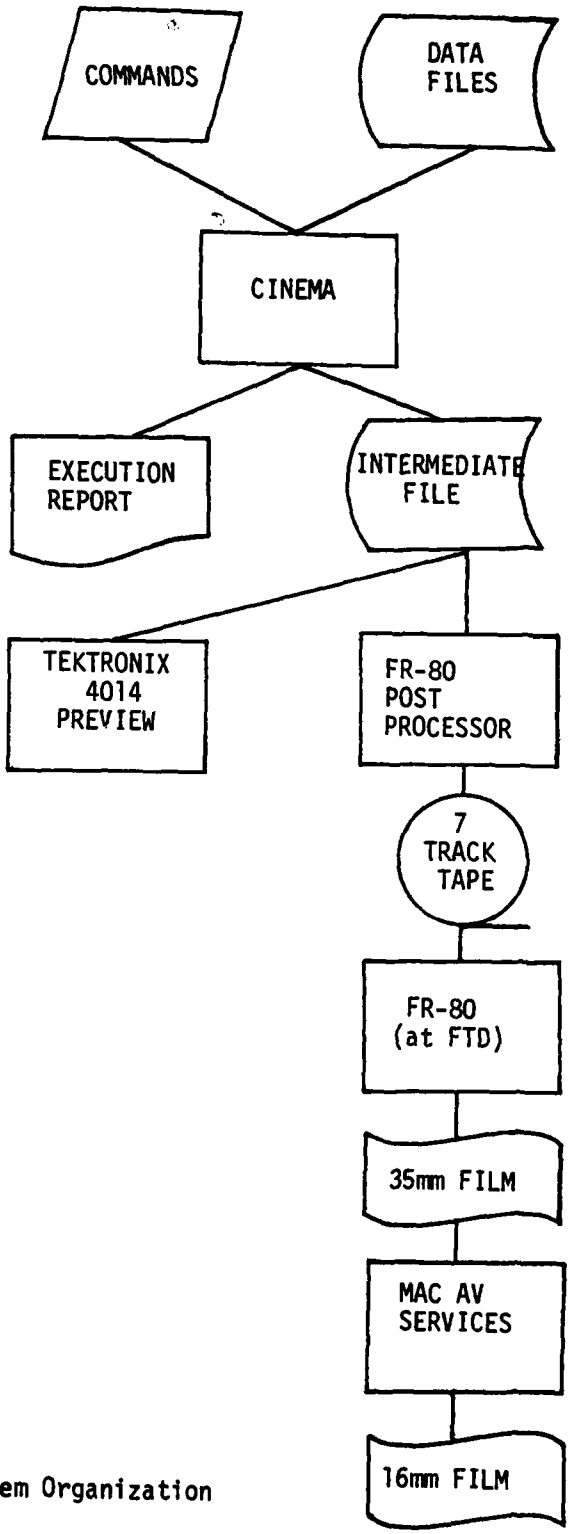


Figure 1. CINEMA System Organization

Display Integrated Software System and Plotting Language (DISSPLA) package, a FORTRAN callable set of graphics routines.

The DISSPLA graphics routines are used to build an intermediate file that contains device independent information. To display the graphics output the DISSPLA package contains post-processors for several pieces of graphics hardware. The TEKTRONIX post-processor displays the graphics data on a TEKTRONIX 4010 or 4014 vector graphics storage scope. The FR-80 post-processor formats the data onto a file (usually a 7-track tape) that can be used to generate 35mm film. The FR-80 is a film/microfilm recorder, and it too is a vector graphics device.

Computer generated movies are only about 20 years old (9:46) but this is the best way to graphically display the numerically calculated flow field data (especially the time-dependent data). All the needed equipment and film processing capabilities are available on Wright-Patterson Air Force Base (WPAFB). Color is often used in computer graphics to provide visual emphasis on portions of a complicated picture. To effectively display the complex flow field data calculated by the AFFDL engineers it was decided that CINEMA should produce color film. The problem with this idea is that the FR-80 on WPAFB is not capable of producing color film, only black and white. To produce color film the following scheme is used. CINEMA will allow one of six colors to be specified: red, blue, green, cyan, yellow or magenta. Red, blue, and green are the three additive primary colors. Cyan, yellow, and magenta are each the combination of two of the primary colors. To produce a single color frame, up to three black and white frames are produced -- one for each of the

three additive primary colors. These black and white frames are exposed with the appropriate color filters and then superimposed. In this way a single color frame is created from the black and white frames.

The remainder of this document describes the various parts of the project. Chapter II explains the requirements of the thesis sponsors, the Computational Aerodynamics Group of the AFFDL. Chapter III outlines the systems analysis done before the software was designed. Chapter IV explains in detail the software design. Chapter V presents the results of this project and recommendations for future work. The appendices provide additional information about parts of this system. Appendix A describes the LLL language processing package that is used as a command language interpreter. Appendix B defines the command language in Backus-Naur Form (BNF). Appendix C is a user's guide that contains command formats and notes on the use of CINEMA. Finally, Appendix D is a programmer's guide that is designed to help in the maintenance of the system.

II REQUIREMENTS DEFINITION

Within the past 10 to 20 years advances in computer technology and improvements in numerical techniques have made it possible for more and more complicated problems to be solved using computers. Aerodynamic engineers at the AFFDL have taken advantage of these new capabilities and have been solving aerodynamic fluid flow problems on very large computers. They use computer networks to gain access to the best engineering computers and typically, their jobs will run for hours on a CDC STAR or a CRAY-1 computer and generate large amounts of data. AFFDL engineers use very sophisticated finite-difference techniques and generate numerical solutions to problems that could not be solved using the best computers available just a few years ago. They have already solved several aerodynamic problems using these techniques, and these problems have had a variety of characteristics. Some problems have been time-dependent and others have been time-independent, and many have been 2-D while others have been 3-D. However, regardless of the type of problem, the solutions have always represented large amounts of raw data. The purpose of this thesis was to design and build a system for the AFFDL that could accurately display these large amounts of data using computer graphics.

The AFFDL requirements fell into three areas: (1) the graphical display of large amounts of data, (2) the necessity of comparing numerical results with experimental results, and (3) the need for a flexible system that could be modified. Each of these areas is discussed in detail below.

Because the Computational Aerodynamics Group uses finite-difference techniques to solve aerodynamic problems, large quantities of data can be generated. To solve these problems a finite-difference mesh is created, and at every intersection point in the mesh, values for several aerodynamic variables are calculated (for example: velocity, air density, temperature, etc.). Increasing the number of mesh points increases the accuracy of the solution, and this is especially true if the mesh is made very dense where the aerodynamic variables are rapidly changing. But, all of this information must be stored in the computer and limits on memory size restrict the density of the mesh. If the problem is time-independent then the variables must be calculated only once since they are static. However, if the flow changes with time then these variables must be calculated for each "time-slice" of data. If the mesh is fine then clearly a 3-D, time-dependent solution can represent a very large quantity of data. In fact, some results have been stored on multiple reels of magnetic tape because of the volume of data.

The fact that the flow field data could change with time presented an interesting problem. Not only did the engineers need some way of displaying large amounts of data, but since the data could be time-dependent they needed some way of presenting the time-dependent aspect of the solution. That particular characteristic of their data presented a challenge since the display of a 3-D, time-dependent solution was essentially a 4-D problem.

Determining what technique or techniques to use to display the data was one of the most important parts of the project. It was necessary that CINEMA clearly and effectively display the numerically

calculated flow field, and in that way the engineers could visualize their results. This would allow a more complete analysis of both the problem and the solution. AFFDL engineers have frequently needed to compare their calculated results to experimental results, and they indicated that it was essential that CINEMA provide for this type of comparison. This meant that the displays created by CINEMA must present the numerically calculated data in a manner similar to that used to present experimental data. This was an interesting requirement because there exists in a numerical solution much more information than is available to an experimenter.

Many effective techniques are currently being used to visualize flow fields. Merzkirch (8) does an excellent job of describing flow visualization techniques. Most experimental work in flow visualization is accomplished in a wind tunnel or a water tunnel. Some of the more common techniques are listed below with a brief description:

Vector Plots:	2-D plots of velocity vectors within a flow field; both magnitude and direction of flow are shown
Oil Trace Photographs:	A thick, specially formulated oil is used to coat a surface. The deformation of the oil after being placed in a wind tunnel provides information about flow field direction close to the surface of the object.
f(x) vs x plots:	Very common technique. A 2-D plot of an aerodynamic variable versus location.
Streamline Photographs:	Seed the fluid with small particles; observe and photographically record their motion.
Schlieren Optical Techniques:	Laser-Schlieren Holography is a sophisticated photographic process for displaying lines of constant air density in a 2-D flow field.

There are many other display techniques being used by tunnel experimenters, the problem was to devise similar techniques for CINEMA.

One visualization technique that was already being used by the engineers was 2-D contouring of calculated data values. As an engineer worked on the solution of a 2-D problem he would write a small computer program that interfaced his data with available contoruing routines. AFFDL engineers have frequently produced plots of lines of constant air density by using the 2-D contouring package. These plots could be compared with available laser-Schlieren photographs and greatly increased confidence in numerical results. It would have been possible for them to contour not only values of air density but also other aerodynamic variables. The engineers liked the results of the contouring package and wanted it as part of the CINEMA system.

The final requirement was that the Computational Aerodynamics Group needed a computer system that could adapt to many different users (i.e., have flexibility) and they also needed a system that could be easily changed (i.e., modifiable). There were several potential users of the system in the group, and their numerical solutions represented data in a variety of formats. Therefore, CINEMA needed to be able to easily accommodate as many users and their various data formats as possible. In addition to the need for flexibility, it was also important that this system be designed with the future in mind. The needs of research engineers have often changed and expanded, and it appeared very likely that this system would eventually have to change to remain a useful tool. In particular, it seemed important that CINEMA be designed so that it

would be easy to implement new flow visualization techniques. The prospect of even better computers and even better numerical techniques only added support to the idea of the need for an easily modifiable system.

In summary, the AFFDL had established the capability of numerically solving some complex and challenging aerodynamic problems. However, to gain the full benefit available from these solutions they needed a computer graphics system that could effectively display their results. It was the purpose of this thesis to design and build that system. The next chapter describes the system analysis that was done as part of this project.

III SYSTEM ANALYSIS

An important step in the development of any large computer system like CINEMA is the evaluation of currently available hardware and software. Several software packages are available, and some are included as part of the system. It is always a good idea to use existing software if it can perform the needed function and if it has already been used and tested. Software is an important part of a computer system, but in a computer graphics project such as this the available hardware is also very important. The graphics hardware determines the type of output available as well as its quality. Finally, since CINEMA is oriented toward film production, it is necessary to have film processing support readily available.

SOFTWARE

Several software packages were considered as possible pieces of CINEMA. As mentioned before, there was no reason to write new software if there were already programs available that could do the job. However, it was necessary to consider conversion costs when evaluating software for it has frequently been very "expensive" to change software from one machine and/or application to another. The software packages considered are discussed below, and they include a 2-D contouring package, several graphics support packages, and a language processing package.

The 2-D contouring package is a set of routines that draws contour lines through data values. It is possible to specify what data values are to be represented by the contour lines and the package is designed to produce graphics output. It is available on the

ASD CDC machines due to the efforts of Petty and Mach (6), and it uses the DISSPLA graphics package. Engineers in the Computational Aerodynamics Group frequently use this package to display their data. This 2-D contouring package is a good set of routines, but it is only one of the software packages considered.

There were several graphics support packages considered as a possible part of the CINEMA system. Three such graphics packages are also available on the ASD CDC computer system. Table 1 lists these packages and provides some information about their capability.

	DISSPLA	GCS	CALCOMP
Core Size* (graphics routines and application routines)	Large, often greater than 100K	65K+	Less than 65K
Reliability/ Maintainability	Good Commercial Software	Poor U.S. Army Software	Good Commercial Software
Interactive	No	Yes	No
3-D Capability	Yes	Yes	Yes
FR-80 Post-Processor	Yes	No	No
Hidden Line Processing	Yes (limited)	No	No

Table 1. Graphics Support Packages

*Core size can be greatly affected by the size of the application program.

Another graphics package that was available was MOVIE.BYU. MOVIE.BYU was not operational on the CDC machines while this project was done, but it was considered because it was written in ANSI standard

FORTRAN. Thus, with a minimum amount of effort it could be modified to run on any machine with an ANSI standard FORTRAN compiler. This package was particularly interesting because MOVIE.BYU produced raster graphics output. All of the packages listed in the table are vector graphics software packages.

The last software package examined was a language processing package obtained from Lawrence Livermore Laboratories (LLL). This package consisted of two parts. Part 1 accepted as input the description of a language in a modified Backus-Naur Form (BNF) format. This format is commonly used by computer scientists to describe programming languages. Its output was a set of parse tables that were to be used by Part 2 of the package. Part 2 of the package was a table-driven parser skelton designed to interpret the language defined for Part 1. It contained all the routines necessary to access and use the parse tables, but it was just a skelton and several required routines were not included. All of the package from LLL was written in ANSI standard FORTRAN so it could easily be modified to run on most machines. This system has been used to describe and implement both interpreters and compilers.

HARDWARE

Not only were several options available in software systems, but WPAFB also had several pieces of computer graphics hardware available. Both raster graphics and vector graphics equipment existed on WPAFB and both types were available to members of the Computational Aerodynamics Group. The group had been using a TEKTRONIX 4014 scope for much of their graphics work. This device is a vector graphics direct view storage tube that can be tied into

the ASD CDC computers. Another type of graphics equipment that was available was a RAMTEK 9000 series scope. This device is a color, raster graphics scope and is currently tied to a PDP-11 host computer.

Another class of computer graphics equipment available on WPAFB are the FR-80 and COMP-80 film/microfilm recorders built by Information International, Inc. The Foreign Technology Division (FTD) has an FR-80, which produces vector graphics output. The COMP-80 on WPAFB is owned and operated by the 2750th ABWg/DAR office, part of the base printing plant; it is raster graphics equipment. Both an FR-80 and a COMP-80 can produce many different types of output depending on what recording attachments are available. The FR-80 at FTD has the following attachments: (1) an attachment for paper plots, (2) 16mm unsprocketed camera, and (3) 35mm sprocketed camera. The COMP-80 can produce 105mm microfiche and 11" wide negatives for printing. The COMP-80 does not have a camera for 16mm or 35mm film. The FR-80 can produce black and white film and can produce color film if the necessary color filters are installed. The FR-80 at FTD does not have any color filters and therefore, can only produce 16mm and 35mm black and white film. Since the 16mm camera is unsprocketed and the 35mm camera is sprocketed, the 35mm film produced has more accurate film advancement and framing. FTD does not use either camera very much since paper plots are by far their most common product.

Both the COMP-80 and FR-80 are used as offline equipment at WPAFB since neither is connected to the ASD computers. Both have their own minicomputer to control their operation and processing

and both have their own magnetic tape drives. To use either the COMP-80 or FR-80 a magnetic tape with graphics commands is created on a separate computer, and that tape can be taken to the FR-80 or COMP-80 to produce the graphics output.

AUDIO-VISUAL SUPPORT

At WPAFB the Military Airlift Command (MAC) audio-visual detachment specializes in film processing. They have the capability to do very professional work, and they have an OXBURY camera that can: (1) copy 16mm film to 16mm film, (2) copy 35mm film to 16mm film, (3) copy 16mm film to 35mm film, (4) make multiple copies of frames, (5) expose a frame with color filters, and (6) superimpose frames. Because of the versatility of the OXBURY camera it is possible to take 35mm black and white (B&W) film and create 16mm color film, if the B&W film is created as follows. For each 16mm color frame up to three 35mm B&W frames are created. A single B&W frame represents information to be plotted in one of the additive primary colors (red, blue, green). By exposing these B&W frames with the correct filter and superimposing the result, a color frame can be generated. Six colors are possible by using the additive colors individually or in pairs: red, blue, green, yellow, cyan, and magenta.

DECISIONS

After a list of available software and hardware had been compiled and the capabilities of the MAC audio-visual detachment had been discovered, the task that remained was to decide what pieces to use to build the CINEMA system. In making these decisions it was necessary to use the AFFDL requirements as guidelines.

The first group of decisions that had to be made involved:

(1) the type of graphics system to build (raster or vector), (2) the displays that CINEMA should create, and (3) the graphics hardware to be used by CINEMA. These areas are interrelated which meant that a decision in one category directly affected a decision in another area. CAG engineers indicated that the two displays that they most wanted to see were 2-D contour plots and particle plots. Based on this CINEMA was designed as a vector graphics system since (1) 2-D contour plots were already being done by the AFFDL using vector graphics, (2) particle plots could be done using vector graphics, (3) a vector graphics scope (TEKTRONIX 4014) was more accessible to projected users than was a raster graphics scope (the RAMTEK), and (4) the FR-80 (vector graphics) had an existing film camera while the COMP-80 (raster graphics) did not. However, some of the flow visualization techniques currently being used in experimental work would be best duplicated with raster graphics, so the idea of eventually using raster graphics was not completely forgotten. It was decided that the first version of CINEMA would generate vector graphics displays only.

Once the type of graphics had been decided, the next step was to choose the underlying graphics package. Although DISSPLA requires a relatively large amount of core memory, it is a very capable system. It was therefore chosen as the graphics support package because of its capabilities and the fact that it has post-processors for both the TEKTRONIX 4014 scope and the FR-80.

Due to the time-dependent nature of much of the data, it was decided that movies would be the best way to present the data. Since an FR-80 was available that decision was feasible.

