# You Want to Produce 83,000 Graphs using SAS® on a PC? You Gotta be Kidding!

Mark Bercov, Bercov Computer Consultants Ltd., Calgary, Alberta
Ned Etris, Canadian Hunter Exploration Ltd., Calgary, Alberta

## ABSTRACT

Canadian Hunter Exploration Ltd. has created a database/graphical evaluation system which characterizes the production performance and reservoir rock quality of all of the oil pools in Alberta, based entirely on SAS 6.04 on the PC. The system consists of two parts: a set of separate programs that process oil well and oil pool data, combining them into a strictly pool-based database, and a set of integrated programs which create one textual report and forty graphs per pool.

Development of this system was truly an adventure. Creation of the database involved consolidation of raw data from different sources for over 54,000 wells and over 8,000 oil pools, as well as correction of errors and inconsistencies in the data, and calculation of the appropriate statistical summary information from these data. Creation of the final one page report and forty graphs for each pool presented the opportunity to develop workarounds to several deficiencies in SAS/GRAPH®, major memory allocation problems in SAS 6.04, and the design of an automated, interruptable, self-recoverable process of creating the desired output. Techniques utilized to successfully achieve these objectives will be discussed in this paper.

## INTRODUCTION

The objective of the research project under discussion was the creation a prototype (rather than production) system to output hard copy reports and graphs displaying production performance and rock quality information for all of the oil pools in Alberta. The project was limited to the use of existing hardware and software at Canadian Hunter. SAS 6.04 was selected as the programming environment for this system due to its database management, analytical, and graphical capabilities, and most specifically its ability to readily create graphs quickly and easily in order to assess their suitability, as well as the fact that it was available at Canadian Hunter. At the start of the project, samples of 20 desired graphs had been prepared. One of the objectives of the project was the evaluation of additional graphs, and 20 additional graphs were selected from the many additional graphs considered for inclusion in the final output. In order to maintain a reasonable volume of printed output (and preserve at least some of the forests of North America), it was decided to output 10 graphs per page utilizing PROC GREPLAY templates (see Figure 5), thereby limiting the output to five pages per pool. Even so, the final production runs generated 19,238 pages of printed output!

A full explanation of the nature of the data, as well as how and why they were input into SAS, and a description of the techniques utilized to create the printed reports, is provided in the companion paper **You Want to Analyze All of the Oil Production Data for 90% of Canada Using SAS® Software on a PC? You Gotta be Kidding!**

This paper is the second of two relating the challenges encountered during this project. It describes some of the enhancements which were added to bypass SAS/GRAPH's defaults and to circumvent some bugs, and relates the unique requirements dictated by the large number of graphs to be created and printed.

## PROCESSING TECHNIQUE

The system created had to be capable of running unattended, because the number of pools to be printed was so large that it was necessary to run the program all day and all night on several computers for several months. If the program stopped for any reason (such as power outage, computer malfunction, or program error), it had to be capable of being restarted without repeating or losing any output. Additionally, the user had to have the ability to specify the pools desired for processing during the current run so that several computers could be utilized simultaneously.

Obviously, there were two basic approaches which could be taken to the processing of the data. The first approach involved the selection of the next pool of interest, complete processing of all of the data for this pool, output of the printed report and graphs for the selected pool, and looping back to repeat this process for the next pool. The second possibility was the creation and storage of graphical output for all of the pools, followed by output of the desired reports and hardcopy graphics for all of the pools. The first procedure was selected for the following reasons:

1. The hundreds of gigabytes of disk storage required to retain all of the graphics output to be produced were not available.
2. No reliable method could be devised for the assignment of unique graph names to all of the graphs produced.
3. This processing method would require two passes through all of the input data, one to create the graphs and one to create the printed reports.