IV SYSTEM SOFTWARE DESIGN

The majority of the work done for this thesis was the development of software. CINEMA is a complex software system designed to process large amounts of aerodynamic data and generate movies that graphically display that data. The routines that comprise the system can be grouped into three categories: (1) command language parsing routines, (2) application routines, and (3) graphics routines. The parsing routines were obtained from LLL, and they proved to be a very capable and useful set of software routines. The application routines read and process the user's data and call the graphics routines to display the data. The graphics routines are all part of the DISSPLA graphics package and are available from a subroutine library on the ASD CDC machine.

The CINEMA software is controlled through the use of a command language designed for CINEMA. The commands allow the user to specify what technique to use to display the data (2-D contour plot, 2-D particle plot, or 3-D particle plot), how many frames to create, particle locations, and many other factors controlling the data display. (A complete description of the commands is in Appendix C.) These commands are read and error-checked by the parsing routines. If the commands are error free then one of the application routines is called to do the indicated action. The parsing routines work in an interpretive mode (one at a time) and the application routines do as much processing as possible when called. For example, when a FRAMES command is issued the following steps occur. The parsing routines determine if all required parameters have been included.

Those parameters are then saved in variables in blank common. Once the command has been checked by the parsing routines, the application routine that controls movie frame generation is called. This application routine controls all the data calculations and graphics work associated with frame generation. Finally, when the application routine is done producing the frames, it returns to the parsing routines. The parsing routines then continue the interpretive processing of user commands. Each of the commands is processed in a similar fashion. Now that this background has been provided, each section will be described in detail.

PARSING ROUTINES

The parsing routines provide the interface between the user and the software. They are part of a language processing package from LLL entitled LR. LR comes in two parts: Part One is an automatic parser generator and Part Two is the language processor skelton. Part Two is not complete by itself, but it does form the basis of a particular type of language processor -- an LR(1) parser. The LR(1) classification indicates that the language being parsed (processed) is constructed in a special way. If the parsing routines "remember" what they have read up to a point, then all they need to know to continue parsing from that point, is the next language element in the input. Although this is a fairly restrictive requirement it still allows the language to be powerful and flexible.

Part One reads a language description in a slightly modified BNF format and creates parse tables. There are several parse tables created by Part One of the package, and they collectively contain all the information needed to parse the language. Some of the

tables contain data about what is the next legal input, others contain information about what words are legal in the language (the vocabulary) and more. These tables are designed to be used by Part Two of the package, which is a skeleton set of parsing routines.

The routines of Part Two read the command and break it up into basic elements of the language (tokens). Then they use the tables created by Part One of the package to check the command for errors in format (syntatic errors). If the command is error free, then the semantic routine is called to perform any necessary action. (This is often done by calling an application subroutine.) As was mentioned earlier, Part Two of the LR package from LLL is not complete. It does contain the routines that access the parse tables (all the tables except the vocabulary tables), but does not contain any routines to: (1) read a command and break it up into tokens (lexical routines), (2) search the vocabulary tables, or (3) do the action indicated by a command (semantic routines). There are good reasons these routines are not included -- they are all installation and implementation dependent. Therefore, it is necessary to write these routines locally to complete Part Two of the package.

In order to make the LR system part of the CINEMA software the following steps were accomplished:

1. Modify and test Part One of the LR Package.
2. Design and debug a command language for CINEMA using Part One of the package.
3. Write and debug the routines needed to complete Part Two of the LR package.

There was very little work required to complete step one. All of the LR package was written in ANSI standard FORTRAN 66. It was necessary to slightly modify two routines and, after that was done, Part One worked. A sample language from the program documentation (11) was used to test its operation.

Once Part One worked the next step was to design a command language for CINEMA. The CINEMA language included commands and also some elementary language constructs. An IF-THEN-ELSE, a FOR-LOOP, and expression evaluation were all included in the CINEMA language (see Appendix B for the complete language definition). Using the modified BNF was fairly easy, but writing a context-free, LR(1) language was not easy. Several runs were required before Part One of the LR package accepted the language description and generated the tables.

Finally, after the language was accepted and parse tables built the routines needed to complete the parser were written. These routines perform lexical analysis, do a vocabulary table search, and provide the interface between the parsing routines and the application routines. The lexical analysis routines are several routines which as a group find and return the next token in the input stream. The vocabulary table search routine does a linear search of the vocabulary tables. The routine that is the interface between the parsing routines and the application routines calls the appropriate application routine after a legal command has been parsed. Additional information on the LLL language processing package is included in Appendix A.

APPLICATION ROUTINES

These routines read the user's data, prepare it for plotting, and call grpahics routines to do the actual plotting. There are several routines in this category and they all do separate but related functions. However, the overriding purpose they all share as a group is the graphical display of aerodynamic flow field data.

There are two main data files the user may provide to CINEMA, a flow field data file and a surface data file. A physical record on the flow field data file will contain several arrays. These multi-dimensional arrays contain the values of aerodynamic variables evaluated throughout the finite-difference mesh. For a 2-D contour plot a physical record might contain three 2-D arrays (x-location, y-location and a data array -- such as air density). For a 3-D particle plot a physical record might contain six 3-D arrays (x-location, y-location, z-location, x-component of velocity, y-component of velocity, and z-component of velocity). In an effort to allow the user a great deal of flexibility in organizing the data and at the same time conserve core space (six 3-D arrays can require a large amount of memory) the following plan is used. A FORTRAN BUFFER IN command is used to read the flow field data file. This command reads in a block (physical record) of data each time it is called. The number of words it reads can be variable; however, CINEMA can correctly access the data. BUFFER IN is a direct transfer of data from file to core, and no type conversions take place. For instance, if 2000 words are in each physical record and the first 1000 are integer numbers and the second 1000 are real numbers,

it is up to the program to place the input data into the correct type of variables. If the integer values are placed into real variables or vice-versa then unwanted data transformations can take place. Therefore, the user must tell CINEMA how the data is arranged. Returning to the 2-D contour plot example, the physical record size might be 2700 words with each of the three arrays (x-location, y-location and air density) being dimensioned 30 by 30. Rather than try to set aside enough arrays to handle all cases, a single large array was set aside whithin blank common. CINEMA requires that the user specify physical record size and the size of the data arrays within the physical record. Then, it reads the flow field data record into this large array and uses the information about the size of the data arrays to do its own accessing of the data. This requires a symbol table and some additional work when using data from the flow field file. However, the flexibility in data handling is worth the effort.

The surface file is not nearly as complex as the flow field data file. This file contains the description of the surface or object within the flow field, such as a wing, a corner, a cavity, etc. The data on this file must be in a specific format which is explained in detail in Appendix C. When the user specifies a surface file CINEMA attaches the file (it assumes a permanent file), uses a FORMATTED READ to get the data, and then prepares the data for plotting. The surface description is stored in blank common in a format that DISSPLA can readily use, and each time the surface is to be plotted the data in blank common is used.

After the data has been read in from the user's files, CINEMA must prepare it for plotting. If 2-D contour plots are requested then a set of routines locally developed by Kervyn Mach and James Petty (6) are used. With a little effort these routines were integrated into the CINEMA system. If particle plots are requested, then several routines developed specifically for CINEMA are used. Particle plots can be done in either 2-D or 3-D but the concepts are basically the same for either case. Several data arrays are associated with particle plot generation, and these arrays are created within the large blank common array. One of the most important routines involved in particle plots is the one that determines the new locations of the particles. CINEMA must "move" the particles between frames, and this requires that velocity data from the flow field file be used along with the current particle location. It is also possible to trail certain particles and/or connect particles with straight lines, so once the particle locations have been updated other routines search the particle arrays and create the requested trails and/or connections.

To be more specific, to move a particle CINEMA keeps tables that contain particle locations in cartesian coordinates and tables that keep mesh location indices. CINEMA starts at the top of these tables and updates each particle location by computing how far a particle would move in an increment of time with a velocity defined by the flow field data for the current location. Since the current location is known it is only necessary to determine how far it moves. The mesh location indices identify the closest mesh point, and these

indices are used to extract from the user supplied velocity arrays the value of velocity in all directions. Now, both the old location and the velocity components at the closest mesh point are known. To determine a new location simply take the user supplied time increment and use elementary linear motion equations from Physics and compute the new location. At this point if the new particle location exceeds the bounds of the problem, then the particle is removed and no longer displayed. If the particle is still active, then arrays used to draw particle trails are updated. This is necessary because to display a particle trail it is necessary to store the previous locations for each trailed particle. In this way all the particle data stored in arrays is updated between frame generation.

Once the data has been read and processed it is ready to be plotted. Some of the application routines deal solely with the tasks of calling graphics routines; these routines pass all the necessary data to the graphics routines. All the graphics work is done using the DISSPLA system. This is a commercially developed graphics package that has 2-D and 3-D capability. It has some hidden line processing in 3-D (i.e., any parts of a single surface hidden from view will not be displayed, but if two or more surfaces are involved there is no processing to account for one surface hiding parts of another).

When DISSPLA routines are used they write their output to an intermediate file. This intermediate file is designed to contain device-independent graphics information. To use this data on a particular piece of graphics hardware it is necessary to use a

post-processor. A post-processor is a stand-alone program that reads the intermediate file and formats the data for use with a particular piece of equipment. For CINEMA users, the TEKTRONIX post-processor is used to examine or preview the CINEMA output. If the results are satisfactory then the FR-80 post-processor is used to generate film.

In summary, CINEMA is a large, complex computer system that is designed to graphically display complex aerodynamic data. Through the use of a command language a user can direct CINEMA in the processing and display of the data. It has a great deal of flexibility and should prove to be a very capable tool. By using this system aerodynamic engineers should be able to gain a much better understanding of their data, and the use of CINEMA will make it much easier to compare theoretical results with experimental results. CINEMA provides theoreticians the ability to visualize their results, something experimentalists have relied upon for many years. Again, it should be noted that CINEMA is not limited to the processing of aerodynamic data. It is designed to accept large data bases, and with little or no modification it can effectively generate the graphical displays required by many computer based research and development programs.

V RESULTS AND RECOMMENDATIONS

The major result of this thesis project is a computer graphics system entitled CINEMA. CINEMA gives aerodynamic engineers the new and unique ability to visualize numerically calculated flow fields. The real advantage of being able to visualize numerically derived flow fields is that a comparison of these theoretical results with experimental results is considerably easier. Any user will find CINEMA a capable and easy to use system that accurately and effectively displays their results. Since CINEMA is designed to produce movies it is impossible to adequately present its full display capabilities. In an effort to provide a sample of CINEMA's output a few frames are included in Appendix C, Section V.

This project not only produced a valuable computer system, but it also provided a good educational experience for the author. In fact, there were three areas in which this project proved to be both a challenge and an enriching experience: (1) the application of automatic parsing techniques, (2) the design and implementation of a command language, and (3) the creation of computer generated movies.

The use of the LR automatic parsing package from LLL was very enjoyable. Although the modification of software can be tedious, LR was well written and it was not too difficult to produce a command interpreter using this parsing package. LR is a very capable and versatile package, and it could be used in many different applications -- anywhere a language interpreter or compiler is needed. Now that LR has been used to build an interpreter on the ASD CDC machines the author hopes that it will be used for other projects.

Associated with the use of the LR package was the design of a command language for CINEMA. As was mentioned earlier, it was quite a task to design, format, and debug an LR(1) type language. Before the language could be completely designed it was necessary to do a majority of the preliminary design work for the entire system. Performing the tasks of designing some commands, evaluating them, and then making needed changes was both important and educational.

There is one last area that should be mentioned, the creation of computer generated movies. Because of this project the author gained experience in a new computer graphics technique and gained experience in the use of a popular commercial graphics package, DISSPLA.

When a project of this size is done it touches upon many different areas, and there are several recommendations for future work that are a direct result of doing this project. The recommendations are listed below.

1. Develop and add new flow visualization techniques. Currently CINEMA only provides contour plots and particle plots. AFFDL engineers would undoubtedly use and benefit from other visualization techniques. Adding the new commands should be fairly easy, and it should even be possible to use the MOVIE.BYU raster graphics package to provide CINEMA users the choice of raster graphics output or vector graphics output. There are many possibilities in this area.

2. Improve the 3-D surface processing. The type of 3-D surface

currently available through the use of DISSPLA is restrictive, and AFFDL engineers could make good use of an improved 3-D surface capability, including complete hidden-line processing. Reference 6 describes two hidden-line packages and both are implemented on the ASD CDC machines.

3. Improve the algorithm used for moving particles. CINEMA's algorithm used to move particles between frames is simple, and it would be possible to refine it by using interpolation and other mathematical techniques.

4. Implement the additional language constructs. Within the BNF description of the command language there are definitions of a FOR loop, an IF-THEN-ELSE statement, and expressions; but these were not implemented in the first version of CINEMA's semantic routine. Their implementation would provide CINEMA a much more powerful language. For example, it would be much easier to use

```
FOR I = 1 to 5
    TRAIL, I;
```

than

```
    TRAIL, 1;
    TRAIL, 2;
    TRAIL, 3;
    TRAIL, 4;
    TRAIL, 5;
```

5. Add color filters to the COMP-80. Although the MAC audio-visual technicians do an excellent job with the OXBURY camera, it would be much better and faster if the COMP-80 had color filters and produced color film directly.

6. Improve and modify the DISSPLA post-processors. Both TEK4010 (the TEKTRONIX post-processor) and FR80 (the FR-80 post-processor) failed to correctly process intermediate graphics files that had more than 499 frames. Also, FR80 can be modified to be used with the COMP-80. This modification should be made and tested by June 30, 1980 which is the expiration date of the agreement between FTD and the AFFDL concerning the 35mm camera on the FR-80.

It is the author's opinion that this was a good thesis topic because it was rewarding, challenging, and educational. Several different topics were included as part of this thesis, and there were several projects suggested by this work.

BIBLIOGRAPHY

1. ASD Computer Center. Battelle Disk File Manipulation Routines User's Guide. Wright-Patterson AFB, Ohio: Air Force Aeronautical Systems Division, July 1978.

2. ASD Computer Center. Preliminary Instructions, Use of the DISSPLA Plotting Library. Wright-Patterson AFB, Ohio: Air Force Aeronautical Systems Division, April 1976.

3. ISSCO. DISSPLA Advanced Manual. San Diego: Integrated Software Systems Corporation, 1970.

4. ISSCO. DISSPLA Beginners/Intermediate Manual. San Diego: Integrated Software Systems Corporation, 1970.

5. ISSCO. DISSPLA Pre-Processor/Post-Processor User Manual. San Diego: Integrated Software Systems Corporation, 1975.

6. Mach, Kervyn D. and James S. Petty. Contouring and Hidden-Line Algorithms for Vector Graphic Displays. AFAPL-TR-77-3. Wright-Patterson AFB, Ohio: Air Force Aerodynamic Propulsion Laboratory, January 1977.

7. McKeeman, William M., et al. A Compiler Generator. Englewood Cliffs: Prentice-Hall, 1970.

8. Merzkirch, Wolfgang F. Flow Visualization. New York: Academic Press, 1974.

9. Scott, John T. "Computer Films for Research," Physics Today, 32:46-52 (January 1979).

10. Vickers, Donald L. "The Role of Computer-Generated Movies in Scientific Research." Digest of Papers Spring COMPCON 79, Eighteenth IEEE Computer Society International Conference. 250-256. New York: Institute of Electrical and Electronics Engineers, Inc., 1979.

11. Wetherell, Charles and Alfred Shannon. LR Automatic Parser Generator and LR(1) Parser. Livermore California: Lawrence Livermore Laboratory, June 1979.

APPENDIX A

LAWRENCE LIVERMORE LABORATORIES (LLL) LANGUAGE PROCESSOR

> "LR is a pair of programs - an automatic parser generator and an LR(1) parser. The parser generator reads a context-free grammar in a modified BNF format and produces tables which describe an LR(1) parsing automaton. The parser is a suite of subroutines which interpret the tables to construct a parse of an input stream supplied by a (locally written) lexical analyzer. The entire system may be used to generate parsers for compilers, utility routines, command interpreters, and the like. LR and its predecessors have been in use at Lawrence Livermore Laboratory (LLL) for ten years. LR's outstanding characteristic is the ease with which new tables can be generated to reflect a change in the language to be parsed. This flexibility is prized by programmers writing utilities and command interpreters whose input languages typically grow and change during program development. LR is written entirely in ANSI standard FORTRAN 66 and requires only minor changes when moved to a new computer." (11:1)

LR was used as a very important part of the CINEMA software. Major Michael Wirth, was instrumental in obtaining this language processing package from LLL, and he was convinced very early in the project of the value of a command language as a part of the CINEMA system. Although LR was written in ANSI standard FORTRAN 66 some work was required to get it operating on the ASD CDC computers. However, as its capabilities were measured against the work associated with getting it to run, it was decided that LR should be used for the command language interpreter section of CINEMA. As mentioned earlier, LR is a pair of programs; each program is described in more detail below. Finally, observations on the use of the LR system are presented.

AUTOMATIC PARSER GENERATOR

This is a relatively large program; its input is a context-free, LR(1) language and its major output is a set of parser tables. The input language is written in a modified Backus-Naur Form (BNF). Some minor restrictions are placed on the input grammer: (1) all productions for a non-terminal must be grouped together, (2) terminals are exactly those symbols which do not appear on the left hand side of a production, and (3) the start symbol is deduced from the context. Input is free format and the modified BNF is as easy to use as standard BNF (11:2).

To get this program to work required very little effort. Two routines were installation dependent (INIT and CHRIND [11:8]). Subroutine INIT is concerned with file input and output and CHRIND needed the local FORTRAN versions of the binary shift and logical and. Once these simple changes were made the automatic parser generator compiled error-free on the local machines. A simple test case taken from the LR program documentation was used to verify that the program worked. The program required about 172,000 octal words of core to execute and the tables generated for the CINEMA command language required just over 10,000 octal words of core. Execution time is directly related to the size and complexity of the input language. The CINEMA command language was processed in about ten seconds of central processor time.

After the automatic parser generator was available on the CDC machines, the next step was to design the CINEMA command language and input its description into the automatic parser generator. Careful and thorough checking of the input language is done by

the program. Several of the error messages and sections of the output from the program are based on the principles and concepts of automatic parser generation. Several runs with errors were made before an acceptable language for CINEMA was developed, and Appendix B contains a complete description of the CINEMA command language.

The most important output from this program is a set of parse tables. This data is available in three formats: (1) ANSI FORTRAN 66 DIMENSION and DATA statements, (2) DATA and PARAMETER statements in LRLTRAN, a language used at LLL, and (3) a pure data file format (11:2). When correct tables have been created the next step is to include these parse tables in the PARSER skeleton (the second program in the set). Since these tables are such an important part of this package each one is described below. These descriptions are summaries of comments taken from an LRLTRAN parser program, but the reader is warned that the entire LR package is based upon the theory of automatic parser generation, a subset of formal language theory. It is not necessary to understand these tables to use the LR package, and these table descriptions are only included for those interested readers familiar with these concepts.

VOCABULARY - two arrays are used to describe the character strings forming the vocabulary. V contains the character strings, one right after another, and VOC is the set of pointers into the V array. Each VOC entry points to the start of a vocabulary string in V.

TRAN - an array of target states for the read transitions.

FTRN - Since the input language is LR(1) it is possible to transition from one state to another in the parsing automaton by knowing only the next symbol and the current state. FTRN is an array of pointers into the transition table. Assume state S is currently in control, then TRAN(FTRN(S)) through TRAN(FTRN(S + 1)) are potential targets. The read symbol that will transition the automaton to the target

state T is ENT(T). Within the target range for control state S the legal next symbols are sorted in token order, and this allows a binary search over the possible transitions for a state.

ENT - ENT(S) is the symbol which must be read to enter state S (see above).

NSET - a vector of pointers into LOOKSET (LS). There are as many entries as there are reductions in the parser. The set LS (NSET(I)) is one-half of the pair of look-ahead set reduction.

PROD - a vector of production numers used as one-half of the look-ahead set reduction pairs.

LHS - left-hand side of a production. Entry in position P is the token for the non-terminal on the left of production P. This token is pushed back into the input stream when a reduction by P is performed.

LEN - Entry P is the length of the right side of production P and is used to pop the stack when reducing by production P.

LOOKSET - arrays LSET and LS comprise the look-ahead sets. LSET is an array of pointers into LS. LS contains a list of terminals in the lookahead set.

LR(1) PARSER

LLL provided an ANSI FORTRAN 66 set of routines that accessed the parse tables created by the automatic parser generator. These routines performed specific functions. For example: (1) find a transition, (2) do a transition, (3) find a reduction, and (4) do a reduction. All these skeleton routines compiled with no errors after the DATA and DIMENSION statements from Part 1 were included. However, the provided routines were only a skeleton for a complete parser. The routines that had to be added were lexical analysis routines, a semantic routine, and a vocabulary table search routine. The great majority of the parsing work is done using parallel stacks and the parser is a modified finite automaton. The parser changes from state to state, and it performs either a reduction or

a read transition. Since this is an LR(1) parser the parser "knows" where it has been because it maintains a state stack and it knows the next token in the input stream.

The lexical analysis routines were some of the first that had to be added to the parser skeleton. The major purpose of the lexical routines was to return to the calling routine the next token in the input stream. SCANNER was the main lexical analyzer routine. It has supporting routines that initialized the system and read the next line (GETLIN), placed the next token in a common block (LEXPAS, the lexical pass routine). All these routines that do the lexical scanning were CDC dependent to some degree, but two routines used by SCANNER were particularly so - DIGIT and LETTER. These routines determine if a single hollerith character was a digit or letter respectively. Much of the lexical analysis is machine dependent, and it was probably excluded from the LLL skeleton for that reason. SCANNER had to be designed to return the terminals and non-terminals in the language. (Fortunately, Part 1 of LR provides a list of the terminals and non-terminals of the language.)

The next part that was added was the vocabulary table search routine, IFNDTK (I find Token). Its function is to search the vocabulary for the token to be returned by SCANNER. SCANNER returns a character string or a numeric value and uses IFNDTK to determine if a string is an identifier or a reserved word in the language. IFNDTK must search the vocabulary table for the string. Although comments in part of the code indicated that the vocabulary table was lexically sorted, it was in fact <u>not</u> sorted. If it were sorted, a binary search would have been possible. Consequently, in IFNDTK,

a linear search is done for the input string.

The final routine added to the parser was the semantic processor, SMANTK. This routine provided the interface between the parser and the application routines. Upon entry into SMANTK the production number is known (each BNF production is given a number by the automatic parser generator). Consequently, SMANTK is a very large computed GO TO which switches on production number. For instance, when SMANTK is called by DORED with a production number of P, then the action that is necessary for production P is performed. If P corresponds to

P. <AXIS CMD> ::= AXIS, YES

then a variable in a COMMON block is set to indicate that an axis is to be displayed. Often a production will be of the form:

<BLOCK SIZE> ::= <INTEGER CON>

and at this time SMANTK can obtain the numeric value of the integer constant from the value stack (VALSTK). When more complicated productions are encountered, subroutines are called.

The LR system from LLL was a powerful addition to the CINEMA system. To use it in the system required a language definition and the generation of several special purpose routines to complete the parser skeleton. The result was well worth the effort. CINEMA has a command language that can be easily changed, and the parsing of that language is quick, accurate, and reliable. The only disadvantage to using the LR package is that the parse tables do require a large amount of core storage. The LR package is a good one that could easily be used for a variety of applications.

APPENDIX B

CINEMA COMMAND LANGUAGE

A user communicates with CINEMA through the use of a command language that was designed specifically for CINEMA. The LR package from Lawrence Livermore Laboratories is described in Appendix A and was used as a command language interpreter for the CINEMA system. The language to be processed by LR is described in a modified Backus-Naur Form (BNF) format and must be a context-free, LR(1) language. There is a considerable amount of effort involved in designing and debugging a language. Throughout this process it can be difficult to maintain the LR(1) characteristic of the language as ambiguities are often a problem. McKeeman in reference 7 calls the preparation of a grammar (language definition) BNF debugging, and he also points out that debugging grammars is a skill acquired through practice. After several attempts the CINEMA command language was successfully defined. The following two listings are (1) the language description in modified BNF in the format required by LR, and (2) the numbered productions. To help understand the language description the following facts are provided:

1. &C - denotes the beginning of a comment. All characters on a card after &C are ignored.
2. &J, &K - are program switches (or toggles) that control what type of parse tables are generated by LR.
3. &A - end of alternate, the end of one alternate definition of a non-terminal.
4. &P - end of production, signals the end of a group of alternate definitions.
5. &G - end of grammar.

The input format is free-form and items are blank delimited. The best way to learn to write BNF for LR is to look at examples, try a simple example, and then try a new language definition. Additional information on the input formats can be found in reference 11. Part one of the LR package is an excellently documented program and contains information about input formats and toggle settings. Once the language had been successfully defined the parse tables are generated. Part of the output from the job that generates the tables is a numbered list of the productions in the language. The CINEMA commands are in the first part of the productions and several language constructs (such as an IF-THEN-ELSE and a FOR loop) are in the last part of the productions. The following lists provide the CINEMA command language description, a complete explanation of the individual commands is in Appendix C, the user's guide.

BNF DESCRIPTION OF CINEMA LANGUAGE

```
%I %C ECHO THE INPUT
%C
%C COMMAND LANGUAGE FOR AFFDL'S CINEMA
%C
%J %K %C GENERATE FORTRAN 66 TABLES
%C
<PROGRAM> <VARIABLE DECLARATION LIST> <BLOCK>  %A
<STATEMENT LIST> <BLOCK>  %P
%C
<BLOCK> <VARIABLE DECLARATION LIST> <BLOCK> %A
<STATEMENT LIST> <BLOCK> %A
<STATEMENT LIST>  %P
%C
%C
<STATEMENT LIST> <STATEMENT> %A
<STATEMENT> END. %A
<STATEMENT LIST> : <STATEMENT> END. %P
%C
<STATEMENT> <SIMPLE STATEMENT> %A
<STRUCTURED STATEMENT> %A
<COMMAND> %P
%C
%C
%C *****COMMANDS*****
%C
<COMMAND> <FILE COMMANDS> %A
<MODE COMMANDS> %A
<PARAMETER COMMANDS> %P
```

BNF DESCRIPTION OF CINEMA LANGUAGE

```
%C
%C
%C FILE COMMANDS
%C
<FILE COMMANDS> <FLOW FIELD FILE COMMAND> &A
<SURFACE FILE COMMAND> &A
<COMMAND FILE COMMAND> &A
<MATRIX COMMAND> &P
%C
<FLOW FIELD FILE COMMAND> FLOW , <FILE NAME> , <FILE TYPE> , <BLOCK SIZE> &P
%C
<FILE NAME> <IDENTIFIER> &P
%C
<FILE TYPE> DISK &A
TAPE &P
%C
<BLOCK SIZE> <INTEGER CON> &P
%C
<MATRIX COMMAND> MATRIX , <MATRIX NAME> , <MAX INDICES> , <MAX VAL LIST> &P
%C
<MATRIX NAME> <IDENTIFIER> &P
%C
<MAX INDICES> <INTEGER CON> &P
%C
<MAX VAL LIST> <INTEGER LIST>   &P
%C
<INTEGER LIST> <INTEGER CON>   &A
<INTEGER LIST> , <INTEGER CON>   &P
```

BNF DESCRIPTION OF CINEMA LANGUAGE

```
%C
<SURFACE FILE COMMAND> SURFACE , <FILE NAME> , <COLOR> %P
%C
<COLOR> R %A
B %A
G %A
Y %A
C %A
M %P
%C
<COMMAND FILE COMMAND> COMMAND , <FILE NAME> %P
%C
%C MODE COMMANDS
%C
<MODE COMMANDS> <MESH CMD> %A
<DATA CMD> %A
<AXIS CMD> %A
<VIEW CMD> %A
<SCALE CMD> %A
<EYEPT CMD> %A
<SPACE CMD> %A
<MAX TRAILS CMD> %A
<MAX CONNECTS CMD> %A
<RANGE CMD> %P
%C
<MESH CMD> MESH , VARIABLE %A
MESH , CONSTANT %P
%C
```

BNF DESCRIPTION OF CINEMA LANGUAGE

```
<DATA CMD> DATA , DYNAMIC &A
DATA , STATIC &P
%C
<AXIS CMD> AXIS , YES &A
AXIS , NO &P
%C
<VIEW CMD> VIEW , SURFACE &A
VIEW , FLOW &A
VIEW , BOTH &P
%C
<SCALE CMD> SCALE , <REAL CON> &P
%C
<EYEPT CMD> EYEPT , <REAL LIST>  &P
%C
<REAL LIST> <REAL CON>  &A
<REAL LIST> , <REAL CON>   &P
%C
<SPACE CMD> SPACE , <INTEGER CON> &P
%C
<MAX TRAILS CMD> MAXTRAILS , <INTEGER LIST> &P
%C
<MAX CONNECTS CMD> MAXCONNECT , <INTEGER CON> &P
%C
<RANGE CMD> RANGE , <REAL LIST>  &P
%C
%C PARAMETER COMMANDS
%C
<PARAMETER COMMANDS> <COORD CMD> &A
```

BNF DESCRIPTION OF CINEMA LANGUAGE

<CONTOUR CMD> &A

<PARTICLE CMD> &A

<MARKER CMD> &A

<SAVE CMD> &A

<FRAMES CMD> &A

<REWIND CMD> &A

<CONNECT CMD> &A

<TRAIL CMD> &P

&C

<COORD CMD> COORD , <REAL LIST> &P

&C

<CONTOUR CMD> CONTOUR , <COLOR> , <MATRIX NAME LIST> , <REAL LIST> &P

&C

<PARTICLE CMD> PARTICLE , <NO. PARTICLES> , <INC> , <MATRIX NAME LIST> &P

&C

<NO. PARTICLES> <INTEGER CON> &P

&C

<INC> <REAL CON> &P

&C

<MATRIX NAME LIST> <MATRIX NAME> &A

<MATRIX NAME LIST> , <MATRIX NAME> &P

&C

<MARKER CMD> MARKER , <PARTICLE NUMBER> , <COLOR> , <LOCATION> &P

&C

<PARTICLE NUMBER> <INTEGER CON> &P

&C

<LOCATION> <INTEGER LIST> &P

&C

BNF DESCRIPTION OF CINEMA LANGUAGE

```
<SAVE CMD> SAVE , <FILE NAME> $A

SAVE , <FILE NAME> ,PURGE $P

$C

<FRAMES CMD> FRAMES , <FRAME COUNT> $P

$C

<FRAME COUNT> <INTEGER LIST>  $P

$C

<REWIND CMD> REWIND $P

$C

<CONNECT CMD> CONNECT , <PARTICLE NUMBER> , <PARTICLE NUMBER> $P

$C

<TRAIL CMD> TRAIL , <PARTICLE NUMBER> $P

$C

$C

$C VARIABLE DECLARATION PART

$C

<VARIABLE DECLARATION LIST> VAR <IDENTIFIER LIST> : <TYPE> $P

$C

<IDENTIFIER LIST> <IDENTIFIER> $A

<IDENTIFIER> , <IDENTIFIER LIST> $P

$C

<TYPE>INTEGER $A

REAL $A

<ARRAY TYPE> $P

$C

<ARRAY TYPE> ARRAY [ <SUBRANGE LIST> ] OF REAL $A

ARRAY [ <SUBRANGE LIST> ] OF INTEGER $P

$C
```

BNF DESCRIPTION OF CINEMA LANGUAGE

```
<SUBRANGE LIST> <SUBRANGE LIST> , <SUBRANGE TYPE> $A
<SUBRANGE TYPE> $P
$C
<SUBRANGE TYPE> <INTEGER CON> .. <INTEGER CON> $P
$C
$C SIMPLE STATEMENT
$C
<SIMPLE STATEMENT> <VARIABLE> := <EXP> $P
$C
<VARIABLE> <IDENTIFIER> $A
<INDEXED VARIABLE> $P
$C
<INDEXED VARIABLE> <IDENTIFIER> ( <INDEX LIST> ) $P
$C
<INDEX LIST> <INDEX LIST> , <INDEX> $A
<INDEX> $P
$C
<INDEX> <EXP> $P
$C
<EXP> <EXPA>     $A
<EXP> .EQ. <EXPA>     $A
<EXP> .NE. <EXPA>     $A
<EXP> .LT. <EXPA>     $A
<EXP> .GT. <EXPA>     $A
<EXP> .GE. <EXPA>     $A
<EXP> .LE. <EXPA>     $P
$C
<EXPA> <EXPB>     $A
```

BNF DESCRIPTION OF CINEMA LANGUAGE

<EXPA> + <EXPB> $A

<EXPA> - <EXPB> $A

+ <EXPB> $A

- <EXPB> $P

$C

<EXPB> <EXPC> $A

<EXPB> * <EXPC> $A

<EXPB> / <EXPC> $P

$C

<EXPC> <INTEGER CON> $A

<REAL CON> $P

$C

$C STRUCTURED STATEMENT

$C

<STRUCTURED STATEMENT> <COMPOUND STATEMENT> $A

<IF STATEMENT> $A

<FOR STATEMENT> $P

$C

<COMPOUND STATEMENT> BEGIN <STATEMENT LIST> END $P

$C

<IF STATEMENT> <IF> <ELSEIF> <ELSE> ENDIF $P

$C

<IF> <IFTEST> <STATEMENT> $P

$C

<IFTEST> IF <EXP> THEN ! $P

$C

<ELSEIF> <ELSEIF> <EFTEST> <STATEMENT> $A

$P

BNF DESCRIPTION OF CINEMA LANGUAGE

```
 %C
 <EFTEST> ELSEIF <EXP> THEN :    %P
 %C
 <ELSE> <ELTEST> <STATEMENT>     %A
                                 %P
 %C
 <ELTEST> ELSE :      %P
 %C
 %C
 <FOR STATEMENT> FOR <CONTROL VARIABLE> := <FOR LIST> DO <STATEMENT> %P
 %C
 <CONTROL VARIABLE> <IDENTIFIER> %P
 %C
 <FOR LIST> <INITIAL VALUE> TO <FINAL VALUE> %P
 %C
 <INITIAL VALUE> <INTEGER CON> %P
 %C
 <FINAL VALUE> <INTEGER CON> %P
 %G %C END OF THE GRAMMAR
****    XPAXXL7   //// END  OF LIST ////
```

NUMBERED LANGUAGE PRODUCTIONS

```
    1 <SYSTEM GOAL SYMBOL> ::= END <PROGRAM> END

    2 <PROGRAM> ::= <VARIABLE DECLARATION LIST> <BLOCK>
    3             / <STATEMENT LIST> <BLOCK>

    4 <BLOCK> ::= <VARIABLE DECLARATION LIST> <BLOCK>
    5           / <STATEMENT LIST> <BLOCK>
    6           / <STATEMENT LIST>

    7 <STATEMENT LIST> ::= <STATEMENT>
    8                    / <STATEMENT> END.
    9                    / <STATEMENT LIST> ! <STATEMENT> END.

   10 <STATEMENT>
= <SIMPLE STATEMENT>
   11               / <STRUCTURED STATEMENT>
   12               / <COMMAND>

   13 <COMMAND> ::= <FILE COMMANDS>
   14             / <MODE COMMANDS>
   15             / <PARAMETER COMMANDS>

   16 <FILE COMMANDS> ::= <FLOW FIELD FILE COMMAND>
   17                   / <SURFACE FILE COMMAND>
   18                   / <COMMAND FILE COMMAND>
   19                   / <MATRIX COMMAND>

   20 <FLOW FIELD FILE COMMAND> ::= FLOW , <FILE NAME> , <FILE TYPE> , <BLOCK SIZE>
```