A loop control SAS Macro Program was created to supervise the processing (as this is the only procedure in

SAS which would allow iterative processing of one pool at a time). This SAS Macro program invoked other SAS Macro programs to perform the actual processing of the data. The first of these programs determined the next pool of interest, and retrieved all of the data for this pool.
The next Macro Program invoked created and output the Pool Summary Sheet. This was followed by a Macro Program which performed all of the calculations necessary for the creation of the desired graphs. A further Macro Program created these graphs, storing the output in a graphics catalog. Yet another Macro Program was invoked to output the graphs on the printer. Finally, a cleanup Macro Program (which also output a message on the console and the Log File describing the Pool which had been processed and the time required for this processing) was invoked.

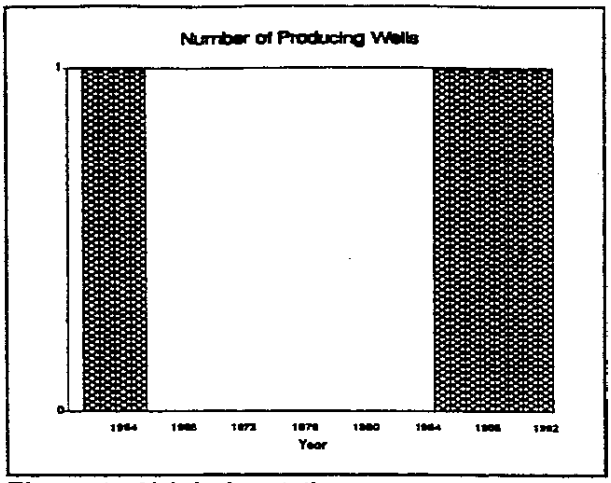

**Figure 1** - X-Axis Annotation

## AXIS ANNOTATION OPPORTUNITIES

The SAS/GRAPH defaults for axis scaling and major tick mark annotation generally were not acceptable for this project. The most obvious example is the labeling of axes representing time measurements, such as years (see Figure 1). Because time had to be treated as a continuous (rather than discrete) variable, often annotated values such as 1983.5 appeared on the axis. If there were a large number of years present, the bars were annotated vertically rather than horizontally, thereby severely reducing the useful graphics area. Finally, the default spacing of the year values was inadequate, with too many years being displayed when the number of years was very large. This meant an algorithm to calculate axis tick mark values to be annotated had to be developed for many of the graphs. The procedure which was adopted was really quite straightforward. A SAS Format was created to translate the range of values present into an axis increment value. The floor value for the axis was calculated as the largest multiple of the increment which was less than the minimum data value detected. The ceiling value for the axis was calculated as the smallest multiple of the increment which was greater than the maximum data value detected. The floor, increment, and ceiling values thus calculated were used to create an ORDER clause, which was stored in a symbolic macro variable, and expanded when the appropriate AXIS statement was processed.

A second problem encountered related to the limits of the axis. In some cases, particularly with logarithmic axes, it was decided to trim certain extreme values, low or high, in order to make the plots readable (see Figure 2). For example, an important measure of the quality of an oil pool is the gas/oil ratio (GOR). Low values of this ratio are extremely low, near zero. But values of GOR in a gas pool that produced a little oil (of which there are many) can be extremely large, in the millions, and nearing billions. Because this ratio was plotted logarithmically, the possibility existed of plotting too many log cycles, since each power of 10 generates a new cycle. However, it is not necessary to display the extremely low values and extremely high values on the same graph. Accordingly, the system was designed to determine the appropriate data range to be displayed, utilizing a somewhat complicated procedure which will not be described in detail.

While speaking of logarithmic axes, another major point is that good practice requires plotting a minimum of two, and preferably three log cycles, regardless of the actual range of the data in the sample. This number guarantees that the plot will always be distinguishable from a linear plot, even if the minor tick marks are invisible (as was the case when the graphs were replayed 10 per page). Since SAS/GRAPH does not guarantee that at least three log cycles will be displayed, it was necessary to insert code which ensured that at least three log cycles were displayed, and that appropriate cycles were chosen so as to make the graph readable.

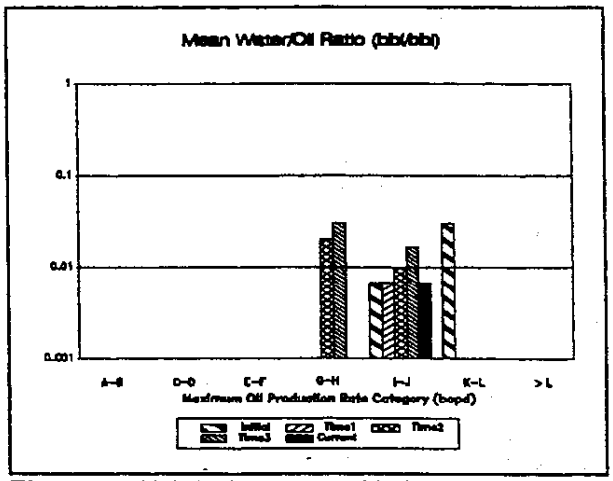

**Figure 2** - Y-Axis Annotation, Variable Formats, and Missing Categories

A third problem related to the width of the values annotated at the Y-Axis major tick marks. Given that there is a fixed

width for the plot for a given output device, the larger the number, the wider the annotation, and the smaller the most important part, the graph, becomes. This is not a trivial problem. In the case of both gas/oil ratio and water/oil ratio, there are pools where the ratio is extremely small. For very small values, two or more places have to be displayed to the right of the decimal point. In some cases, however, the values to be displayed are extremely large, in the hundred of thousands. If a fixed format was selected for the display of tick mark values along the Y-Axis, it would be necessary to leave at least 10 positions for this annotation, thus shrinking the effective graphics display area significantly. Accordingly, the data to be displayed on each graph were analyzed dynamically, and the most appropriate format for axis annotation was selected, stored in a symbolic macro variable, and utilized by a FORMAT statement in the PROC GCHART procedure.

## MISSING DATA CHALLENGE

One of the biggest hurdles was dealing appropriately with missing data. The three missing data situations below all had to be handled correctly.

1. Missing data for one category in a graph, but not all.
2. Entirely missing data for a given graph.
3. Discontinuous data for a line graph.

The first case of missing data was the case where data for one or more categories, but not all, were missing (see Figure 2). It was desired that all possible categories be displayed, even if there was nothing in one or more of these categories. Unfortunately, SAS/GRAPH, by default, will only display non-missing categories. However, one of the main objectives of this project was to be able to compare and contrast pools by looking at the same graphs across many pools. This is extremely difficult to do unless there is some consistency to the output format. Accordingly, the system had to determine whether there were missing data for a given category, but data available for others, and force an empty category to appear. It was critical that this situation be distinguished from the situation in which there were no data for any category. Additionally, the system had to ensure that it did not alter any calculations by forcing an empty category, so the raw input dataset could not just be littered with zero values. The final system dynamically created false zero values just for the purposes of creating graphics output, but went to great lengths to differentiate these false zero values from real zero values coming from the input dataset.

The second case of missing data was the case in which all of the data for the graph were missing. In this case, the SAS/GRAPH default practice is the production of a warning message on the LOG, and no creation of graphics output. It was not acceptable to put up with a blank spot on the page where a graph should be, because it would suggest that there was an error in the program. However, entirely missing data for one graph for one pool is only not an error, it is to be expected when dealing with so many pools. It was decided to detect the situation in which all of the data for any given graph were missing, and to create a specially designed graph with a message stating that there were no data available (see Figure 3.)