NUMBERED LANGUAGE PRODUCTIONS

21 <FILE NAME>
= <IDENTIFIER>

22 <FILE TYPE>
= DISK
23 / TAPE

24 <BLOCK SIZE> ::= <INTEGER CON>

25 <MATRIX COMMAND> ::= MATRIX , <MATRIX NAME> , <MAX INDICES> , <MAX VAL LIST>

26 <MATRIX NAME> ::= <IDENTIFIER>

27 <MAX INDICES> ::= <INTEGER CON>

28 <MAX VAL LIST> ::= <INTEGER LIST>

29 <INTEGER LIST> ::= <INTEGER CON>
30 / <INTEGER LIST> , <INTEGER CON>

31 <SURFACE FILE COMMAND> ::= SURFACE , <FILE NAME> , <COLOR>

32 <COLOR> ::= R
33 / B
34 / G
35 / Y
36 / C

NUMBERED LANGUAGE PRODUCTIONS

```
37          / M

38 <COMMAND FILE COMMAND> ::= COMMAND , <FILE NAME>

39 <MODE COMMANDS> ::= <MESH CMD>
40                   / <DATA CMD>
41                   / <AXIS CMD>
42                   / <VIEW CMD>
43                   / <SCALE CMD>
44                   / <EYEPT CMD>
45                   / <SPACE CMD>
46                   / <MAX TRAILS CMD>
47                   / <MAX CONNECTS CMD>
48                   / <RANGE CMD>

49 <MESH CMD> ::= MESH , VARIABLE
50              / MESH , CONSTANT

51 <DATA CMD> ::= DATA , DYNAMIC
52              / DATA , STATIC

53 <AXIS CMD> ::= AXIS , YES
54              / AXIS , NO

55 <VIEW CMD> ::= VIEW , SURFACE
56              / VIEW , FLOW
57              / VIEW , BOTH
```

NUMBERED LANGUAGE PRODUCTIONS

58 <SCALE CMD>
= SCALE , <REAL CON>

59 <EYEPT CMD>
= EYEPT , <REAL LIST>

60 <REAL LIST>
= <REAL CON>
61 / <REAL LIST> , <REAL CON>

62 <SPACE CMD>
= SPACE , <INTEGER CON>

63 <MAX TRAILS CMD> ::= MAXTRAILS , <INTEGER LIST>

64 <MAX CONNECTS CMD> ::= MAXCONNECT , <INTEGER CON>

65 <RANGE CMD>
= RANGE , <REAL LIST>

66 <PARAMETER COMMANDS> ::= <COORD CMD>
67 / <CONTOUR CMD>
68 / <PARTICLE CMD>
69 / <MARKER CMD>
70 / <SAVE CMD>
71 / <FRAMES CMD>
72 / <REWIND CMD>
73 / <CONNECT CMD>

NUMBERED LANGUAGE PRODUCTIONS

74 / <TRAIL CMD>

75 <COORD CMD>
= COORD , <REAL LIST>

76 <CONTOUR CMD> ::= CONTOUR , <COLOR> , <MATRIX NAME LIST> , <REAL LIST>

77 <PARTICLE CMD> ::= PARTICLE , <NO. PARTICLES> , <INC> , <MATRIX NAME LIST>

78 <NO. PARTICLES> ::= <INTEGER CON>

79 <INC> ::= <REAL CON>

80 <MATRIX NAME LIST> ::= <MATRIX NAME>

81 / <MATRIX NAME LIST> , <MATRIX NAME>

82 <MARKER CMD> ::= MARKER , <PARTICLE NUMBER> , <COLOR> , <LOCATION>

83 <PARTICLE NUMBER> ::= <INTEGER CON>

84 <LOCATION> ::= <INTEGER LIST>

85 <SAVE CMD> ::= SAVE , <FILE NAME>

86 / SAVE , <FILE NAME> ,PURGE

87 <FRAMES CMD> ::= FRAMES , <FRAME COUNT>

88 <FRAME COUNT> ::= <INTEGER LIST>

NUMBERED LANGUAGE PRODUCTIONS

```
   89 <REWIND CMD> ::= REWIND

   90 <CONNECT CMD> ::= CONNECT , <PARTICLE NUMBER> , <PARTICLE NUMBER>

   91 <TRAIL CMD>
= TRAIL , <PARTICLE NUMBER>

   92 <VARIABLE DECLARATION LIST> ::= VAR <IDENTIFIER LIST> : <TYPE>

   93 <IDENTIFIER LIST> ::= <IDENTIFIER>
   94                     / <IDENTIFIER> , <IDENTIFIER LIST>

   95 <TYPE> ::= INTEGER
   96          / REAL
   97          / <ARRAY TYPE>

   98 <ARRAY TYPE> ::= ARRAY [ <SUBRANGE LIST> ] OF REAL
   99                / ARRAY [ <SUBRANGE LIST> ] OF INTEGER

  100 <SUBRANGE LIST> ::= <SUBRANGE LIST> , <SUBRANGE TYPE>
  101                   / <SUBRANGE TYPE>

  102 <SUBRANGE TYPE> ::= <INTEGER CON> .. <INTEGER CON>

  103 <SIMPLE STATEMENT> ::= <VARIABLE> := <EXP>

  104 <VARIABLE> ::= <IDENTIFIER>
```

NUMBERED LANGUAGE PRODUCTIONS

```
105              / <INDEXED VARIABLE>

106 <INDEXED VARIABLE> ::= <IDENTIFIER> ( <INDEX LIST> )

107 <INDEX LIST> ::= <INDEX LIST> , <INDEX>
108                 / <INDEX>

109 <INDEX> ::= <EXP>

110 <EXP> ::= <EXPA>
111         / <EXP> .EQ. <EXPA>
112         / <EXP> .NE. <EXPA>
113         / <EXP> .LT. <EXPA>
114         / <EXP> .GT. <EXPA>
115         / <EXP> .GE. <EXPA>
116         / <EXP> .LE. <EXPA>

117 <EXPA> ::= <EXPB>
118          / <EXPA> + <EXPB>
119          / <EXPA> - <EXPB>
120          / + <EXPB>
121          / - <EXPB>

122 <EXPB> ::= <EXPC>
123          / <EXPB> * <EXPC>
124          / <EXPB> / <EXPC>

125 <EXPC> ::= <INTEGER CON>
```

NUMBERED LANGUAGE PRODUCTIONS

```
126          / <REAL CON>

127 <STRUCTURED STATEMENT> ::= <COMPOUND STATEMENT>
128                          / <IF STATEMENT>
129                          / <FOR STATEMENT>

130 <COMPOUND STATEMENT> ::= BEGIN <STATEMENT LIST> END

131 <IF STATEMENT> ::= <IF> <ELSEIF> <ELSE> ENDIF

132 <IF> ::= <IFTEST> <STATEMENT>

133 <IFTEST> ::= IF <EXP> THEN ;

134 <ELSEIF> ::= <ELSEIF> <EFTEST> <STATEMENT>
135            /

136 <EFTEST> ::= ELSEIF <EXP> THEN ;

137 <ELSE> ::= <ELTEST> <STATEMENT>
138          /

139 <ELTEST> ::= ELSE ;

140 <FOR STATEMENT> ::= FOR <CONTROL VARIABLE> := <FOR LIST> DO <STATEMENT>

141 <CONTROL VARIABLE> ::= <IDENTIFIER>
```

NUMBERED LANGUAGE PRODUCTIONS

142 <FOR LIST> ::= <INITIAL VALUE> TO <FINAL VALUE>

143 <INITIAL VALUE> ::= <INTEGER CON>

144 <FINAL VALUE> ::= <INTEGER CON>

****** XPAXXL7 //// END OF LIST ////

APPENDIX C

CINEMA USER'S GUIDE

CINEMA is a computer system that uses computer graphics to visualize aerodynamic flow fields. It is designed to produce 16mm film that displays numerically calculated flow field data. The CINEMA software executes on the ASD CDC machines and uses the DISSPLA graphics package.

A user controls CINEMA with a command language. There are three categories of commands in CINEMA's language: (1) file commands, (2) mode commands, and (3) parameter commands. The file commands describe the data files to be processed by CINEMA. The mode commands provide general information about the data (for example 2-D or 3-D, etc.). The parameter commands provide specifics about the techniques to be used in the flow visualization (for example, particle plot, particle location and color, contour plot, etc.). All of CINEMA's commands are described in detail in Section I.

After the commands have been entered and the CINEMA software has executed, an intermediate file is created. This intermediate file is created by the DISSPLA routines and contains graphics information in a coded form. Several DISSPLA post-processors are available that can accept this intermediate file as input. Two post-processors are of particular interest to a CINEMA user: (1) TEK4010 for use with a TEKTRONIX 401X series scope, and (2) FR80 for use with an Information International, Inc., FR-80 film/microfilm recorder. After the intermediate file has been generated it is possible to examine it on a TEKTRONIX 4014 scope using TEK4010. If the user is satisfied with the contents of the

intermediate file he can then execute the FR-80 post-processor that builds a 7-track tape with standard FR-80 graphics commands. This 7-track tape can then be processed on an FR-80 where film is produced.

The final step in using CINEMA is film processing. The FR-80 on WPAFB is in the Foreign Technology Division (FTD), and produces 35mm film. This 35mm film is taken to MAC audio-visual detachment on WPAFB where 16mm film is created from the 35mm film.

The remainder of this user's guide is in five sections:

Section I	Command Formats
Section II	DISSPLA Post-Processors
Section III	FR-80 Notes
Section IV	Film Processing
Section V	Example Input/Output

The following diagram indicates the organization of the steps in using CINEMA.

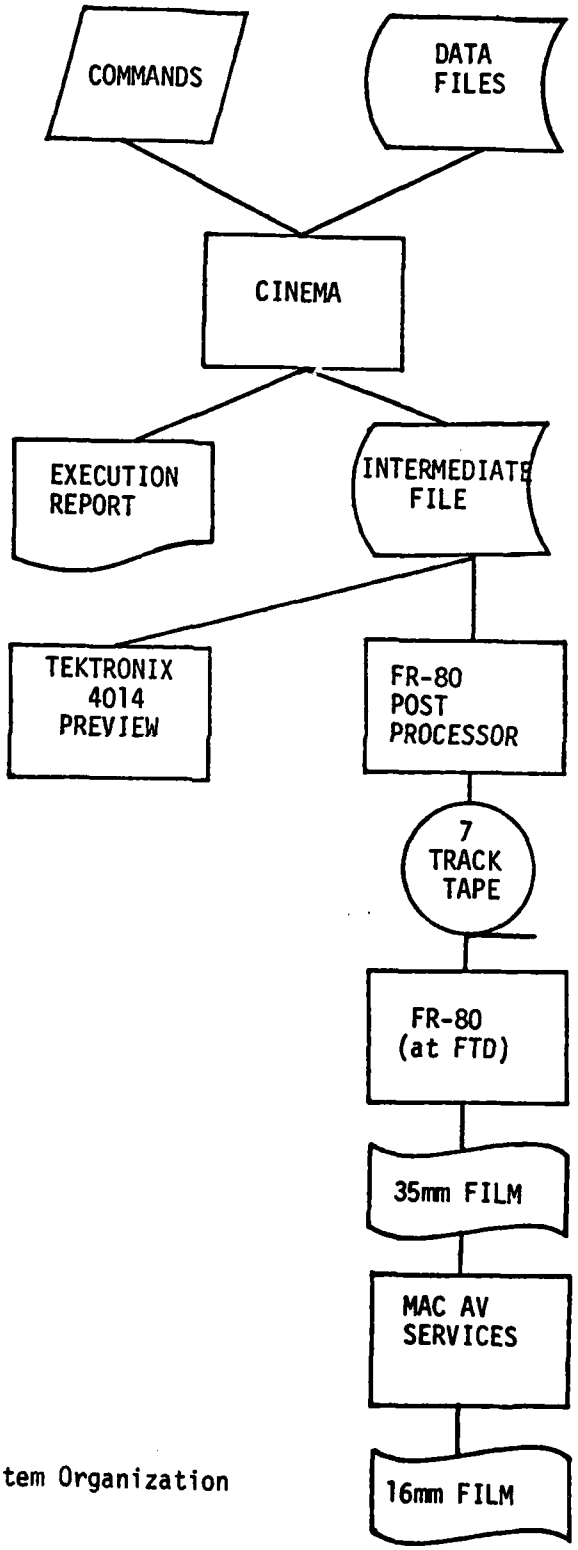


Figure 2. CINEMA System Organization

Section I

Command Formats

This command language was designed to be easy to use and yet very capable and flexible. Each command was included for a specific purpose. The format descriptions are grouped into three categories - Mode Commands, Parameter Commands, and File Commands.

CINEMA uses a semi-colon as an end of line terminator and all of the examples show a semi-colon. If the user does not include one with the command then CINEMA will automatically place one at the end of the input line. CINEMA requires that all command lines be limited to 79 characters. Comments are delimited by /* and */.

Parameter Commands

1. Data Description

   DATA, option;

   option--

   a. DYNAMIC - the data changes with time (default).

   b. STATIC - the data does not change with time.

   Description: If the data does not change with time then CINEMA will only read data from the flow field file once. If the data is dynamic then CINEMA will read data from the flow field file until all requested processing has been done or until an end of file is reached.

   Example: DATA, STATIC;

2. Axis Request

   AXIS, option;

   option--

   a. YES - display an axis in each frame.

b. No - do not display an axis (default).

Example: AXIS, YES;

3. Scaling

SCALE, number;

Number - any real number (default 1.0).

Description: This allows the user to scale all of the information in the frame by the indicated value.

Example: SCALE, 0.5;

4. Eye Point Location

EYEPT, location;

Location - the x, y, z, location (real numbers) of the observer (default -10., -10., 10.).

Description: This command allows the user to specify the location of the reference point for a 3-D plot. The eye-point is not used for 2-D plots.

Example: EYEPT, 100., 100., -7.;

5. Specifying the dimension of the problem.

SPACE, option;

option--

a. 2 - a 2-D aerodynamic problem (default)

b. 3 - a 3-D aerodynamic problem

Description: To do a 3-D particle plot this command must be used to tell CINEMA that this is a 3-D problem.

Example: SPACE, 3;

6. Range

RANGE, range-list;

range-list--list of 4 or 6 real numbers.

a. xmin, xmax, ymin, ymax for 2-D problem

b. xmin, xmax, ymin, ymax for 3-D problem (default -0., 1., 0., 1., 0., 1.)

Description: This specifies the range of coordinate values for all the data to be displayed. Data falling outside these ranges do not appear.

Example: RANGE, 0., 5., 2., 6.;

7. Viewing options

VIEW, option;

option--

a. SURFACE - view only the surface file data

b. FLOW - view only the flow field data

c. BOTH - view both surface and flow field data (default)

Description: This allows the user to specify what data CINEMA is to display. Although both surface and flow field files may have been described, this command controls what appears in the frames.

Examples: VIEW, BOTH;
VIEW, FLOW:

8. Terminate Commands

END.;

Description: This command tells CINEMA that there are no more commands to be read.

Example: END.;

Mode Commands

1. 2-D Contouring

CONTOUR, color, matrix-list, c-min, c-inc, c-max;

color--color specifies the color of the contour lines.

R - red
B - blue
G - green
Y - yellow
C - cyan
M - magenta

matrix-list - this provides the names of the three data arrays

c-min - the minimum data value to be contoured.(real number)
c-inc - the increment added to c-min.(real number)
c-max - the maximum data value to be contoured. (real number).

Description: The 2-D contouring routine is a package developed by Kervyn Mach and James Petty (6). The matrix list provides the names of the data arrays read from the flow field file. These matrices must be defined by MATRIX commands. The first data matrix is the x-location matrix, the second is the y-location matrix, and the third is the functional value matrix. The total list of functional values to be contoured is bounded by c-min and c-max and intermediate values are obtained by adding an integer multiple of c-inc to c-min. For example, if c-min = 1.0, c-inc = 2.0, and c-max = 5.0, then the contour lines drawn would represent values 1.0, 3.0, and 5.0.

Example: CONTOUR, M, XX, YY, FF, .0017, .001, .035;

2. Particle Plots

PARTICLE, n, t, matrix-list;

n - total number of particles (integer number)
t - time-increment (real number)
matrix-list - this provides the names of the 4 (for 2-D) or 6 (for 3-D) data matrices.

Description: The time increment, t, is used to compute the actual time step involved in calculating particle movement. The data matrix that contains the x-component of the velocity is searched for its maximum value, VXMAX. Let VXMAX be at mesh point I, J, then the actual time step, t-step is computed as t-step = t-inc * (X(I + 1, J) - X(I, J))/VXMAX, where the "X" array holds the x values. This technique easily generalizes to three dimensions since only the x-component of the velocity and the x values are used. The intent of this calculation is to allow the user to easily restrict particle movement so that a particle doesn't move too far in one time step.

The matrix-list for 2-D is X matrix, Y matrix, VX matrix and VY matrix, where VX represents the X component of the velocity and the VY represents the Y component of the velocity. Again, these names must appear on MATRIX commands. For 3-D the list is X matrix, Y matrix, Z matrix, VX matrix, VY matrix, and VZ matrix. Note: The SPACE command must be used for 3-D particle plots.