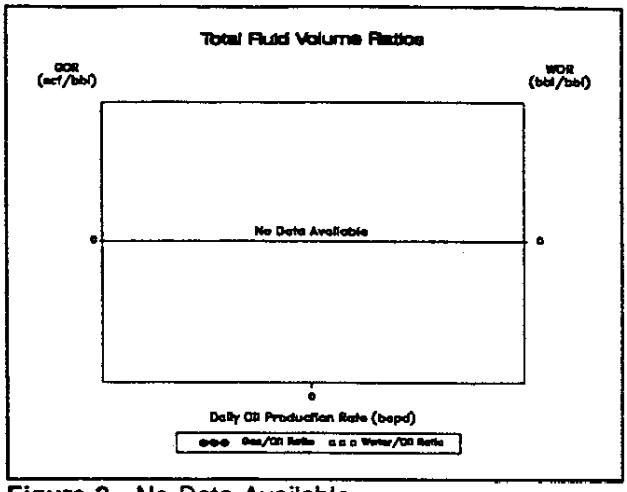


**Figure 3** - No Data Available

The final missing data situation involved discontinuous data for line plots. Once these plots were squeezed down to fit 10 on a page, it became impractical to plot symbols at the data points to differentiate between different variables on the plot. Either the symbols were large enough to be seen, but they were so big that they obscured the line and overlapped each other, making the plot unreadable, or they were so small that they could not be seen at all. How then could single (discontinuous) points be plotted on a line graph so that they could be visible? Since these points were discontinuous, they were not joined by the selected line to other points. Dropping the symbols altogether and including only lines worked fine for pools with lots of continuous data, but then the significant number of plots with only one or two points would not display anything. The solution to this problem was to recognize those plots which had discontinuous data and use appropriately sized symbols in those cases, and to use lines with symbols displayed at a height of 0.01 PCT for those graphs with continuous data (see Figure 4).

## GRAPHICS STREAM OUTPUT CONSIDERATIONS

It soon became apparent that the choice of the HPLJ300 graphics driver presented significant performance problems to the system.

The four pages of graphics output created for each pool required approximately 5 minutes to rasterize, and required an additional 5 minutes to transmit to the HP Laserjet printer via the parallel port on the computer. While this output was being sent to the printer, no processing was performed on the computer. An alternative solution was obviously required. Unfortunately, resolution of this problem was not as easy as it looked. Creation of HPGL output was not viable because there were no printers available which could accept an HPGL graphics stream.

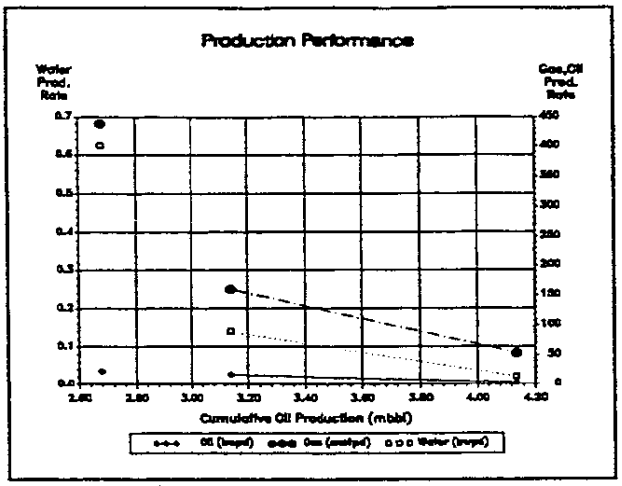

**Figure 4** - Discontinuous Data

The use of data compression techniques, such as run length encoding and packbits encoding, was not possible because they are not supported on the HP Laserjet II printers available to the project. Attempts to use a network printer were not overly effective once it was realized that the network server took over 20 minutes to transmit each page of graphics to the attached printer. At this rate, printing would have taken over 200 days, assuming that we could have kept the network up for that long. Installation of a high speed network interface card connected to a printer was not supported by the Information Services Department (which was in the throes of installing completely new network software at the time). Acquisition of high speed parallel ports on the computers being utilized to run this system violated the edict that only existing hardware be used. Attempts to utilize hardware character sets were defeated by bugs in SAS 6.04. Finally, the use of Postscript was precluded by the fact that crosshatch patterns all merged into an almost solid black mat when the graphs were replayed 10 per page. Undaunted by the minor setbacks noted above, we persevered, and ultimately came up with a viable solution. Installation of PrintCache™ from LaserTools Corporation allowed the simultaneous output of printout via the parallel port (in DOS background mode) and continued processing of the following pool (in DOS foreground mode), with a minor penalty of approximately 10% on processing time during the print despooling process.