Example: PARTICLE, 14, .75, XX, YY, ZZ, VX, VY, VZ;

3. Marker Particle

MARKER, n, color, location;

n - particle's number

color--

R - red
B - blue
G - green
Y - yellow
C - cyan
M - magenta

location - particle's location in the mesh is given by using indices into the coordinate arrays.

I, J - for 2-D
I, J, K - for 3-D

Decription: The particle number is used as an index into an array and therefore must be greater than 0 and less than or equal to the maximum number of particles. The particle number is used in other commands such as the CONNECT and TRAIL commands.

The location of a particle is not given in cartesian coordinates but in terms of indices into the location arrays. The x location of a particle when first described is X-ARRAY (I, J, K) in a 3-D problem, and the user supplies the values for I, J, K. The X-ARRAY matrix is the one described in the PARTICLE command. The Y and Z locations are determined in the same manner.

Example: MARKER, 1, Y, 7, 4, 12;

4. Connect Particles

CONNECT, n, m;

n - particle number (integer number)
m - particle number (integer number)

Description: This causes CINEMA to draw a line between particles n and m every time these particles are plotted. This must be used in conjunction with PARTICLE and MARKER commands, and the particle numbers must match numbers used on MARKER commands.

Example: CONNECT, 1, 7;

5. Trail Particles

TRAIL, n;

n - particle number (integer number)

Description: This directs CINEMA to keep track of every position of particle n starting with its location in the next frame. CINEMA will draw a line through all the recorded locations of this particle to display a trace of this particle's path. This is used in conjunction with a MAXTRAILS command.

Example: TRAIL, 3;

6. Particle Trail Parameters

MAXTRAILS, n, m;

n - maximum number of particles to be trailed (integer value)
m - maximum number of times the particle will be moved (integer value)

Description: This is used if a particle plot is being done and some particles are to be "trailed" (i.e., each trailed particle has a line drawn connecting its previous locations). These values must be specified before trying to generate frames with trailed particles.

Example: MAXTRAILS, 2, 10;

7. Maximum Number of Particle Connections

MAXCONNECT, n;

n - the maximum number of connections between particles (integer number).

Description: This must be used if a particle plot is being done and connections between particles are requested.

Example: MAXCONNECT, 7;

8. Save Command

SAVE, name (,PURGE);

Description: This saves the commands on a permanent file with the name indicated (up to 8 alphanumeric characters). If the user wishes the highest cycle of the permanent file of the same name purged he must include the PURGE parameter.

Examples: SAVE, OLDCMDS, PURGE;
SAVE, OLDFILE;

9. Frame Control

FRAMES, n, m, k;

n - number of first frame (default 1, integer number)
m - number of last frame (default 10, integer number)
k - optionally included frame increment (default 1, integer number)

Description: This command directs CINEMA to create individual frames on the intermediate file. The parameters indicate how many frames to generate. If the data type is STATIC (see DATA command) then the flow field data file is read only once, and the requested number of frames are generated using that single block of data. If the data type is DYNAMIC then the flow field file is read each time a frame is created, and the requested frames are created using one block of data per frame.

Example: FRAMES, 1, 10;

This will produce ten frames. If the data is STATIC (time-independent) then the ten frames will be produced using a single block of data (one FORTRAN BUFFER IN) from the flow field data file. If the data is DYNAMIC (time-independent) then the ten frames will be produced using ten blocks of data (ten FORTRAN BUFFER IN) from the flow field file.

Example: FRAMES, 1, 5, 2;

This will produce three frames. Again STATIC data involves only one data read so this command produces the same effect as FRAMES, 1, 3;. However, if the data is DYNAMIC then five FORTRAN BUFFER IN's are done but the information from data blocks one, three, and five are used to generate the three frames (data blocks two and four are completely ignored).

10. Rewind Flow Field Data File

REWIND;

Description: This command rewinds the flow field that has been described by the user with a FLOW command.

Example: REWIND;

File Commands

1. Command File

COMMAND, pfname;

pfname - the name of the permanent file (up to 8 characters)

Description: This provides the name of the permanent file that contains CINEMA commands. Two ways to build this file are: (1) use the CDC editor and (2) use the CINEMA save command. Upon receipt of this command CINEMA uses the BATTELLE DISK FILE MANIPULATION routines [1] to attach pfname. For the permanent file attach the SN and ID parameters are not specified, therefore, the file name must be catalogued using the ID and SN with which the user is logged in. If the file is already attached, an error occurs. CINEMA will read commands from the file INPUT unless this command is used. CINEMA uses TAPE9 as the local file name for a command file.

Example: COMMAND, MYCMDS;

2. Flow Field Data File

FLOW, name, type, block-size;

name - file name (up to 8 characters)
type - DISK or TAPE
block-size - the size of a physical record on the file (integer constant).

Description: This command describes the flow field data file. The type parameter specifies what file handling CINEMA will do. If the type specified is tape, all file handling (such as attaching, etc.) is to be done by the user. For data files on magnetic tape or those disk files catalogued with different ID and SN parameters the user should use TAPE as the type parameter. He may then use a REQUEST, LABEL or ATTACH command and make the flow field data available as local file TAPE7 since CINEMA uses TAPE7 as the flow field file. If the type specified is DISK then CINEMA will attach permanent file "name" using the BATTELLE FILE MANIPULATION routines.

The block-size parameter is used regardless of the type parameter. CINEMA uses a FORTRAN BUFFER IN to read the flow field file and block-size words are read with each BUFFER IN. (Since BUFFER IN is used to read the data it is strongly suggested that a BUFFER OUT statement be used in the building of the flow field file.) The block-size parameter specifies the size of a physical record. The

The user is given a great deal of flexibility in describing how that physical record is to be broken up with the use of a MATRIX command below.

Examples: FLOW, CAVITY, DISK, 3000;
FLOW, XXX, TAPE, 1432;

3. Matrix Specification

MATRIX, name, no., index-list;

name - unique name for the matrix (up to 10 characters)
no. - the number of indices (integer number)
index-list - list of maximum values for indices (integer constants separated by commas)

Description: This allows the user to partition the physical record read from the flow field file. Several data matrices will probably be included within a single physical record. The order of the MATRIX commands is very important as this indicates the order of the matrices within the input record.

The no. parameter indicates if the matrix is two or three dimensional, and the index-list provides the upper limits of the indices.

The user is cautioned to carefully order the MATRIX commands and ensure that these matrices are correctly described (i.e., maximum values for indices correctly specified and that 2-D matrices are used for 2-D problems, etc.). Additionally, the name parameter used here should correspond to the names used on the CONTOUR or PARTICLE command.

Examples: MATRIX, XX, 2, 10, 20;
Analogous to DIMENSION XX(10, 20) in FORTRAN

MATRIX, YY, 3, 5, 5, 6;
Analogous to DIMENSION YY(5, 5, 6) in FORTRAN

4. Surface Description File

SURFACE, name, color;

name - the permanent file containing the surface description (up to 8 characters)
color - the color of the surface

R - red
B - blue
G - green
Y - yellow
C - cyan
M - magenta

Description: The surface is the object in the flow field such as a wing, a corner, etc. The file "name" is attached by CINEMA and is used as local file TAPE8. The surface is drawn in the color specified. The SPACE command must be set correctly before this command is issued since 2-D surfaces are handled much differently than 3-D surfaces.

2-D surfaces are described by listing points (X, Y locations) along the edges of the surface (similar to a child's connect the dot picture). CINEMA draws the outline of the surface by starting at the first point and connecting point to point to the last point. The interior of the surface is shaded. The points are read using a FORTRAN formatted read; the format is 2F10.5.

A 3-D surface is described by listing points (X, Y, Z) in groups of four. These four points must define a planar polygon, and they must be listed in a clock-wise order. The user must manually divide the 3-D surface into these planar polygons and list the four corner points of all the polygons in the required clock-wise order. CINEMA will read the user's surface data file and use DISSPLA to form a composite surface. Hidden line processing is done by DISSPLA. The user can specify the surface using both triangles and quadrilaterals, but the triangles must be specified using four points -- simply make two of them the same.

A good description of how DISSPLA forms a 3-D surface is in reference 3. It is important to remember that DISSPLA builds 3-D surfaces with techniques that consider the Z coordinate as a function of X and Y and therefore, the 3-D surface will rise or fall out of the X-Y plane. In reference 3 a good analogy is given to help describe the types of 3-D surfaces created by DISSPLA routines. The analogy is to imagine a tight-fitting stocking being draped over the data points.

The data points are read using a FORTRAN formatted read; the format is 3F10.5.

Example: SURFACE, CSURF, M;

Section II

DISSPLA Post-Processor Notes

CINEMA uses the DISSPLA software to generate its graphics output, and as a result of the use of DISSPLA an intermediate plot file is generated. This intermediate plot file, often called a metafile, is designed to be device independent. It contains graphics information in a coded format, therefore to obtain the desired output a DISSPLA post-processor program is used to convert the information on the metafile into the necessary commands for a particular graphics device.

There are several post-processors available with the DISSPLA library on the CDC machines: ONLINE and OFFLINE for use with a CALCOMP plotter, TEK4010 for use with a TEKTRONIX 401X scope, and others. Reference 2 provides a list. A post-processor program for an FR-80 film/microfilm recorder was obtained from the ASD computer center. After some modifications to the program and tests on an FR-80, it was usable.

After CINEMA has been executed, the user will have access to a DISSPLA metafile and post-processor programs can be used to process the file and create graphics output. Often a CINEMA user would like to preview his graphics output at a terminal before using the FR-80 post-processor to generate an FR-80 tape. This can be done at a TEKTRONIX 4010 or 4014 scope. After previewing the file the user may build a 7-track tape for an FR-80 by using the FR-80 post-processor. This 7-track tape can then be taken to the FR-80 for the generation of film output.

TEK4010 and the FR-80 post-processor are the most commonly used post-processors after execution of CINEMA. These programs are executed with the following command lines:

TEK4010 (plotin, dir.file, list file)

FR-80 (plotin, dir.file, list file, plotout)

where

plotin - intermediate plot file to be read. It must be in DISSPLA format and the default name is PLFILE.

dir.file - contains user's processing directives. Default name is INPUT.

list file - post-processor list file. Default name is OUTPUT.

plotout - output file with device commands.

The following command formats have been extracted from reference 5 to provide the CINEMA user a handy reference to commonly used post-processor commands. All DISSPLA post-processors use the same commands and reference 5 has a complete list and description of all the DISSPLA post-processor commands. The DISSPLA commands BGNPL (begin plot) and ENDPL (end plot) delimit a single plot, and all the graphics output generated between these commands create a single plot or frame.

DRAW = 'list'

where 'list' consists of elements, separated by commas, of the form

a. n - plot n will be plotted

b. n-m - plots n through m will be plotted

c. n-m(k) - implied loop. Plot n to m in steps of size k, i.e., plots n, n+k, n+2k...m.

d. n-END or n-END(k) - same as b and c with m being the last plot generated.

Example: DRAW = 22-END, 1, 7-9, 12-20(4)

The MODIFY command has several options; however, the CORNER and OVERPLOT options can be particularly handy to a CINEMA user. The modify specification

OVERPLOT

will prevent screen erasure or frame advance which otherwise would take place before the drawing of the plot.

CORNER = x, y

will locate the corner of the specified plots at x, y (in inches) from the location of a previous plot if one has already been plotted. It will locate the corner at x, y from the corner of the plotting surface if this is the first plot plotted onto a fresh screen or frame.

MODIFY = 2-5 (OVERPLOT * CORNER = 0, 0) * DRAW = 2-5

NOTE: CORNER = 0, 0 was used to have the plots plotted on top of each other.

Finally, post-processor commands are terminated by:

1. An all blank card image or input line;
2. An end-of-file; or
3. A $ sign is encountered.

Commands may be separated by an asterisk or a comma (an asterisk makes the command string more readable).

Section III

FR-80 Notes

This section contains information on the use of the Information International, Inc. (III) FR-80 film/microfilm recorder. The FR-80 on Wright-Patterson AFB is owned and operated by the Aeronautical Systems Division's Foreign Technology Division (FTD). They have the capability of producing 35mm sprocketed black and white (B&W) film. FTD has an agreement with the AFFDL to maintain the 35mm camera through 30 June 1980. Present plans are to use an III COMP-80 with a 16mm camera after that time. This discussion covers use of the FR-80 at FTD.

There are several types of output available with the FR-80 at FTD. They can produce paper plots, 35mm B&W film, and 16mm B&W film. Paper plots are often used for debugging plots, and FTD operators use the program named MON;HCFR80$J to get paper plots (often called hard copy). The FTD operators often produce paper plots using the FR-80 and they are familiar with the HCFR80 program. Although they have done a lot of hard copy work they have done very little 35mm work. The 35mm camera was used for CINEMA work throughout this thesis because it uses sprocketed film. To use the 35mm camera for CINEMA output the FTD operators must use the program named MON;35FR80$J. If a CINEMA user provides the FR-80 operator with this program name the operator can produce the desired output. However, there are several programs within the FR-80 operating system that can be used for 35mm work, and the CINEMA user must be sure to specify which program to use.

While either of these programs is running it is possible for the operator to enter commands to alter the plots or frames being created by the FR-80. There are two commands, OFFSET and SETSIZE, that an operator can use to improve the quality of the film being produced by the FR-80.

The OFFSET command is used to specify the location of the lower left-hand corner of the frame in FR-80 screen coordinates. By using this command the plot can be centered on the 35mm film. Centering the picture results in a much better movie. The default values are OFFSET, 0, 0; the first number is the x value and the second number is the y value.

The other command that can be used to improve the quality of a CINEMA film is the SETSIZE command. This command is very similar to a scale command available with many graphics packages. The default value is SETSIZE, 16384 (16384 is the resolution in scope points of an FR-80). To reduce the size of the plot the SETSIZE command can be used with a number smaller than 16384, such as SETSIZE, 11000. Any number less than or equal to 16384 can be used, so the plot size can be changed to just about any size.

Both of these commands (and several others) may be included on the 7-track input data tape. Currently, the DISSPLA FR-80 post-processor does not issue a SETSIZE command, but it does issue an OFFSET command. It sets the lower, left-hand corner of the frame at the corner of the camera aperature. When doing paper plots this OFFSET setting is not correct and the best procedure is to have the operator use an OFFSET,3000,3000 after the first frame is plotted. The reason for using an OFFSET command after the first

frame is that the last OFFSET command received by the program is the one in effect. Therefore, the plotting of the first frame will always be controlled by the OFFSET command on the 7-track tape. After the first frame is plotted, the operator can issue the OFFSET, 3000,3000 command so that the remaining paper plots will be positioned correctly. Remember, no additional OFFSET command is necessary for 35mm processing.

FTD has job submission sheets to use when requesting FR-80 work, and the user should use this sheet to communicate with the FR-80 operator. It is important to specify the program (HCFR80 or 35FR80) and the OFFSET and SETSIZE parameters on the job sheet. Additionally, the FR-80 creates a teletype listing of any operator actions taken during a job, and the user may want to request the operator to return the teletype listing with other FR-80 output.

Section IV

Film Processing

After using the FR-80 to create 35mm black and white (B&W) film, the last step is to create 16mm color film from the 35mm B&W film. CINEMA creates its graphics output in a special way so that the 35mm B&W film created on the FR-80 can be used to create 16mm color film. After explaining the processing necessary to create color film from B&W film, there is a brief discussion of film processing capabilities available on WPAFB.

In order to create color film from B&W film CINEMA creates the B&W film in an unusual way. The three additive primary colors (red, blue, green) can be combined in pairs to create yellow, cyan and magenta. Therefore, CINEMA allows the user to request one of those six colors for his plots. To create a single color frame CINEMA creates up to three B&W frames on the FR-80. These B&W frames contain information to be plotted in the additive primary colors (one frame for each color). CINEMA will generate the number of frames required (i.e., if only red is requested then only one B&W frame is created, if yellow is requested then two B&W frames are created). CINEMA provides a summary listing detailing how many B&W frames will be generated per 16mm frames. What remains then is to use color filters to expose the B&W frames and also superimpose these frames. In this way all six colors are available.

Fortunately, the complicated process of reducing 35mm to 16mm, exposing individual frames with color filters, and superimposing

frames can be done at WPAFB. The Military Airlift Command (MAC) audio-visual detachment on WPAFB has some excellent equipment and facilities. In particular, they have an OXBURY camera that can do all three of the necessary transformations. The OXBURY camera can reduce 35mm film to 16mm, it can expose frames with color filters, and it can be used to superimpose frames. Additionally, it can be used to multiply expose frames. It is much more cost effective to generate one frame with an FR-80 and then make multiple copies on the OXBURY than it is to make copies of a frame using the FR-80.

Because the OXBURY is a very capable piece of equipment, it is essential that the CINEMA user make clear to the OXBURY operator what he wants done. It is necessary to fill out a work request to have the OXBURY work done, and the user is advised to take some time and clearly state his needs.

Section V

Example Input/Output

This section has two purposes: (1) provide some examples of typical CINEMA input, and (2) provide some examples of CINEMA output. Two sets of input commands are listed in order to give a prospective user an idea of how several commands can be used together. These two input examples are not meant to show every possibility, there is too much flexibility in the command language for that. Providing examples of CINEMA output is considerably more difficult than providing examples of input. Since CINEMA is designed to produce movies they are CINEMA's primary type of output. Because it is physically impossible to show movies in a document like this, the next best approach is used. Several frames of CINEMA output are included along with a description of the data displayed.

The first input example produces 103 frames of 2-D contour plots. The data is dynamic (default). The range of the x values in the data is from .45 to 1.5, and the range of the y values is from 0. to 3. There is a permanent file named CSURF that contains the 2-D surface description, and it is to be displayed in red. There is a magnetic tape that contains the flow field data. The contour lines are to be displayed in blue.

```
/* 2-D CONTOUR PLOT */
RANGE,.45,1.5,0.,3.;
SURFACE,CSURF,R;
FLOW,CAVITY,TAPE,12168;
MATRIX,XX,2,78,52;
MATRIX,YY,2,78,52;
MATRIX,F,2,78,52;
CONTOUR,B,XX,YY,F,.000601,.00005,.0012;
FRAMES,1,103;
END.
```

The second example produces 10 frames of 3-D particle plots. The data is static. The range of data values is as follows: x = 10. to 45., y = 10. to 45., and z = 25. to 40. The eyepoint location is moved from the default position to -3.,-5.,1.5. There is no 3-D surface; only the flow field data and an axis are displayed. There are seven particles and two are trailed. The flow field is on a permanent disk file.

```
/* 3-D PARTICLE PLOT */
DATA,STATIC;
RANGE,10.,45.,10.,45.,25.,40.;
EYEPT,-3.,-5.,1.5;
AXIS,YES;
SPACE,3;
VIEW,FLOW;
MAXTRAILS,2,10;
FLOW,PDATA,DISK,2940;
MATRIX,XX,3,10,7,7;
MATRIX,YY,3,10,7,7;
MATRIX,ZZ,3,10,7,7;
MATRIX,U,3,10,7,7;
MATRIX,V,3,10,7,7;
MATRIX,W,3,10,7,7;
PARTICLE,7,.25,XX,YY,ZZ,U,V,W;
MARKER,1,G,1,1,1;
MARKER,2,G,1,4,1;
MARKER,3,G,2,7,1;
MARKER,4,B,2,6,1;
MARKER,5,B,3,6,1;
MARKER,6,B,3,7,2;
MARKER,7,B,4,6,4;
TRAIL,1;
TRAIL,2;
FRAMES,1,10;
END.
```

Input command sequences such as this produced the sample output presented next. The first output example is the graphical display of a well known aerodynamic problem -- flow around a rotating cylinder. The air is moving from left to right, and the cylinder is rotating counter-clockwise. Figure 3 shows streamlines around the cylinder. These streamlines are the mathematically predicted particle paths, and

are displayed using the 2-D contouring capability of CINEMA. Figures 4-6 show particles and their traces in the flow field. The rotating cylinder example output was created using the TEKTRONIX scope post-processor for DISSPLA graphics files.

The second output example, Figures 7-9, is a set of frames from a movie produced for an AFFDL study of a buffetting problem in an open bomb bay. The contour lines represent lines of constant air density. This movie was compared with experimental results from a water tunnel and enabled the researchers to verify that their numerical procedures could accurately predict actual physical phenomena. In this situation CINEMA was instrumental in providing this information.