## THE FINAL CHALLENGE

It now appeared that all of the opportunities for innovative thinking presented by this project had been successfully resolved. A looping mechanism had been implemented and tested, axis annotation problems had all been addressed, missing data had been handled, and graphics output was under control. Accordingly, the system was fired up, and started producing output. However, it soon became apparent that every iteration of the loop took longer and longer, until the system eventually died and returned to the DOS prompt (without processing all of the selected pools). What was happening? After a week of extensive analysis, the CDE and CDM commands finally provided the answer. A portion of memory below the line was being allocated during each iteration of the loop (by some unknown process), and not properly released after use. Accordingly, the amount of memory available for rasterization of the final graphics output got smaller and smaller, and the rasterization process took longer and longer. Finally the memory fragmentation became so bad that the DATA step compiler generated erroneous diagnostics and terminated the process. Technical Support at the SAS Institute was unable to shed any light on the actual cause of this problem.

The solution to this problem was quite complex, and took over one month to implement. It required the creation of several additional SAS Macro Programs, and significant modifications to the DOS Batch Files utilized to execute the system. One Macro Program kept track of which pools had been completely processed, another monitored performance and terminated the SAS processing when the time required for the current iteration was more than 1.5 times the time required for the first iteration, and others performed various necessary maintenance functions and invoked new DOS Batch Files to set DOS environment variables to the appropriate values.

A major loop was added to the DOS Batch File which invoked this system. This DOS Batch File initiated operation by utilizing the Norton Utilities® Batch Enhancer ASK command to ask the user whether or not the Pool selection list had been changed. (For the uninitiated, the ASK command allows the developer to output a message on the console, to specify a set of valid responses, and to specify a default time after which a predefined response will be returned to the system. It would be nice if SAS had a similar capability.) If the user responded YES, a SAS program to reset all of the file pointers to the start of the files was executed. Normal SAS processing for this system was then invoked. Each time that SAS terminated (for any reason) and returned control to the DOS Batch File, the SAS LOG File was captured and appended to a master log file, and it was determined whether or not all of the selected Pools had been processed. If so, processing was terminated. If not, the Norton Utilities ASK command was utilized to query the user whether or not he wished to terminate execution of the system (with a default answer or No being given after 15 seconds). If the answer to this question was YES, the system terminated. If the answer to this question was NO, control looped back to the point at which normal SAS processing for this system was invoked.

## CONCLUSION

The successful completion of this project provides ample proof of the power and utility of the SAS System on the PC, even under DOS. This project required the processing of extremely large datasets and the creation and output of significant amounts of printed output. Despite the problems encountered, total development of this system required only

approximately 18 man-months of programming effort. Production of the final output under SAS 6.04 took approximately 3 months on a 486/33 DX System. (A quick and dirty conversion to SAS 6.08 under Windows 3.1 resulted in a reduction in processing time of approximately 50%.)

The primary strengths of the SAS System demonstrated by this project are noted below.

1. It executed well on the available PC hardware and desktop laser printers.
2. It demonstrated the ability to effectively process large amounts of data on a desktop system.
3. It provided an excellent prototyping tool, allowing the rapid testing of different procedural methods, dealing with many changes in raw data structure and content, and evaluating different kinds of graphs.
4. It provided the capability of quickly creating large amounts of production quality graphical output.

## ACKNOWLEDGEMENTS

The authors wish to thank Canadian Hunter Exploration Ltd. for the opportunity to publish this paper and to present this material at SUGI 19. Special thanks go to my co-author, Ned Etris, for his patience and assistance in the preparation of this paper. The figures in this paper have been altered slightly to protect proprietary information.