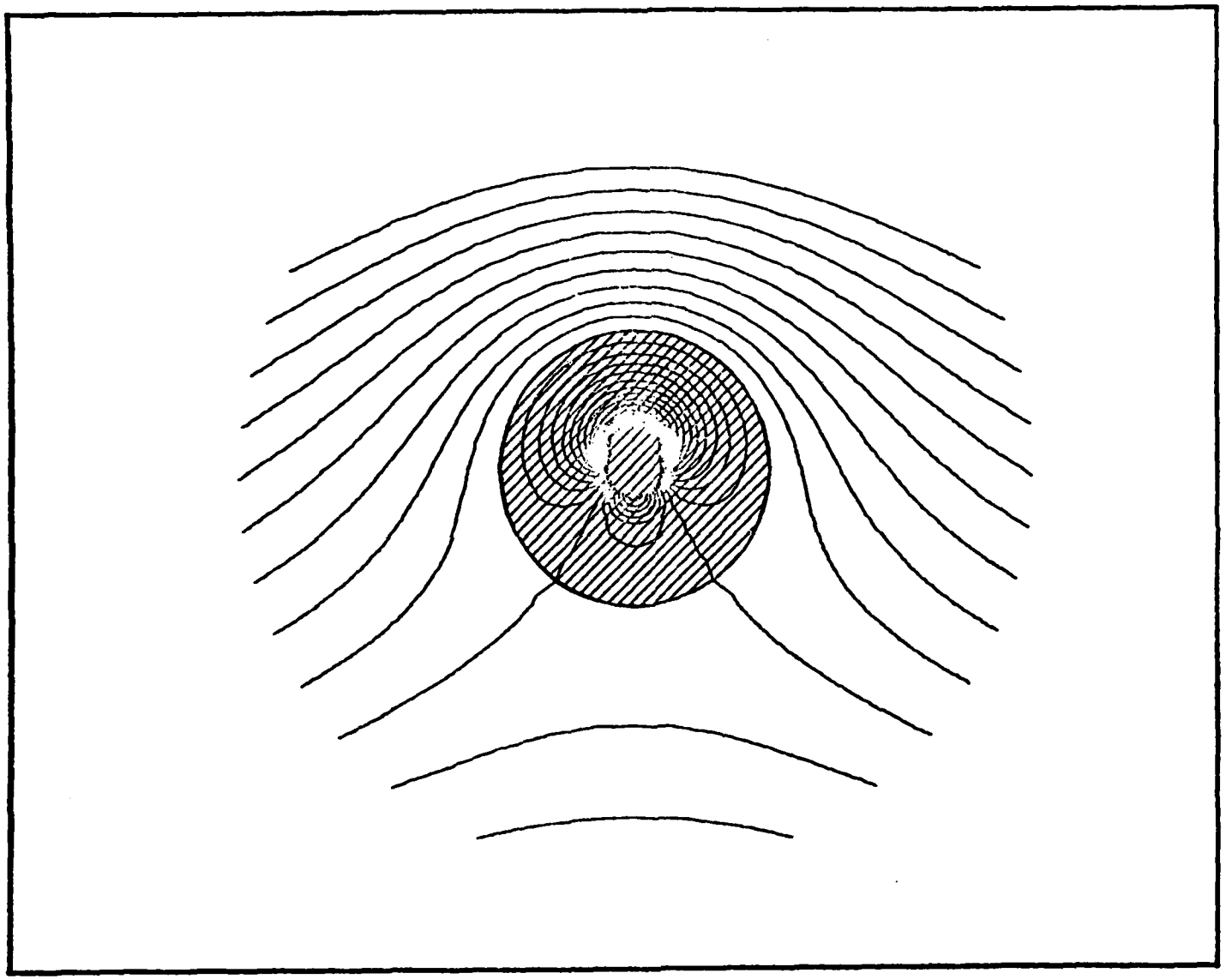


Figure 3. Rotating Cylinder Streamlines

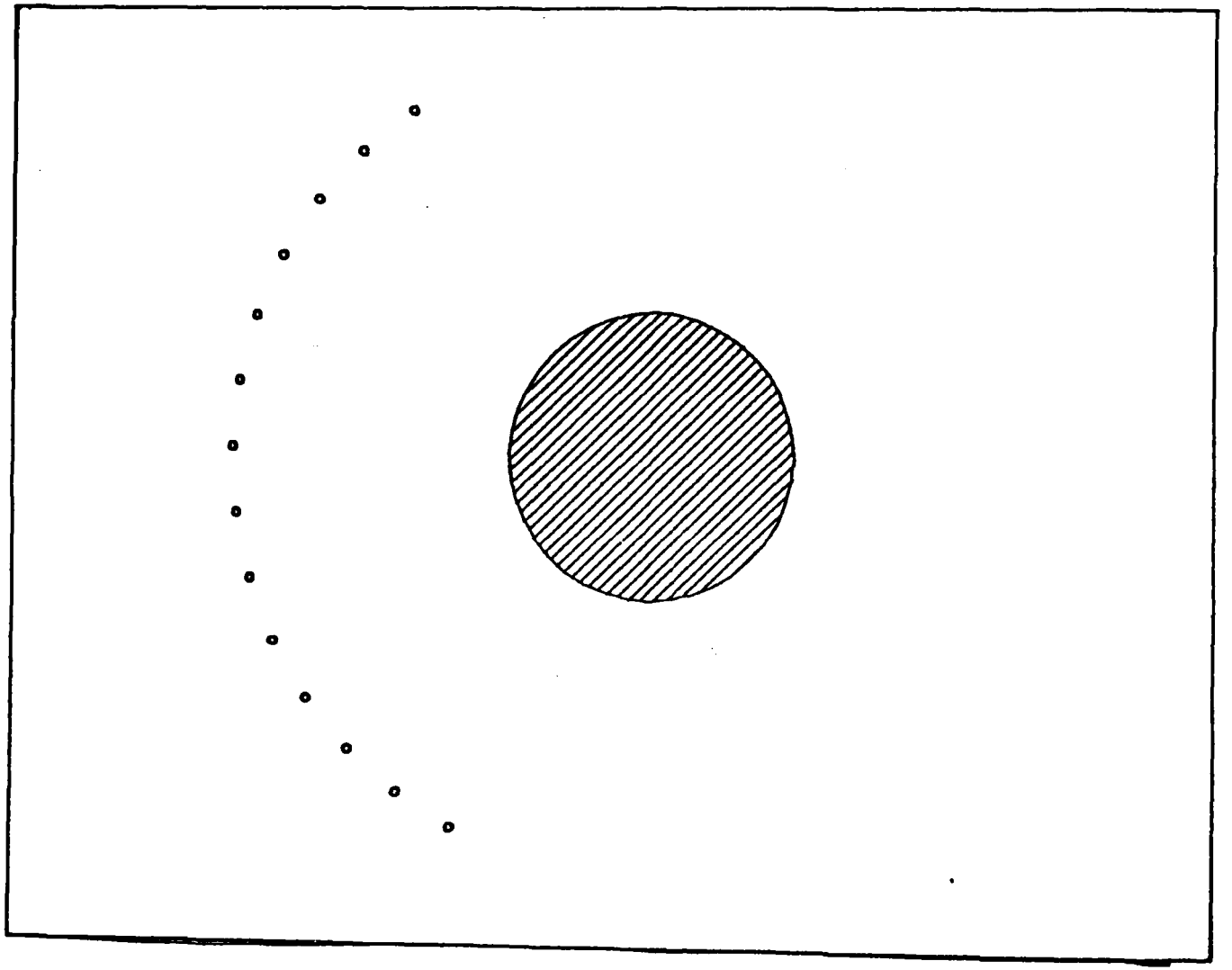


Figure 4. Rotating Cylinder, Starting Particle Locations

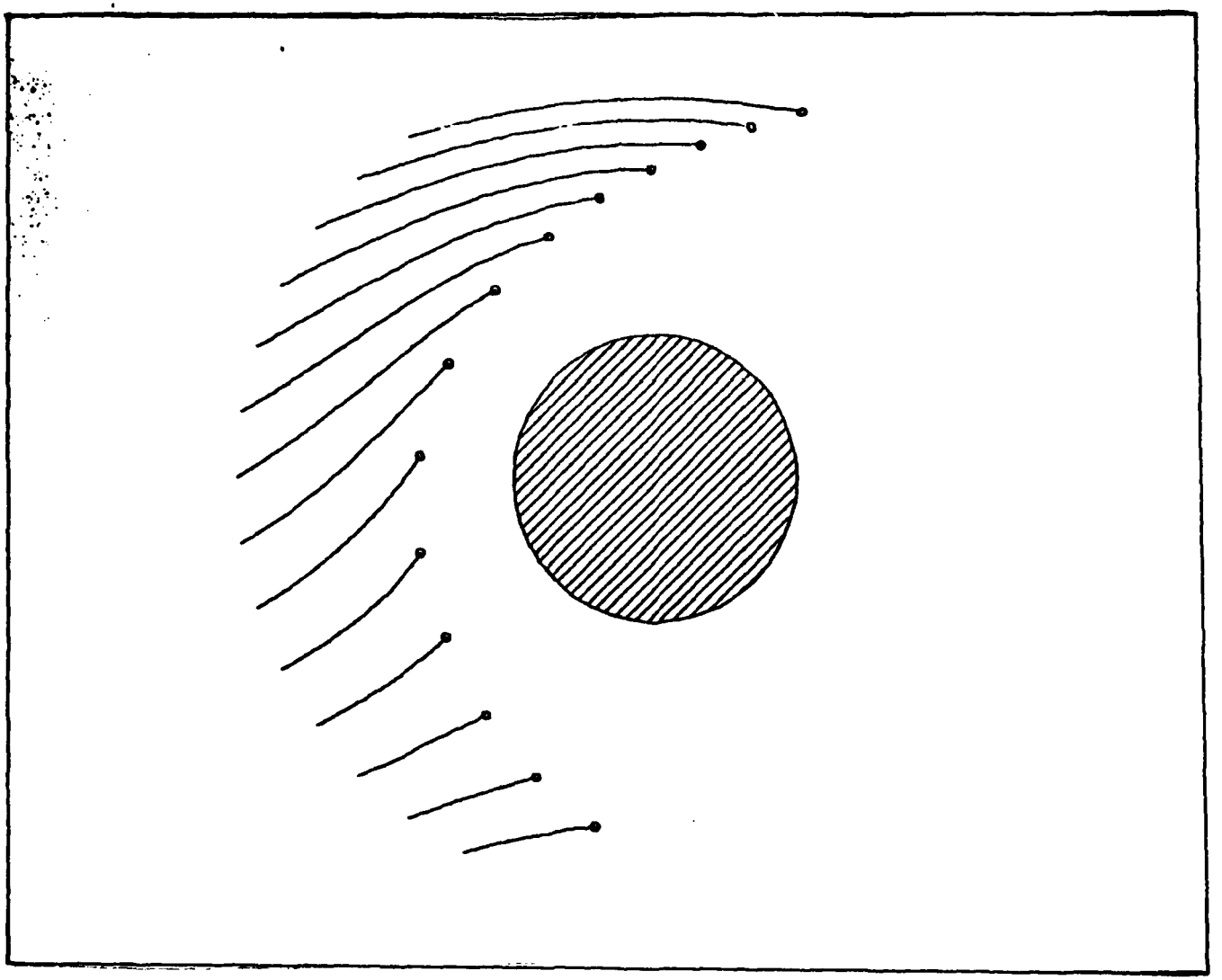


Figure 5. Rotating Cylinder, Particle Locations in 50th Frame

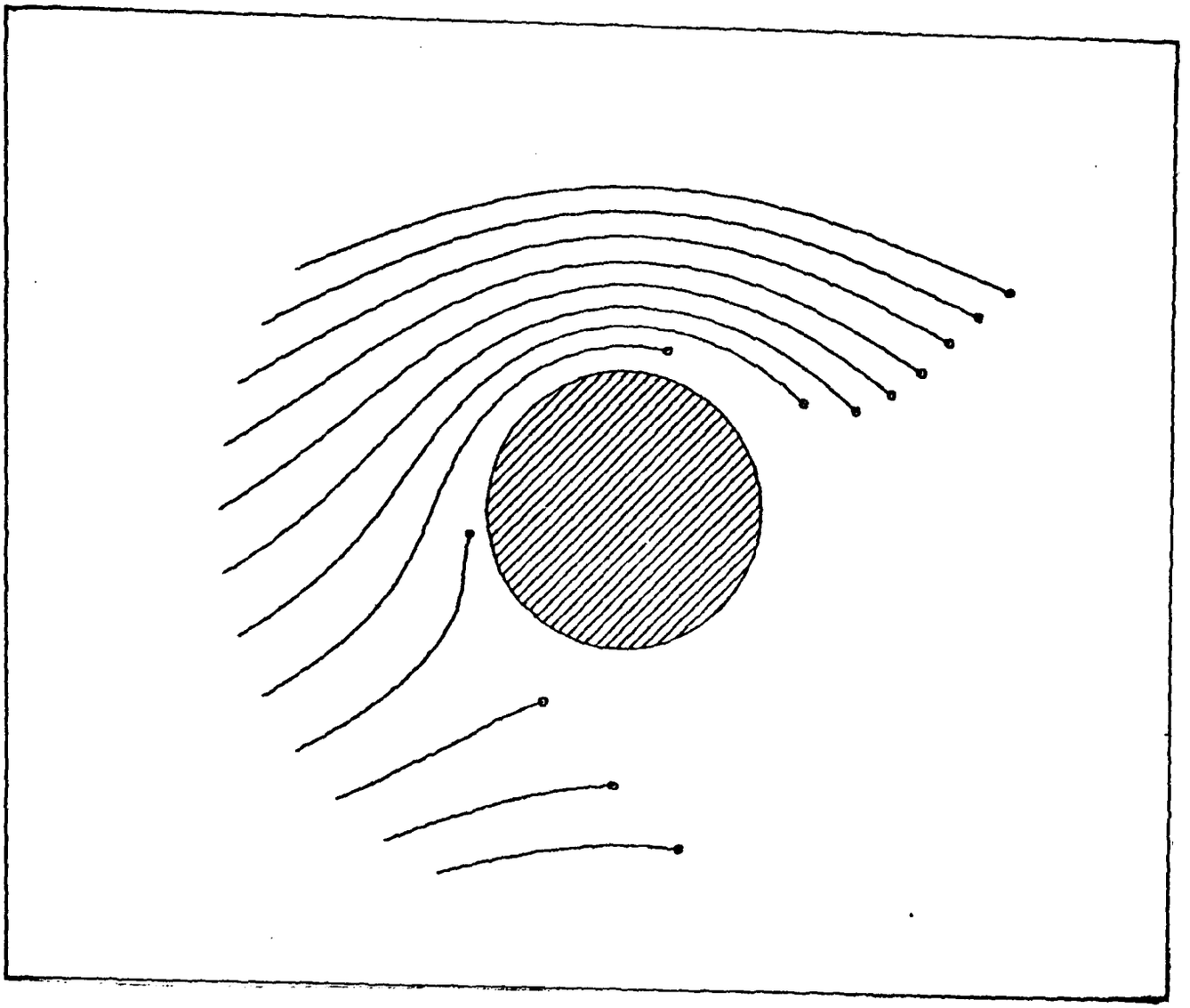


Figure 6. Rotating Cylinder, Particle Locations in 100th Frame

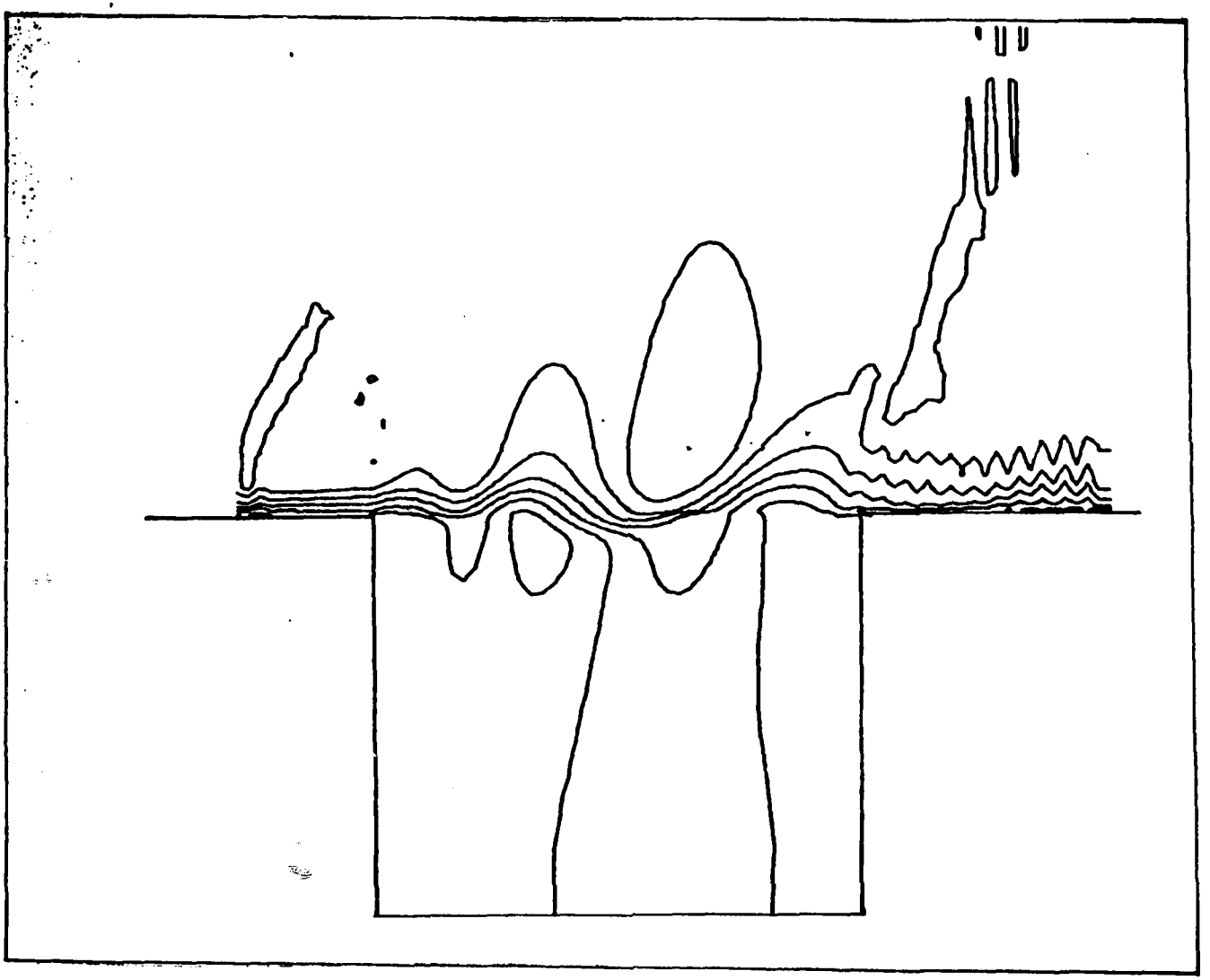


Figure 7. 2-D Cavity Contour Lines, Frame 1

AD-A080 359 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 20/4
THE GRAPHICAL DISPLAY OF MULTI-DIMENSIONAL AERODYNAMIC FLOW FIE--ETC(U)
DEC 79 E P AMBURN
UNCLASSIFIED AFIT/GCS/MA/79D-1 NL

END
DATE FILMED
3-80
DTIC

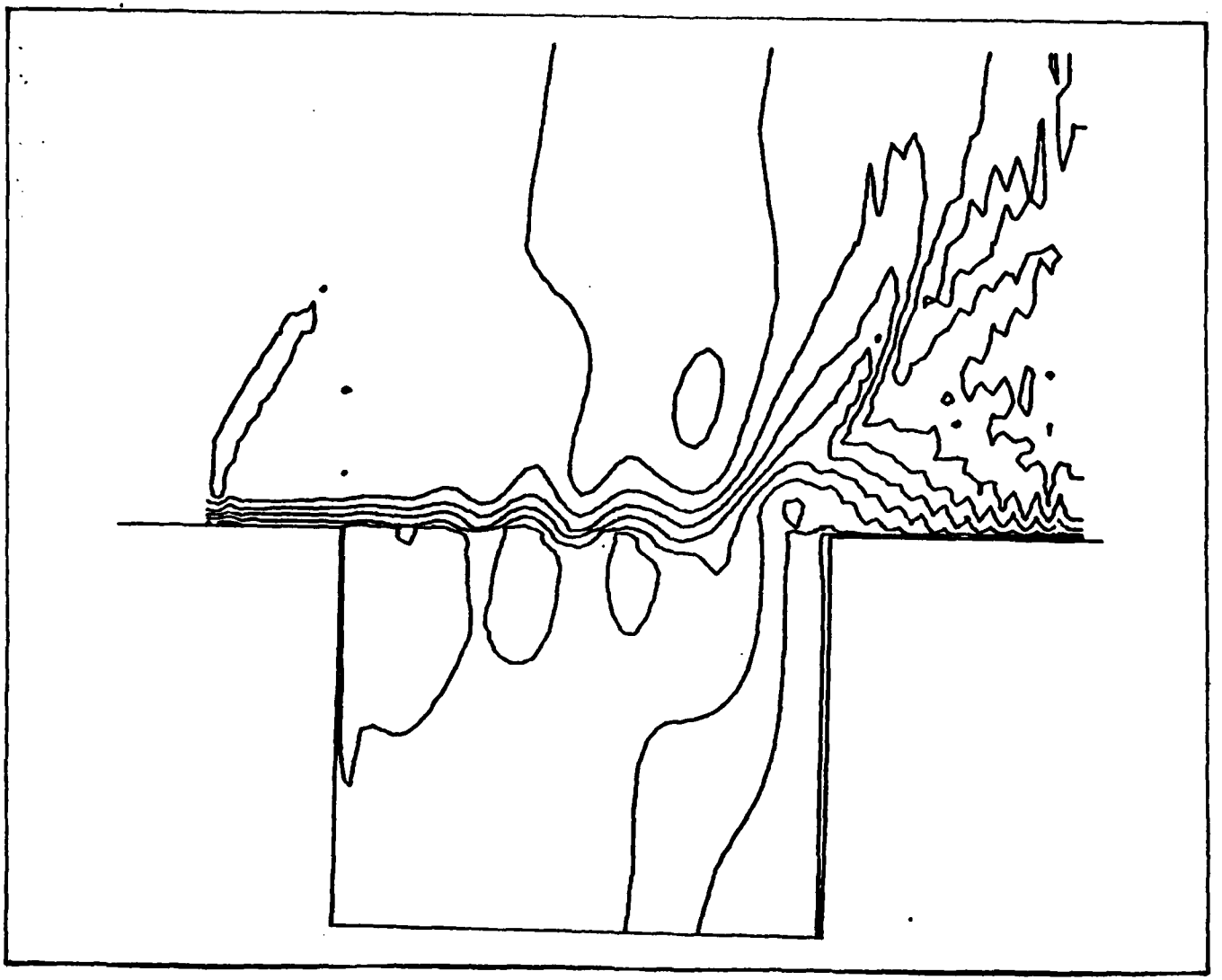


Figure 8. 2-D Cavity Contour Lines, Frame 50

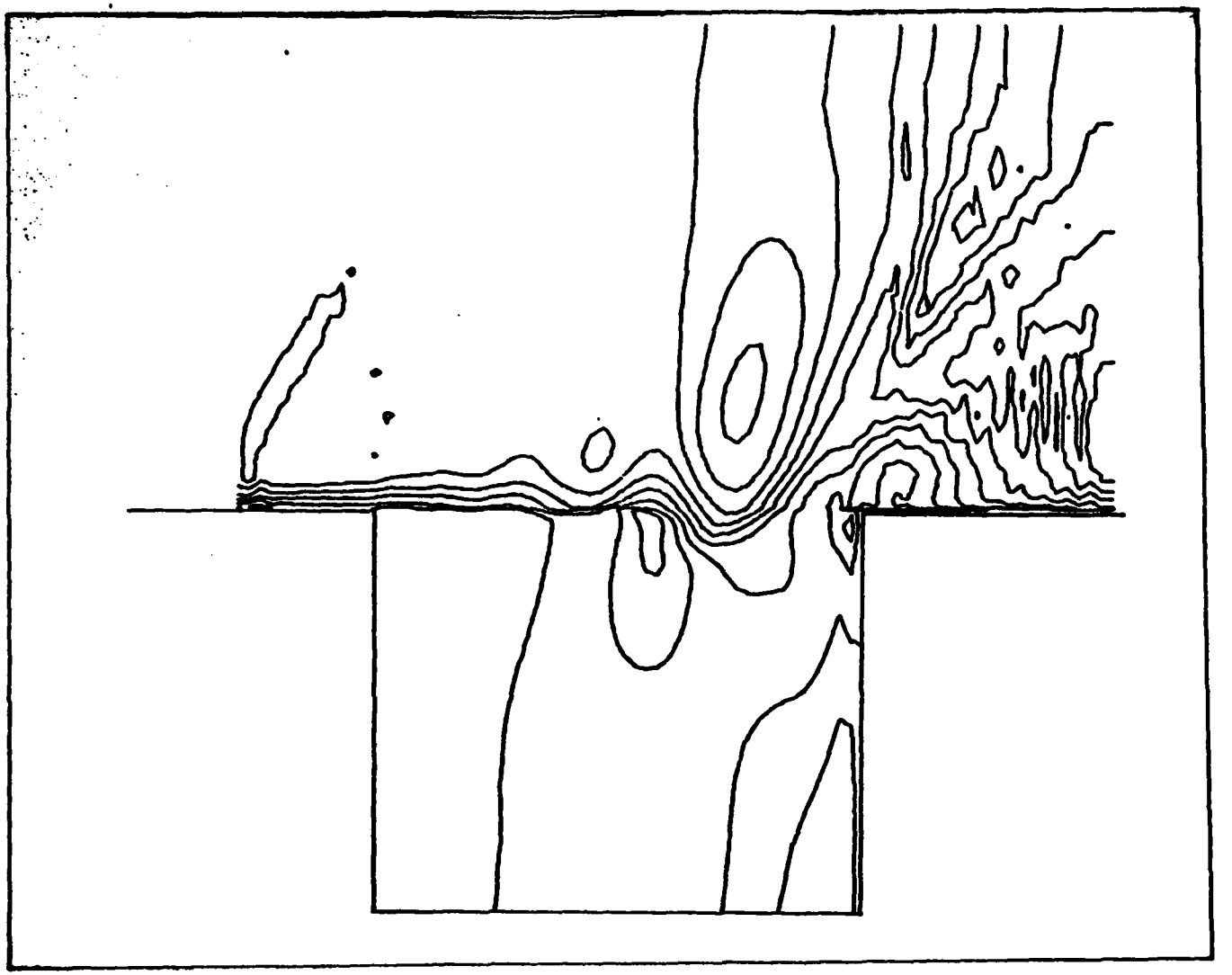


Figure 9. 2-D Cavity Contour Lines, Frame 100

APPENDIX D

PROGRAMMER'S GUIDE

This appendix clarifies the organization of CINEMA and specifies the function of each module in the system. This will hopefully be beneficial to a maintenance programmer. First, a general overview is given; second, structure charts for the system are provided; and finally, a brief description of the function of each CINEMA routine is presented.

CINEMA uses the LR language processing system to read and interpret CINEMA's command language, and it uses the DISSPLA graphics package to do the computer graphics work. The CINEMA subroutines can be grouped into two categories: parsing routines and application routines. The parsing routines are part of the LR package; they process the user commands. More information about the LR system is contained in Appendix A. The application routines read the user's input data and prepare it for the graphics output. The application routines then call DISSPLA routines to generate the graphics output. There is one additional fact that should be explained. Some of the application routines do unusual variable and array handling. Within CINEMA's blank common there is a very large block of core set aside to be used as a work area. The user describes the input data to be graphically displayed and chooses the technique to be used to display the data. Rather than set aside fixed size arrays and allocate variables at FORTRAN compile time, CINEMA allows the user much more flexibility by doing its own allocation of that work area of core at execution time.

This requires CINEMA to have its own symbol table and consequently, CINEMA is required to do its own accessing of those variables and arrays. However, since CDC FORTRAN uses call by address for subroutine arguments, often CINEMA "locates" the address of an array within the large work area and passes the address to a subroutine. FORTRAN will do the accessing of CINEMA defined array if it is passed as an argument in a subroutine call.

What immediately follows is a set of structure charts for the CINEMA system; they show the organization of the routines within the system. After the structure charts is a list of the routines and a brief description of their functions.