Author Contacts:

Mark Bercov
Bercov Computer Consultants Ltd.
3641 - 13 St. S. W.
Calgary, Alberta, Canada
T2T 3R2
(403) 243 - 0686

Ned Etris
Canadian Hunter Exploration Ltd.
#2000, 605 - 5 Ave. S. W.
Calgary, Alberta, Canada
T2P 3H5
(403) 260 - 1183

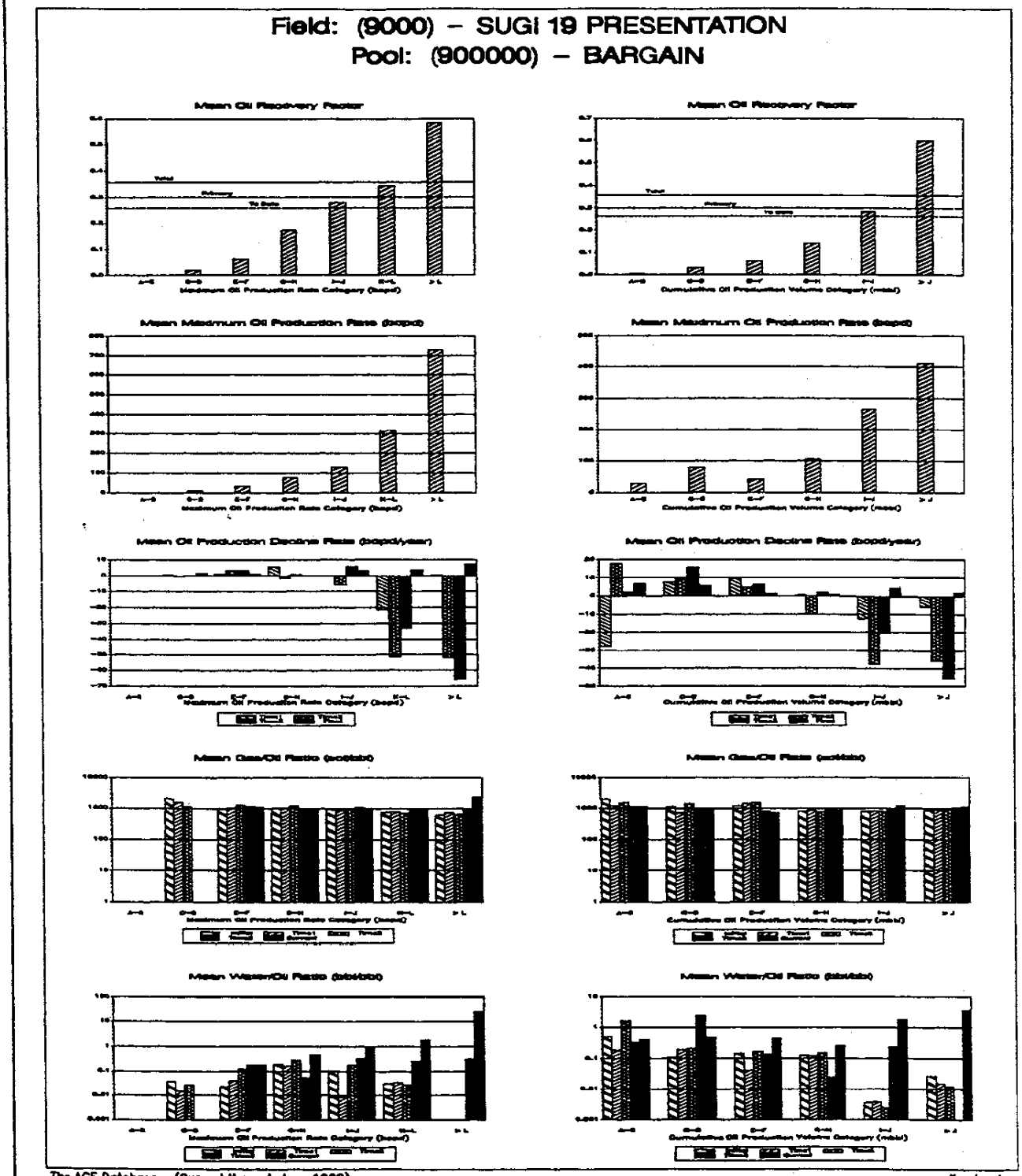


**Figure 5** - Sample Graphics Output Page