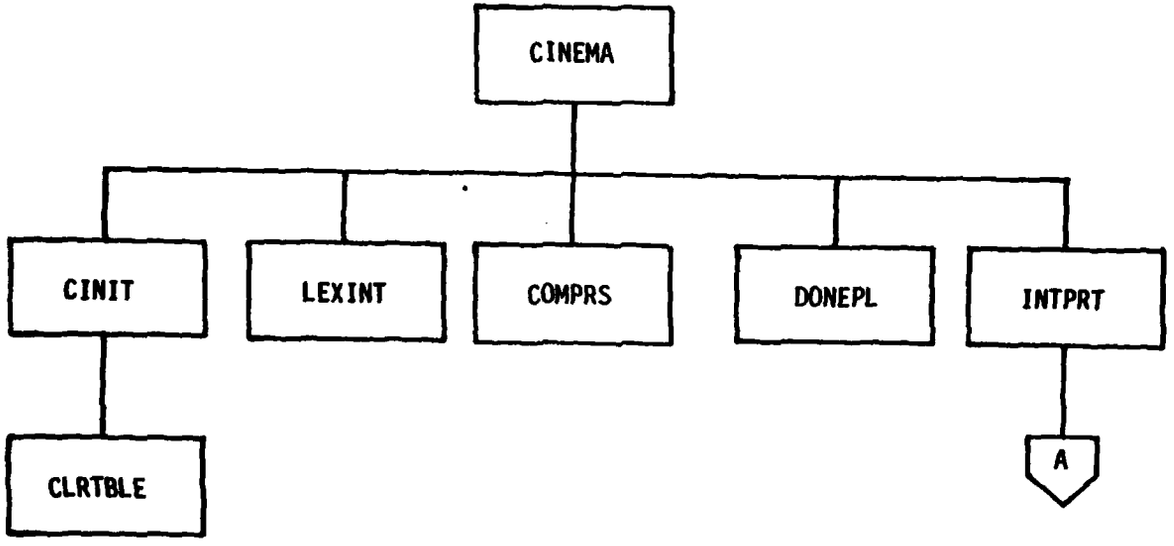


Figure 10. CINEMA

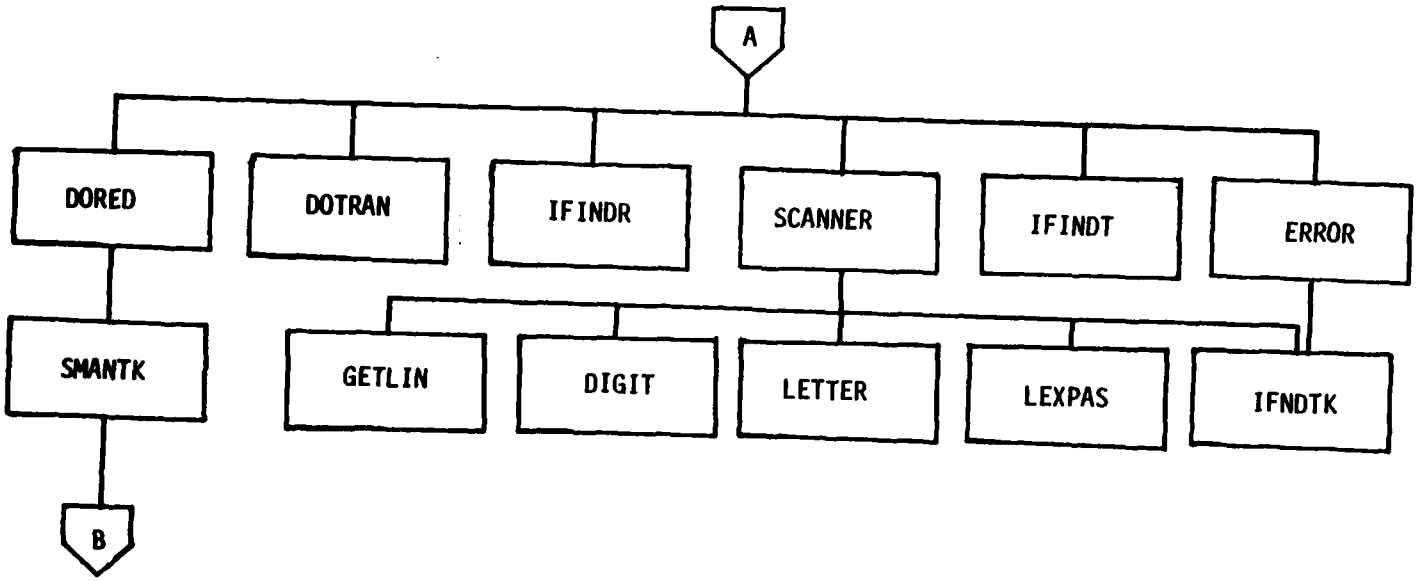


Figure 11. Parsing Routines

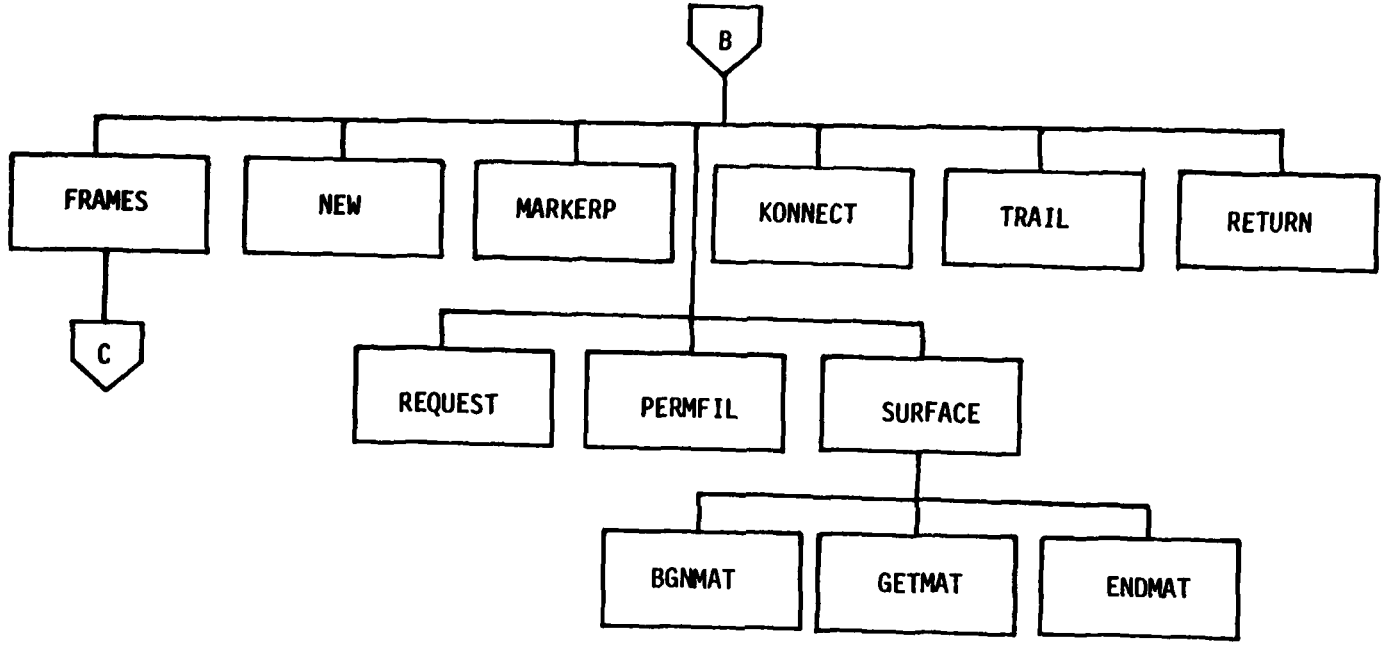


Figure 12. Application Routines

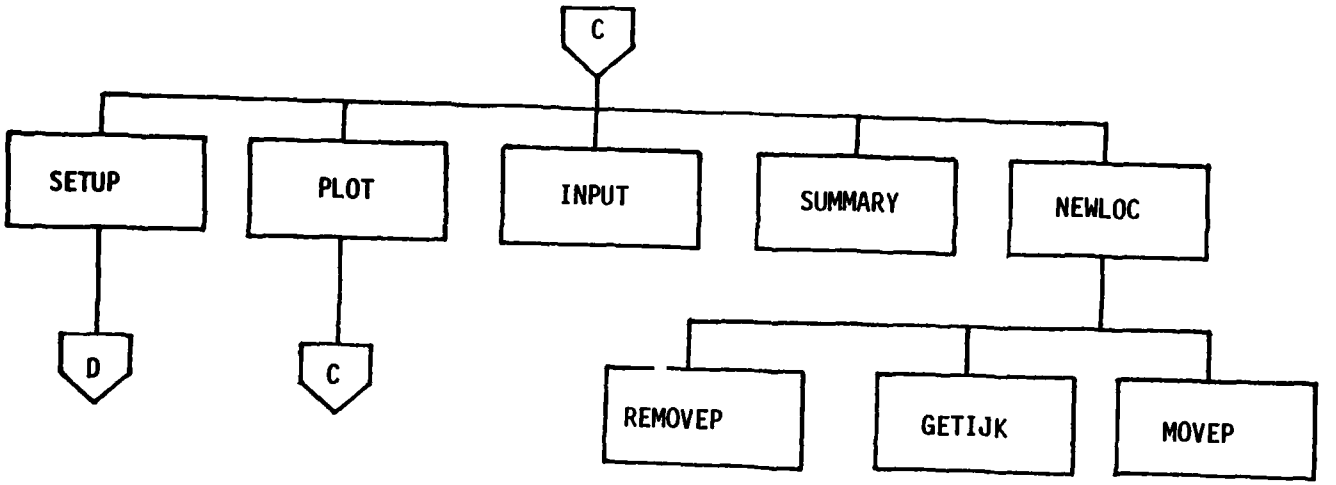


Figure 13. Application Routines

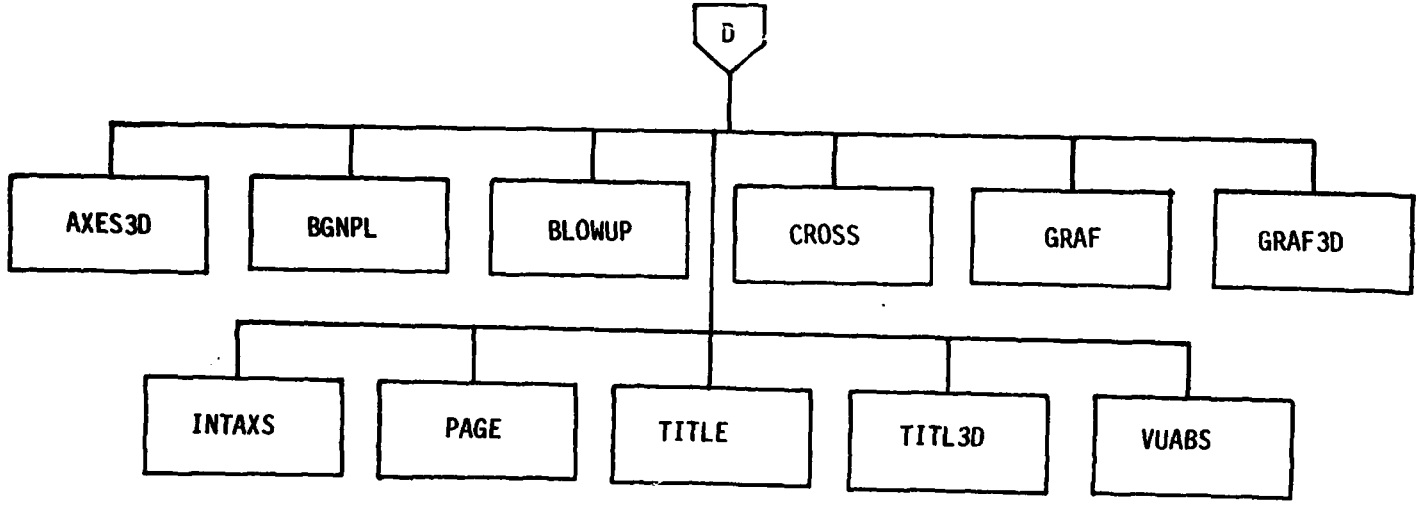


Figure 14. Graphics Routines

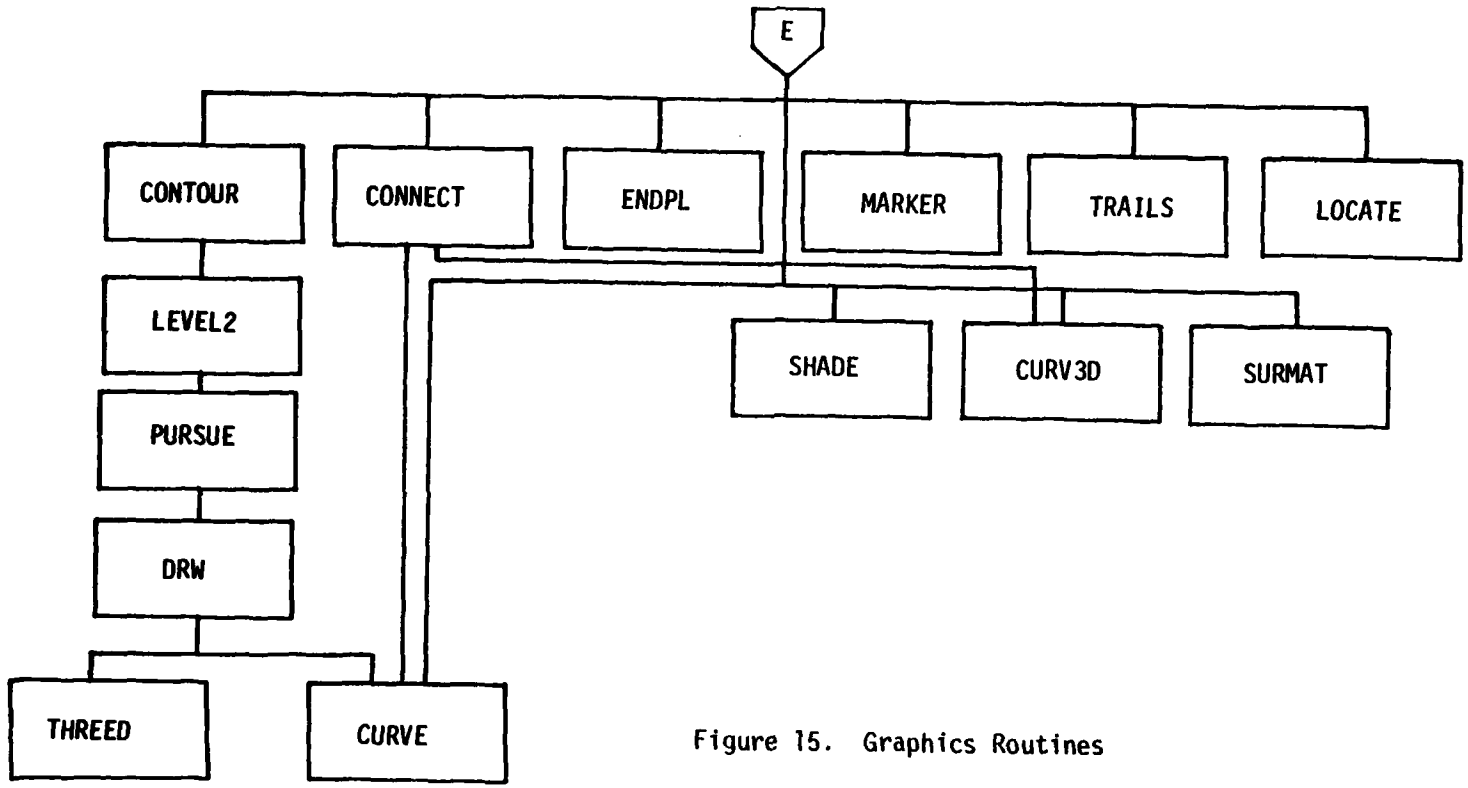


Figure 15. Graphics Routines

ROUTINE	FUNCTION
CERROR	Error printing routine for the application routines. Each error is assigned a number, error numbers over 100 are fatal and CERROR stops after printing the message.
CINEMA	This is the main routine of the system; it does initialization and then calls INTPRT to begin parsing the input commands.
CINIT	CINEMA blank common initialization. Variables in the blank common must be preset using assignment statements.
CLRTBLE	Clears CINEMA's symbol table (the table used for allocating variable storage within the large blank common array). Routine courtesy of Major Michael Wirth
CONNECT	Searches the particle arrays and "draws" any connections.
CONTOUR	The controlling subroutine for 2-D contour plots.
DIGIT	Parsing routine. Decides if input hollerith character is a digit (very CDC dependent).
DORED	Parsing routine. This performs a reduction according to a language production.
DOTRAN	Parsing routine. Transitions the parser to a new state. Parsing stacks are maintained here.
DRW	One of the 2-D contouring routines.
ERROR	Error routine for the parsing routines. Very simplistic error recovery is used here; it "backs up" to a "solid token" (which is statement list) and then returns to calling routine. The parsing routines will try to continue the parse.
FRAMES	This routine is called when movie frames are to be created. It coordinates their generation and controls all activity until the requested frames have been done.
FINDXYZ	Given a mesh location, I, J or I, J, K, this returns the corresponding X, Y or X, Y, Z values. A separate routine is used to cause FORTRAN to access CINEMA allocated arrays.

ROUTINE	FUNCTION
GETLIN	Gets the next input line. It adds an end-of-line character, a semicolon, if one was not provided.
GETIJK	Given a particle location in cartesian coordinates this routine searches the location arrays and returns the indices of the point in the mesh closest to the particle location.
IFINDR	Parsing routine. Given the current state of the parser and the next look-ahead token it determines if a reduction is possible.
IFINDT	Parsing routine. Given the current state of the parser and the next look-ahead token it determines if a transition is possible.
IFNDTK	Parsing routine. Searches the vocabulary table for a character string. If found, it returns the string's location within the vocabulary table, otherwise a -1 is returned.
INPUT	Reads data from the flow field data file using a FORTRAN BUFFER IN statement. The flow field file is assumed to be unit 7.
INTPRT	This is the actual parsing loop. It tries to perform a reduction first. If that fails, it then tries a transition. If that fails, it has found an incorrectly formatted command and calls ERROR.
KONNECT	Loads particle numbers into the particle connect arrays when a connection is defined.
LETTER	Decides if input hollerith character is a letter (very CDC dependent).
LEVEL2	One of the 2-D contouring routines.
LEXINT	Parsing routine. It does lexical initialization and presets variables needed by the lexical analysis routines and the semantic routine (SMANTK).

ROUTINE	FUNCTION
LEXPAS	Parsing routine. After SCANNER has found the next token, LEXPAS returns the token identifier (integer value), string, and numeric value to the routine that called SCANNER by placing these items in variables in a labelled common.
LOCATE	Searches for a variable that has been allocated space by CINEMA. If it is an indexed variable (an array) it returns the maximum values for the indices.
LOOKUP	Searches CINEMA's symbol table for a variable. The search is performed using a hashing scheme. Routine courtesy of Major Michael Wirth.
MARKERP	Loads the particle arrays when a new particle is defined.
MOVEP	This routine calculates the new location of a particle given its old location and the X, Y and Z components of velocity.
NEW	Allocates space within the large array in blank common. It is capable of equivalencing variables and causing variables to occupy contiguous locations within the large array in common.
NEWLOC	Coordinates all the activities involved in updating particle locations. Not only is the new location calculated (using MOVEP) but arrays for use in trailing particles must be updated.
PLOT	This routine is called to plot all the information in one of the additive primary colors. It controls all the plotting of data done by CINEMA.
PRNTSTK	Print the stacks. A utility routine that can be used for debugging. It prints the current values on the parsing stacks.
PURSUE	One of the 2-D contouring routines.

ROUTINE	FUNCTION
REMOVEP	NEWLOC checks each new particle location against the range of coordinate values specified by the user. If the new location is outside those bounds, then REMOVEP is called to remove all references to that particle.
SCANNER	Gets the next lexical item (token) in the input stream.
SETUP	Makes all the necessary calls to DISSPLA routines to prepare for the plotting.
SMANTK	Parsing routine. This performs any necessary action required with each production in the language. The production number in the call corresponds to one of the numbered productions in the list of productions produced by the LR package during parse table generation. This routine is the link between the parsing routines and the application routines.
SUMMARY	After each set of frames is produced this routine provides a list of the actions taken by CINEMA.
SURFACE	Reads the user's surface file (unit 8) and prepares the surface for plotting.
THREED	One of the 2-D contouring routines.
TINIT	Initializes the arrays used for trailing particles.
TRAIL	Loads trail arrays with particle numbers; called when a TRAIL command is processed.
TRAILS	Draws trails for particles; called during frame generation.

VITA

Elton Philip Amburn was born on February 19, 1950 in Memphis, Texas. He graduated from Roosevelt High School in Emporia, Kansas in 1968 and received a Bachelor of Arts Degree in Physics and Mathematics from Kansas State Teachers College in May 1972. Shortly after graduation from college he enlisted in the USAF and he was eventually assigned to Offutt AFB, Nebraska as a computer programmer. In August 1974 he entered Officer's Training School at Lackland AFB, Texas and graduated and received his commission in October 1974. After commissioning he served as a computer programmer and systems analyst at the NORAD Cheyenne Mountain Complex at Colorado Springs, Colorado from November 1974 to May 1978. In June 1978 he entered the School of Engineering, Air Force Institute of Technology to pursue a graduate degree in computer systems.

Permanent address: 817 Waverly Way
Emporia, Kansas 66801